The $26^{\text {th }}$ Annual International Conference on Information Security and Cryptology

# ICISC 2023 

November 29 (Wed) ~ December 1 (Fri), 2023
KOREANA HOTEL, Seoul, Korea | Hybrid(on-off mix) Conference

## Conference Program

Hosted by
Korea Institute of Information Security \& Cryptology

## Sponsored by

(1) NIS NSR ETRI \$KISA KISTitemm

든 SmartM2M
스마트엠투임

UML
COONTEC ESTsecurity

## Contact Information

○ General Information
Young-Ho Park (Sejong Cyber University, Korea)
Email: youngho@sjcu.ac.kr
Junbeom Hur (Korea University, Korea)
Email: jbhur@korea.ac.kr

OTechnical Program
HwaJeong Seo (Hansung University, Korea)
Email: hwajeong84@gmail.com
Suhri Kim (Sungshin Women's University, Korea)
Email: suhrikim@sungshin.ac.kr

O Registration
Korea Institute of Information Security \& Cryptology(KIISC)
Email: kiisc@kiisc.or.kr

## Table of Contents

Messages from Program Chairs ..... 4
Organization ..... 5
Conference Program ..... 8
Conference Information ..... 11
Invited Talks ..... 19
Papers ..... 23

## Messages from Program Chairs

International Conference on Information Security and Cryptology (ICISC 2023) is held from November 29 - December 1, 2023. This year's conference is hosted by the KIISC (Korea Institute of Information Security and Cryptology).

The aim of this conference is to provide an international forum for the latest results of research, development, and applications within the field of information security and cryptology. This year, we received 78 submissions and were able to accept 31 papers at the conference. The challenging review and selection processes were successfully conducted by program committee (PC) members and external reviewers via the EasyChair review system. For transparency, it is worth noting that each paper underwent a blind review by at least three PC members. For the LNCS post-proceeding, the authors of selected papers had a few weeks to prepare for their final versions, based on the comments received from the reviewers.

The conference features three invited talks, given by Prof. Rei Ueno, Dr. Tung Chou, and Dr. Anubhab Baksi. We thank the invited speakers for their kind acceptances and respectable presentations. We would like to thank all authors who have submitted their papers to ICISC 2023, as well as all PC members. It is a truly wonderful experience to work with such talented and hardworking researchers. We also appreciate the external reviewers for assisting the PC members. Finally, we would like to thank all attendees for their active participation and the organizing members who successfully managed this conference. We look forward to seeing you again at next year's ICISC.

November 2023
HwaJeong Seo, Suhri Kim
Program Chairs

## Organization

## GENERAL CHAIR

- Yoojae Won (Chungnam National University, Korea)


## ORGANIZING CHAIRS

- Young-Ho Park (Sejong Cyber University, Korea)
- Junbeom Hur (Korea University, Korea)


## ORGANIZING COMMITTEE

- Daewan Han (National Security Research Institute, Korea)
- Hyun-O Kwon (Korea Internet \& Security Agency, Korea)
- Jeong Nyeo Kim (Electronics and Telecommunications Research Institute, Korea)
- Jungsuk Song (Korea Institute of Science and Technology Information, Korea)
- Kihyo Nam (UMLogics, Korea)
- Jonghwan Park (Sangmyung University, Korea)
- Jongsung Kim (Kookmin University, Korea)
- Youngjoo Shin (Korea University, Korea)
- Dongyoung Koo (Hansung University, Korea)
- Changhee Hahn (Seoul National University of Science and Technology, Korea)
- Hyunsoo Kwon (Inha University, Korea)


## PROGRAM CHAIRS

- Hwa Jeong Seo (Hansung University, Korea)
- Suhri Kim (Sungshin Women's University, Korea)


## PROGRAM COMMITTEE

-Wenling Wu (Institute of Software Chinese Academy of Sciences, China)

- Zhenfu Cao (East China Normal University (ECNU), China)
- Swee-Huay Heng (Multimedia University, Malaysia)
-Taehwan Park (National Security Research Institute, Korea)
- Toshihiro Yamauchi (Okayama University, Japan)
- Jiqiang Lu (Beihang University, China)
- Joonsang Baek (University of Wollongong, Australia)
- Katsuyuki Okeya (Hitachi High-Tech Corporation, Japan)
- Keita Emura (Kanazawa University, Japan)
- Bimal Roy (ASU, Indian Statistical Institute, India)
- Dongseong Kim (University of Queensland, Australia)


## Organization

- Donghoon Chang (IIIT-Delhi, India)
- Hung-Min Sun (National Tsing Hua University, Taiwan)
- Iraklis Leontiadis (Inpher, USA)
- Baodong Qin (School of Cyberspace Security, Xi'an University of Posts and Telecommunications, China)
- Xinyi Huang (Fujian Normal University, China)
- Sherman S. M. Chow (Chinese University of Hong Kong, Hong Kong)
- Daniel Slamanig (Universität der Bundeswehr München, Germany)
- Reza Azarderakhsh (Florida Atlantic University, USA)
- Ben Lee Wai Kong (Gachon university, Korea)
- Anubhab Baksi (Nanyang Technological University, Singapore)
- Olivier Sanders (Orange Labs, France)
- Kwangsu Lee (Sejong University, Korea)
- Munkyu Lee (Inha University, Korea)
- Jooyoung Lee (KAIST, Korea)
- SeogChung Seo (Kookmin University, Korea)
- Jaehong Seo (Hanyang University, Korea)
- Jihye Kim (Kookmin University, Korea)
- Jongsung Kim (Kookmin University, Korea)
- Aaram Yun (Ewha Woman's University, Korea)
- Taekyoung Youn (DanKook University, Korea)
- Jungyeon Hwang (Sungshin Women's University, Korea)
- Minhye Seo (Duksung Women's University, Korea)
- Dohyun Kim (Catholic University of Pusan, Korea)
- Seongmin Kim (Sungshin Women's University, Korea)
- Haehyun Cho (Soongsil University, Korea)
- Myungseo Park (Kangnam University, Korea)
- Dongyoung Koo (Hansung University, Korea)
- Hyojin Jo (Soongsil University, Korea)
- Wonsuk Choi (Korea University, Korea)
- Daehee Jang (Kyunghee University, Korea)
- Yeonjoon Lee (Hanyang University, Korea)
- Jonghwan Park (Sangmyung University, Korea)
- Seungkwang Lee (Dankook University, Korea)
- Yongwoo Lee (Inha University, Korea)
- Konwoo Kwon (Hongik University, Korea)
- Youngjoo Shin (Korea University, Korea)


## Organization

- Ilgu Lee (Sungshin Women's University, Korea)
- Joohee Lee (Sungshin Women's University, Korea)
- Joonwoo Lee (Chungang University, Korea)
- Dongyoung Roh (National Security Research Institute, Korea)
- Changmin Lee (Korea Institute for Advanced Study, Korea)
- Heeseok Kim (Korea University, Korea)
- Seunghyun Park (Hansung University, Korea)
- Kiwoong Park (Sejong University, Korea)
- Sokjoon Lee (Gachon University, Korea)
- Byoungjin Seok (Seoul National University of Science and Technology, Korea)
- Taejin Lee (Hoseo University, Korea)
- Donghyun Kwon (Pusan National University, Korea)
- Kyunbaek Kim (Chonnam National University, Korea)
- Dooho Choi (Korea University, Korea)
- Seongkwang Kim (Samsung SDS, Korea)
- Jihoon Kwon (Samsung SDS, Korea)
- Seunghyun Seo (Hanyang University, Korea)
- Namsu Chang (Sejong Cyber University, Korea)


## Conference Program

| Wednesday (2023-11-29) |  |
| :---: | :---: |
| $\begin{aligned} & \text { KST 09:30-09:40 } \\ & \text { UTC 12:30-12:40 } \end{aligned}$ | Opening Remarks |
| $\begin{aligned} & \text { KST 09:40-11:00 } \\ & \text { UTC 12:40-02:00 } \end{aligned}$ | Session 1 : Cryptanalysis \& Quantum Cryptanalysis 1 (Session Chair : Prof. Suhri Kim (Sungshin Women's University)) |
|  | Enhancing the Related-Key Security of PIPO through New Key Schedules Seungjun Baek, Giyoon Kim, Yongjin Jeon and Jongsung Kim |
|  | Optimized Quantum Implementation of SEED <br> Yujin Oh, Kyoungbae Jang, Yu-Jin Yang and Hwajeong Seo |
|  | Depth-Optimized Quantum Implementation of ARIA Yu-Jin Yang, Kyung-bae Jang, Yu-jin Oh and Hwa-Jeong Seo |
|  | Finding Shortest Vector using Quantum NV Sieve on Grover Hyunji Kim, Kyoungbae Jang, Yujii Oh, Woojin Seok, Wonhuck Lee, Kwangil Bae, Ilkwon Sohn and Hwajeong Seo |
| $\begin{aligned} & \hline \text { KST 11:00-11:10 } \\ & \text { UTC 02:00-02:10 } \end{aligned}$ | Break Time |
| $\begin{aligned} & \text { KST 11:10-12:10 } \\ & \text { UTC 02:10-03:10 } \end{aligned}$ | Session 2: Side Channel Attack I <br> (Session Chair : Dr. Byoungjin Seok (Seoul National University of Science and Technology)) |
|  | Extended Attacks on ECDSA with Noisy Multiple Bit Nonce Leakages Shunsuke Osaki and Noboru Kunihiro |
|  | Single Trace Analysis of Comparison Operation based Constant-Time CDT Sampling and Its Countermeasure Keonhee Choi, Ju-Hwan Kim, Jaeseung Han, Jae-Won Huh and Dong-Guk Han |
|  | A Lattice Attack on CRYSTALS-Kyber with Correlation Power Analysis Yen-Ting Kuo and Atsushi Takayasu |
| $\begin{aligned} & \hline \text { KST 12:10-13:30 } \\ & \text { UTC 03:10-04:30 } \\ & \hline \end{aligned}$ | Break Time (Lunch Time in Korea) |
| KST 13:30-14:30UTC 04:30-05:30 | Session 3 : Cyber Security I <br> (Session Chair: Prof. Seung-Hyun Seo (Hanyang University)) |
|  | A Comparative Analysis of Rust-Based SGX Frameworks: Implications for building SGX applications Heekyung Shin, Jiwon Ock, Hyeon No and Seongmin Kim |
|  | BTFuzzer: a profile-based fuzzing framework for Bluetooth protocols Min Jang, Yuna Hwang, Yonghwi Kwon and Hyoungshick Kim |
|  | mdTLS: How to make middlebox-aware TLS more efficient? Taehyun Ahn, Jiwon Kwak and Seungjoo Kim |
| $\begin{aligned} & \hline \text { KST 14:30-14:40 } \\ & \text { UTC 05:30-05:40 } \\ & \hline \end{aligned}$ | Break Time |
| $\begin{aligned} & \text { KST 14:40-15:40 } \\ & \text { UTC 05:40-06:40 } \end{aligned}$ | Session 4 : Cyber Security II \& Side Channel Attack II (Session Chair: Prof. Kwangsu Lee (Sejong University)) |
|  | PHI: Pseudo-HAL Identification for Scalable Firmware Fuzzing Seyeon Jeong, Eunbi Hwang, Yeongpil Cho and Taekyoung Kwon |
|  | Lightweight Anomaly Detection Mechanism based on Machine Learning Using Low-Cost Surveillance Cameras Yeon-Ji Lee, Na-Eun Park and II-Gu Lee |
|  | Side-Channel Analysis on Lattice-Based KEM using Multi-feature Recognition - The Case Study of Kyber Yuan Ma, Xinyue Yang, An Wang, Congming Wei, Tianyu Chen and Haotong Xu |
| $\begin{aligned} & \hline \text { KST 15:40-15:50 } \\ & \text { UTC 06:40-06:50 } \\ & \hline \end{aligned}$ | Break Time |
| $\begin{aligned} & \text { KST 15:50-17:10 } \\ & \text { UTC 06:50-08:10 } \end{aligned}$ | Session 5 : Applied Cryptography I <br> (Session Chair: Dr. Dongyoung Roh (National Security Research Institute)) |
|  | Enhancing Prediction Entropy Estimation of RNG for On-the-Fly Test Yuan Ma, Weisong Gu, Tianyu Chen, Na Lv, Dongchi Han and Shijie Jia |
|  | Leakage-Resilient Attribute-based Encryption with Attribute-hiding Yijian Zhang, Yunhao Ling, Jie Chen and Luping Wang |
|  | Constant-Deposit Multiparty Lotteries on Bitcoin for Arbitrary Number of Players and Winners Shun Uchizono, Takeshi Nakai, Yohei Watanabe and Mitsugu Iwamoto |
|  | Single-Shuffle Card-Based Protocols with Six Cards per Gate Tomoki Ono, Kazumasa Shinagawa, Takeshi Nakai, Yohei Watanabe and Mitsugu Iwamoto |

## Conference Program

## Thursday (2023-11-30)

| $\begin{aligned} & \text { KST 10:00-11:00 } \\ & \text { UTC 01:00-02:00 } \end{aligned}$ | [Invited Talk I] <br> (Session Chair : Prof. Hwajeong Seo (Hansung University)) <br> Title: Secure Implementation of Post-Quantum Cryptography: Challenges and Opportunities Prof. Rei Ueno (Tohoku University) |
| :---: | :---: |
| $\begin{aligned} & \text { KST 11:00-11:10 } \\ & \text { UTC 02:00-02:10 } \end{aligned}$ | Break Time |
| $\begin{aligned} & \text { KST 11:10-12:10 } \\ & \text { UTC 02:10-03:10 } \end{aligned}$ | Session 6: Signature Schemes (Session Chair : Prof. Mun-Kyu Lee (Inha University)) |
|  | 1-out-of-n Oblivious Signatures: Security Revisited and a Generic Construction with an Efficient Communication Cost Masayuki Tezuka and Keisuke Tanaka |
|  | Compact Identity-based Signature and Puncturable Signature from SQISign Surbhi Shaw and Ratna Dutta |
|  | High Weight Code-based Signature Scheme from QC-LDPC Codes Chik How Tan and Theo Fanuela Prabowo |
| KST 12:10-13:30 <br> UTC 03:10-04:30 | Break Time (Lunch Time in Korea) |
| $\begin{aligned} & \text { KST 13:30-14:30 } \\ & \text { UTC 04:30-05:30 } \end{aligned}$ | [Invited Talk II] <br> (Session Chair : Prof. Joonwoo Lee (Chungang University)) <br> Title: CryptAttackTester: Formalizing Attack A nalyses Dr. Tung Chou (Academia Sinica) |
| $\begin{aligned} & \text { KST 14:30-14:40 } \\ & \text { UTC 05:30-05:40 } \end{aligned}$ | Break Time |
| $\begin{aligned} & \text { KST 14:40-16:00 } \\ & \text { UTC 05:40-07:00 } \end{aligned}$ | Session 7 : Applied Cryptography II \& Quantum Cryptanalysis II (Session Chair : Prof. Yongwoo Lee (Inha University)) |
|  | Efficient Result-Hiding Searchable Encryption with Forward and Backward Privacy Takumi Amada, Mitsugu Iwamoto and Yohei Watanabe |
|  | Finsler Encryption Tetsuya Nagano and Hiroaki Anada |
|  | Experiments and Resource Analysis of Shor's Factorization Using a Quantum Simulator Junpei Yamaguchi, Masafumi Yamazaki, Akihiro Tabuchi, Takumi Honda, Tetsuya Izu and Noboru Kunihiro |
|  | Quantum Circuits for High-Degree and Half Multiplication For Post-Quantum Analysis Rini Wisnu Wardhani, Dedy Septono Catur Putranto and Howon Kim |
| $\begin{aligned} & \text { KST 16:00-16:10 } \\ & \text { UTC 07:00-07:10 } \end{aligned}$ | Break Time |
| $\begin{aligned} & \text { KST 16:10-17:10 } \\ & \text { UTC 07:10-08:10 } \end{aligned}$ | Session 8 : Korean Post Quantum Cryptography (Session Chair : Prof. Seongmin Kim (Sungshin Women's University)) |
|  | Theoretical and Empirical Analysis of FALCON and SOLMAE using their Python Implementation Kwangjo Kim |
|  | Security Evaluation on KpqC Round 1 Lattice-based Algorithms Using Lattice Estimator Suhri Kim, Eunmin Lee, Joohee Lee, Minju Lee and Hyun A Noh |
|  | On the security of REDOG Tanja Lange, Alex Pellegrini and Alberto Ravagnani |
| KST 17:10-18:00 UTC 08:10-09:00 | Break Time |
| $\begin{aligned} & \text { KST 18:00-20:30 } \\ & \text { UTC 09:00-11:30 } \end{aligned}$ | Banquet (Hotel Koreana Diamond Hall 2F) |

## Conference Program

## Friday (2023-12-01)

| $\begin{aligned} & \text { KST 10:00-11:00 } \\ & \text { UTC 01:00-02:00 } \end{aligned}$ | [Invited Talk III] <br> (Session Chair : Prof. Hwajeong Seo (Hansung University)) <br> Title: Hash Based Signatures and Ascon-Sign <br> Dr. Anubhab Baksi (Nanyang Technological University) |
| :---: | :---: |
| $\begin{aligned} & \text { KST 11:00-11:10 } \\ & \text { UTC 02:00-02:10 } \end{aligned}$ | Break Time |
| $\begin{aligned} & \text { KST 11:10-12:30 } \\ & \text { UTC 02:10-03:30 } \end{aligned}$ | Session 9 : Cryptanalysis \& Applied Cryptography III (Session Chair : Dr. Taehwan Park (National Security Research Institute)) |
|  | Distinguisher and Related-Key Attack on HALFLOOP-96 Jinpeng Liu and Ling Sun |
|  | Not optimal but efficient: a distinguisher based on the Kruskal-Wallis test Yan Yan, Elisabeth Oswald and Arnab Roy |
|  | Feasibility Analysis and Performance Optimization of the Conflict Test Algorithms for Searching Eviction Sets Zhenzhen Li, Xue Zihan and Wei Song |
|  | Revisiting Key Switching Techniques with Applications to Light-Key FHE Ruida Wang, Zhihao Li, Benqiang Wei, Chunling Chen, Xianhui Lu and Kunpeng Wang |
|  | Farewell |

The $26{ }^{\text {th }}$ Annual International Conference on Information Security and Cryptology


## Conference Information



## Conference Location

ICISC 2023 Conference is held at Koreana Hotel(2th floor).

## Lunch

Free Lunch for Day 1, Day 2 are provided at the Diamond hall, Koreana hotel (2th floor), in the form of a Lunch box.

## Banquet

In the evening of day 2 (Nov. 30) free banquet is scheduled. It will be held at the Diamond hall, Koreana hotel (2th floor) from 18:00.

## Internet Access

Free Wi-Fi access is provided during the conference.

## 블록체인을 실현하기 위한 가장 빠른 방법


(주)스마트엠투엠에서 출발한 ACCIO 는
누구나 블록체인을 손쉽게 구축할 수 있는 BaaS 솔루션 입니다.

## 당신의 비즈니스에만 집중하세요. 블록체인 인프라에서 자유로워지세요.

AWS/GCP/NHN 등 다양한 클라우드 연계 지원 Hyperledger Fabric, Ethereun, Quorum, Polygon 등 다양한 블록체인 프로토콜 지원

https://accio.technology

## IT-OT 융합보안 솔루션

## SFConSS

IT-OT Converged Security Threat Response System

## 스마트공장 IT-OT 융합보안 위협대응 슬투전

## SFConSS

## 스마트공장 IT-OT 융합보안 위협대응 슬루션 <br> 

## 스마트공장의 가용성을 보장하는 IT-OT 융합보안 위협대응 솔루션 SFConSS

SFConSS는 스마트 공장의 보안위협을 지능적으로 탐지하고 대응하는 IT-OT 융합보안 위협대응 솔루션으로 IT와 OT 환경을 아우르는 폭넓은 가시성을 통해 보안위협의 해석 용의성과 보안 운영관리 편의성을 제공합니다.

IT 프로토콜은 물론 OT망을 구성하는 다양한 프로토콜을 수집•분석한 결과를 기반으로 보안 취약점 분석 및 탐지가 가능 합니다.

위협 인텔리전스 기반 기술로 고성능 이상징후 예측을 통해 크리티컬한 사이버 공격을 사전에 탐지하고 이에 대응하여 보안위협을 최소화 할 수 있습니다.

## 특장점



표준 프로토콜 변환 및 데이터 분산처리


IT프로토콜 및 OT프로토콜 수용


IT-OT 이상(징후) 탐지 및 위협 예측


IT-OT 융합보안 지능형 위협 분석 및 대응


다중 스마트공장 관리

## 주요기능



스마트공장 IT-OT 보안위협 가시화


IT-OT 이상(징후) 탐지 및 예측


IT-OT 융합보안 위협 인텔리전스 분석


스마트공장 IT-OT 계층별 자산 위협 분석


보안정책 설정
(주)유엠로직스

DX 융합보안 전문기업 쿤텍은 파라오의 무덤을 지킨 것으로 알려진 고대 고양이치립,
진화하는 보안 위협에 대한 빠르고 전문적인 대응방안을 제시하여
다양한 환경의 고객 자산을 보호합니다.
공급망보안 임베디드보안 OT보안

# 공공기관 시장 점유율 1위 문서중앙화 솔루션 시큐어디스크 




## 문서 유출 차단

주요 문서 프로그램을 등록 하여 로컬 저장 원천 차단

##  <br> 문서 유실 방지

랜섬웨어 감염, 사용자 실수 또는 변경/삭제된 파일 복원


협업 / 공유
필요에 따라 특정 그룹의
프로젝트 디스크 생성하여 협업


## 데이터 통합 관리

로컬에 저장된 관리 확장자 파일 중앙화 서버로 강제 이관

* 조달청 디지털서비스몰 2016년 이후 누적 판매금액 기준

The $26^{\text {th }}$ Annual International Conference on Information Security and Cryptology ICISC 2023

Invited Talks

INVITED TALK 1

INVITED TALK 1 : Prof. Rei Ueno (Tohoku University)

## Title : Secure Implementation of Post-Quantum Cryptography: Challenges and Opportunities

## Biography

Rei Ueno received the B.E. degree in information engineering and the M.S. and Ph.D. degrees in information sciences from Tohoku University, Japan, in 2013, 2015, and 2018, respectively. He is an Assistant Professor at the Research Institute of Electrical Communication, Tohoku University, and had been joined the JST as a researcher for a PRESTO project for 2018-2022. His research interests include arithmetic circuits, cryptographic implementations, formal verification, and hardware security. Dr. Ueno received the Kenneth C. Smith Early Career Award in Microelectronics at ISMVL 2017.


#### Abstract

Post-quantum cryptography (PQC), which is public key cryptography based on quantum-resistant mathematical problems, is emerging as the recent development of quantum computers. Many studies have been devoted to the design and security analysis of PQC schemes, while their efficient and secure implementation are also very active research topics. Recently, side-channel attacks on re-encryption, which is employed by most postquantum CCA-secure key encapsulation mechanisms (KEMs), have attracted much attention due to its generality and practicality. This talk introduces attacks and defenses on post-quantum cryptographic implementations, with a focus on re-encryption.


## INVITED TALK 2



INVITED TALK 2 : Dr. Tung Chau (Academia Sinica)

Title : CryptAttackTester: formalizing attack analyses

## Biography

Tung Chou is an assistant research fellow at Academia Sinica, Taiwan. He received his Ph.D. degree from Eindhoven University of Technology. Many of his works were about fast software implementations for post-quantum cryptosystems. His recent works are mainly about cryptanalysis and novel ways to reduce signature sizes.
Tung Chou is one of the designers of Classic McEliece, a post-quantum key encapsulation mechanism. Classic McEliece is currently considered by NIST for standardization. He is the main implementer for the 4 official software implementations. He is also the designer of the 5 ' $f$ ' parameter sets, which allow faster key generation. Tung Chou is also one of the designers of MEDS, a post-quantum signature scheme. MEDS is a candidate for NIST's recent call for additional signatures.


#### Abstract

Quantitative analyses of the costs of cryptographic attack algorithms play a central role in comparing cryptosystems, guiding the search for improved attacks, and deciding which cryptosystems to standardize. Unfortunately, these analyses often turn out to be wrong. This talk presents a case study demonstrating the feasibility and value of successfully formalizing what state-of-the-art attack analyses actually do. The formalization process enforces clear definitions, systematically accounts for all algorithm steps, simplifies review, improves reproducibility, and reduces the risk of error. Concretely, our CryptAttackTester (CAT) software includes formal specifications of (1) a general-purpose model of computation and cost metric, (2) various attack algorithms, and (3) formulas predicting the cost and success probability of each algorithm. The software includes generalpurpose simulators that systematically compare the predictions to the observed attack behavior in the same model.




INVITED TALK 3 : Dr. Anubhab Baksi (Nanyang Technological University)
Title: Hash Based Signatures and Ascon-Sign

## Biography

Anubhab Baksi did PhD from Nanyang Technological University, Singapore in 2021. Before that, he finished BSc (Statistics) and BTech (Computer Science \& Engineering). Currently he is employed as a Post-Doctoral researcher. His research interest lies in various aspects of cryptography/cyber security and quantum computing.


#### Abstract

Digital signatures are among the most commonly used cryptographic tool. However, it is believed that the security of existing state-of-the-art signatures would face a serious challenge against an attacker equipped with a functional quantum computer. To overcome this issue, a relatively new direction of research, which aims at designing signatures secured against the quantum attacks, is currently going on in full swing. One such candidate, called the hash based signatures, is based on the cryptographic hash functions. In this talk, we will go through the basic construction of the hash based signatures. We shall also briefly talk about ciphers like SPHINCS+, SPHINCS-alpha and specially Ascon-Sign.


The $26^{\text {th }}$ Annual International Conference on Information Security and Cryptology ICISC 2023

Papers

# Enhancing the Related-Key Security of PIPO through New Key Schedules* 

Seungjun Baek ${ }^{1}$, Giyoon Kim ${ }^{1}$, Yongjin Jeon ${ }^{1}$, and Jongsung Kim ${ }^{1,2}$<br>${ }^{1}$ Department of Financial Information Security, Kookmin University, Republic of Korea<br>\{hellosj3,gi0412,idealtop18,jskim\}@kookmin.ac.kr<br>${ }^{2}$ Department of Information Security, Cryptology, and Mathematics, Kookmin University, Republic of Korea


#### Abstract

In this paper, we present new key schedules for the PIPO block cipher that enhance its security in the related-key setting. While PIPO has demonstrated noteworthy resistance against attacks in the single-key setting, its security in the related-key setting is very vulnerable owing to its simple key schedule. Given the lightweight property of PIPO, we tweak the key schedule algorithm of PIPO by applying computation only within a single register or from one register to another in key states. By adopting our new key schedules, the tweaked version of PIPO achieves better resistance to related-key attacks and demonstrates competitive implementation results in an 8-bit AVR environment. We expect that this paper will contribute to a better understanding of the PIPO block cipher.


Keywords: symmetric-key cryptography • PIPO • block cipher • relatedkey attacks • key schedule

## 1 Introduction

Plug-In Plug-Out (PIPO) [11], proposed at ICISC 2020, is a lightweight block cipher with a substitution permutation network (SPN) structure that supports 64 -bit block size and 128 - and 256 -bit keys. PIPO was designed to be suitable for the AVR embedded processor, which is a typical 8-bit microcontroller. PIPO128 achieved the highest speed in an 8-bit AVR environment among lightweight block ciphers such as SIMON [3], CRAFT [5], PRIDE [1], and RECTANGLE [19]. PIPO is also a block cipher standard that was approved by the Telecommunications Technology Association (TTA) of Korea in 2022 [16]. Since PIPO was developed, its security has been scrutinized by several cryptographers, and its full-round security has not yet been broken in the single-key setting.

[^0]In designing a lightweight block cipher, related-key attacks are often dismissed because, from a practical perspective, they are unlikely to occur. Nevertheless, a block cipher vulnerable to a related-key attack presents some security concerns. It may not be suitable for other cryptographic primitives that use block ciphers as building blocks, e.g., block cipher-based hash functions. A concrete real-world example is the use of a hash function based on the block cipher TEA [10]. Microsoft's Xbox architecture employed a Davies-Meyer hash function instantiated with TEA, and a security vulnerability related to related-key characteristics of TEA was exploited in a hacking [18]. Another security concern arises when secret keys are frequently updated in protocols or when differences can be incorporated through fault attacks.

Recently, several analyses [17,14] of related-key characteristics for PIPO have been proposed. In these analyses, researchers have reported PIPO's full-round characteristics based on iterative characteristics with a high probability. This weakness is attributed to PIPO's simple key schedule. Given that cryptographers repeatedly analyze the related-key security of PIPO, enhancing its resistance against related-key attacks might give them confusions. Furthermore, considering that PIPO is designed for embedded processors, it could also be employed to construct a hash function, which motivates us to scrutinize its security in the context of related-key setting.

Our Contributions. In this paper, we propose tweaks to the key schedule algorithm of PIPO-128. We take into account two conditions for tweaking PIPO128's key schedule. First, our proposed tweaks must ensure better related-key security than the original PIPO-128. This is achieved by rotating the registers of key states in the key schedule algorithm to break the 2-round iterative relatedkey differential characteristics that occur. We also add additional bit-rotation within a register to further improve security. Second, we strive to ensure that the tweaked PIPO algorithm has minimal overhead in an 8-bit AVR environment. To inherit the lightweight nature of PIPO-128 while keeping implementation cost low, we completely exclude nonlinear operators, such as AND or OR gates, in the proposed tweaks. Instead, we mainly apply computation within a single register or from one register to another.

We evaluate the related-key security of our tweaks in terms of the number of active S-boxes in a characteristic. We first construct a Mixed Integer Linear Programming (MILP) model for PIPO-128 and evaluate the number of active S-boxes. Comparing our tweak to the original PIPO-128, we achieve more than twice the number of active S-boxes in characteristics with large rounds. For example, the 10 -round characteristic of the original PIPO-128 had five active S-boxes, while ours has 11 . While this measurement may not yield the characteristic with the lowest probability, it is sufficient to demonstrate the related-key security of PIPO-128. We also examine the implementation efficiency of our tweaks in an 8 -bit AVR environment. Even though our tweaks involve slightly more computation compared to the original PIPO-128, their overhead is minimal. Thus, we
preserve the lightweight property of PIPO-128. Furthermore, we confirm that our tweaks are useful for PIPO-256 as well.

Paper Organization. Section 2 describes the specifications of PIPO and relatedkey differential attacks. Section 3 describes our new tweaks for PIPO's key schedule. Section 4 describes security analysis for the tweaked PIPO in the related-key setting. Section 5 describes our implementation results for the tweaked PIPO in an 8 -bit AVR environment. Section 6 presents our conclusion.

## 2 Preliminaries

### 2.1 Description of PIPO

Figure 1 depicts the process of PIPO [11,12]. The internal state of PIPO is represented by an $8 \times 8$ bit matrix. In the bit matrix, the least significant bit (LSB) is located at the top right and is filled from right to left. When one row is filled, the next row is filled again from the right.

The plaintext is XORed with the whitening key and then undergoes a sequence of $r$ rounds. For PIPO-128, $r$ is 13 , while for PIPO-256, $r$ is 17. Each round consists of three layers: S-layer, R-layer, and round key and constant XOR additions.


Fig. 1. Description of PIPO

S-Layer (SL). PIPO uses the defined 8-bit S-box as shown in Table 1. Each column in the state is independently substituted using eight S-boxes. The top bit of the state becomes the LSB of the S-box input value.

Table 1. PIPO S-box

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | , |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 0 | 5 E | F9 | FC | 00 | 3F | 85 | BA | 5B | 18 | 37 | B2 | C6 | 71 | C3 | 74 |  |
| 1 | A7 | 94 | OD | E1 | CA | 68 | 53 | 2E | 49 | 62 | EB | 97 | A4 | OE | D |  |
| 2 | 16 | 25 | AC | 48 | 63 | D1 | EA | 8F | F7 | 40 | 4 | B1 | 9E | 34 | 1B |  |
| 3 | B9 | 86 | 03 | 7F | D8 | 7A | DD | 3 C | E0 | CB | 52 |  | 15 | AF | 8 C |  |
| 4 | C | 75 | 70 | 1C | 33 | 99 | B6 | C7 | 04 | 3B | BE | 5A | FD | 5 F | 8 |  |
| 5 | 93 | A0 | 29 | 4D | 66 | D4 | EF | OA | E5 | CE | 57 | A3 | 90 | 2A | 09 |  |
| 6 | 22 | 11 | 88 | E4 | CF | 6D | 56 | AB |  | DC | D |  | 82 | 38 | 07 |  |
| 7. | B | 9A | 1F | F3 | 44 | F6 | 41 | 30 | 4 C | 67 | EE | 12 | 21 | 8B |  |  |
| 8 | 55 | 6 E | E7 | OB | 28 | 92 | A1 | CC | 2B | 08 | 91 | ED | D6 | 64 | 4 F |  |
| - | BC | 83 | 06 | FA | 5D | FF | 58 | 39 | 72 | C5 | C0 | B4 | 9B | 31 |  |  |
| A- | 01 | 3 E | BB | DF | 78 | DA | 7D | 84 | 50 | 6B | E2 | 8 E | AD | 17 |  |  |
| B- | AE | 8D | 14 | E8 | D3 | 61 | 4A | 27 | 47 | F0 | F | 19 | 36 | 9C |  | 42 |
| C_ | 1D | 32 | B7 | 43 | F4 | 46 | F1 | 98 | EC | D7 | 4 E | AA | 8 | 23 |  | 65 |
| D- | 8A | A9 | 20 | 54 | 6F | CD | E6 | 13 | DB | 7C | 79 | 05 | 3A | 80 | Br | DE |
| E_ | E9 | D2 | 4B | 2 F | OC | A6 | 95 | 60 | OF | 2 C | A5 | 51 | 6 A | C8 | E3 | 96 |
|  | B0 | 9 F | 1A | 6 | C1 | 73 | C4 | 35 | FE | 59 | 5 C | B8 | 87 | 3D | 02 |  |

R-Layer (RL). RL rotates each row of the state to the left. The rotation values from the top row to the bottom row are $0,7,4,3,6,5,1$, and 2 , respectively.

Round Key and Constant XOR Additions. This layer XORs round constants and the round keys to the internal state. We denote the $i$-th round key as $K_{i}$. We also denote the $j$-th row of $K_{i}$ is $k_{j}^{i}$, i.e., $K_{i}=k_{7}^{i}\left\|k_{6}^{i}\right\| \cdots \| k_{0}^{i}$. In PIPO, there is a whitening key, and we treat it as the 0 -th round key $K_{0}$.
$c_{i}$ is the $i$-th round constant, defined as $c_{i}=i$. This definition includes the case of $i=0$ (i.e., $c_{0}=0$ ). Since $c_{i}$ cannot be higher than 19, the constant XOR addition only affects the 0 -th row of the internal state.

Key Schedule. For PIPO-128, the master key $M K$ is split into two 64-bit states and used alternately (see Figure 2). Let $M K=M K_{1} \| M K_{0}$ for 64-bit values $M K_{0}$ and $M K_{1}$. The $i$-th round key $K_{i}$ is defined by $K_{i}=M K_{i}(\bmod 2)$.

For PIPO-256, the master key $M K$ is split into four 64 -bit states and used in sequence. That is, $K_{i}=M K_{i}(\bmod 4)$ where $M K=M K_{3}\left\|M K_{2}\right\| M K_{1} \| M K_{0}$ for 64 -bit values $M K_{0}, M K_{1}, M K_{2}$, and $M K_{3}$.


Fig. 2. Key schedule of PIPO-128

### 2.2 Related-Key Differential Attack

Related-key attack, independently introduced by Biham [6] and Knudsen [13], is a powerful cryptanalytic tool for the analysis of block ciphers. In this attack, the adversary can obtain the encryption of plaintexts under several related keys, where the relationship between the keys is known to (or can be chosen by) the adversary. Kelsey et al. [9] introduced the related-key differential attack. The adversary can ask for the encryption of plaintext pairs with a chosen difference of $\alpha$, using unknown keys that have a difference of $\Delta K$ in a manner that is known or chosen by the adversary. To attack an $n$-bit cipher, the adversary exploits a related-key differential characteristic $\alpha \rightarrow \beta$ for target (sub-)cipher $E$ with a probability $p$ larger than $2^{-n}$, i.e.,

$$
\operatorname{Pr}_{(P, K)}\left[E_{K}(P) \oplus E_{K \oplus \Delta K}(P \oplus \alpha)=\beta\right]=p>2^{-n}
$$

where $P$ represents a plaintext. Here, the adversary's task is to find a related-key characteristic with as high a probability as possible. This attack is based on the key schedule and on the encryption/decryption algorithms, so a cipher with a weak key schedule may be vulnerable to this kind of attack.

## 3 New Key Schedules of PIPO

In this section, we propose new key schedules of PIPO to enhance the security in the related-key setting. We first observe the existing iterative related-key differential characteristics for PIPO-128 and PIPO-256. Here, we omit the constant addition, as it is not relevant to our analysis.

### 3.1 Related-Key differential characteristics of PIPO

Yadav and Kumar [17] showed a 2-round iterative related-key differential characteristic with probability $2^{-4}$ and constructed a full-round characteristic with probability $2^{-24}$ for PIPO-128. Soon after, Sun and Wang [14] reported full-round differential characteristics of PIPO-256 for the first time. Due to the simple key schedule of PIPO, we can construct several related-key differential characteristics containing only a few active S-boxes. Concretely, 2-round iterative related-key differential characteristics can be found straightforwardly (see Figure 3).


Fig. 3. 2-round iterative related-key differential characteristics of PIPO-128

In the transition $X_{r} \xrightarrow{\mathrm{SL}} Y_{r} \xrightarrow{\mathrm{RL}} Z_{r}$, the 2-round characteristic is constructed by setting $\Delta X_{r}$ and $\Delta Z_{r}$ as iterative keys. Considering the differential distribution table (DDT) of PIPO, there are 224 entries with probability $2^{-4}$ (see Table 2). Since the difference of $X_{r}$ can also be placed in the remaining seven columns, there are a total of $224 \times 8=1792$ characteristics.

Table 2. Distribution of non-trivial probabilities in DDT of PIPO's S-box

| DDT value | 2 | 4 | 6 | 8 | 10 | 12 | 16 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| \# of entries | 12552 | 6226 | 651 | 951 | 9 | 7 | 224 |
| probability | $2^{-7}$ | $2^{-6}$ | $2^{-5.415}$ | $2^{-5}$ | $2^{-4.678}$ | $2^{-4.415}$ | $2^{-4}$ |



Fig. 4. 4-round iterative related-key differential characteristics of PIPO-256

Similarly, there exists a full-round differential characteristic with probability $2^{-16}$ for PIPO-256 based on the 4-round iterative differential characteristic (see Figure 4).

### 3.2 Introducing New Key Schedules: KS1 and KS2

We propose new key schedules for PIPO-128. There are two factors to consider in order to simultaneously satisfy the related-key security and implementation efficiency of our key schedules. Note that we are not considering changing the entire algorithm of PIPO-128; we are only changing the key schedule. Our considerations for these tweaks are as follows:

1. Increasing resistance against related-key differential attacks - In PIPO-128's key schedule, the master key is divided into two 64 -bit key states, and the attacker determines the difference of the two states by selecting the difference of the master key. Given this simple key schedule, the initially selected difference remains fixed within the two key states and is XORed throughout the entire algorithm every two rounds. Ultimately, there are 2round iterative differential characteristics resulting from this property, so our main goal is to prevent such characteristics from occurring. To increase resistance against related-key differential attacks, we induce diffusion within the key schedule in the column-wise as well as row-wise directions. Specifically, we measure the minimum number of active S-boxes using the MILP tool. This number enables us to establish the bounds on the probability of differential characteristics, considering the best probability of $2^{-4}$ from the PIPO S-box's DDT table.
2. Achieving minimal overhead - When considering tweaks to the key schedule, one might choose to apply various operators to induce diffusion of differences within key states. Recall that PIPO-128 is a block cipher optimized for 8 -bit microcontrollers, so it primarily relies on computations in terms of register level throughout the encrypting/decrypting process. To inherit this advantage, we strictly limit our key schedule tweaks to computing within a single register or from one register to another. Specifically, to preserve the low implementation cost, we completely exclude nonlinear operators such as AND or OR gates. Finally, we tweak the key schedule in a way that ensures security while minimizing the overhead in an 8-bit AVR environment.

Now we introduce two new key schedules of PIPO-128, which we refer to as KS1 and KS2. We refer to the original key schedule of PIPO-128 as KSO. The evaluation of their related-key security is discussed in Section 4.

KS1. KS1 is our first proposal for PIPO-128's key schedule. Our aim is to eliminate the 2-round iterative characteristics of PIPO-128 (see Figure 3) in KS1. To do this, we simply rotate each row register within each of the two key states by 1 in the upward direction. For the first two rounds, two key states are input as $M K_{0}$ and $M K_{1}$, but from then on, rotation is applied to each register every two rounds. If we apply this operation to the key schedule, a 2 -round iterative pattern is easily broken due to the RL of PIPO-128. In Figure 5, we describe one example demonstrating our claim. We distinguish the differences between two key states: one represented by the color orange and the other by the color blue. In addition, we use hatch patterns to denote all possible differences. Here, we can see that the difference cancellation does not occur in the transition $X_{2} \xrightarrow{\mathrm{SL}} Y_{2} \xrightarrow{\mathrm{RL}} Z_{2} \rightarrow X_{3}$. This is because the possible differences caused by $\Delta X_{2}$ are not canceled out, mainly due to the rotation of the key state. In this way, difference cancellation patterns are prevented by applying rotations of registers. Therefore, KS1 can amplify the diffusion of key differences more than the original


Fig. 5. Breaking the occurrence of the 2-round iterative characteristic of PIPO-128
one. KS1 is represented as follows:

$$
\begin{aligned}
& \left(K_{r-1}, K_{r}\right)=\left(k_{7}^{r-1}\left\|k_{6}^{r-1}\right\| \cdots\left\|k_{0}^{r-1}, k_{7}^{r}\right\| k_{6}^{r}\|\cdots\| k_{0}^{r}\right) \\
& \quad \xrightarrow{2 \text { rounds }}\left(K_{r+1}, K_{r+2}\right)=\left(k_{0}^{r+1}\left\|k_{7}^{r+1}\right\| \cdots\left\|k_{1}^{r+1}, k_{0}^{r+2}\right\| k_{7}^{r+2}\|\cdots\| k_{1}^{r+2}\right) .
\end{aligned}
$$

In the rotation of two key states, one may consider rotating or changing only a few registers in each state. Let the attacker choose a key difference in the unchanged registers in one key state, typically one bit, and then set the difference determined by the RL operation in the unchanged registers in the other state. Since the differences in the unchanged registers are fixed, a 2 -round characteristic occurs repeatedly. This is not a desirable proprety for us, so we do not adopt this method.

KS2. While KS1 offers better related-key security compared to KSO, there is still room for further improvement in security. The focus of KS2 is to improve the related-key security of KS1 by applying bit-rotation to one register in each key state. In KS1, there is no row-wise directional diffusion of the key difference,
allowing us to consider bit-rotation within the registers. Since our goal is to minimize overhead in an 8-bit AVR environment, we consider the optimized 8bit rotation operations presented in [11] based on [7]. We mainly consider 1-bit and 4 -bit left rotations, which require 2 and 1 clock cycles, respectively (see Table 3). The remaining bit-rotation operations require 3 to 5 clock cycles, so we do not take into account other cases. We also apply bit-rotation to only one upper register to keep the implementation efficient. Surprisingly, according to our examinations, applying 4 -bit rotation yields better results than 1-bit rotation, even though 4 -bit rotation is less expensive. Thus, we adopt the 4 -bit rotation for KS2. KS2 is represented as follows:

$$
\begin{aligned}
& \left(K_{r-1}, K_{r}\right)=\left(k_{7}^{r-1}\left\|k_{6}^{r-1}\right\| \cdots\left\|k_{0}^{r-1}, k_{7}^{r}\right\| k_{6}^{r}\|\cdots\| k_{0}^{r}\right) \\
& \xrightarrow{2 \text { rounds }}\left(K_{r+1}, K_{r+2}\right)=\left(k_{0}^{r+1}\left\|k_{7}^{r+1}\right\| \cdots\left\|\left(k_{1}^{r+1} \lll 4\right), k_{0}^{r+2}\right\| k_{7}^{r+2}\|\cdots\|\left(k_{1}^{r+2} \lll 4\right)\right) .
\end{aligned}
$$

Table 3. 8-bit rotations on 8-bit AVR

| <<1 | <<2 | < 3 | <<4 | < 5 | < 6 | < 7 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\left\lvert\, \begin{array}{ll} \text { LSL X1 } \\ \text { ADC X1, } & \\ \text { ZERO } \end{array}\right.$ | LSL X1 <br> ADC X1, ZERO <br> LSL X1 <br> ADC X1, ZERO | $\left\lvert\, \begin{array}{lll} \text { SWAP X1 } \\ \text { BST X1, } & 0 \\ \text { LSR } & \text { X1 } & \\ \text { BLD X1, } & 7 \end{array}\right.$ | SWAP X1 | $\begin{aligned} & \text { SWAP X1 } \\ & \text { LSL X1 } \\ & \text { ADC X1, ZERO } \end{aligned}$ | SWAP X1 LSL X1 ADC X1, ZERO LSL X1 ADC X1, ZERO | $\begin{aligned} & \text { BST X1, } 0 \\ & \text { LSR X1 } \\ & \text { BLD X1, } 7 \end{aligned}$ |
| 2 cycles | 4 cycles | 4 cycles | 1 cycle | 3 cycles | 5 cycles | 3 cycles |

## 4 MILP-based Search for Related-Key Characteristics for PIPO with New Key Schedules

Now, we present a security analysis for new key schedules for PIPO-128. We adopt the MILP framework for bit-oriented ciphers proposed by Sun et al. [15] and describe the MILP model for PIPO-128. To optimize the model, we use the Gurobi MILP solver. We utilized the MILES tool [17] for generating linear inequalities of the PIPO-128 S-box. Finally, we apply our MILP model to search for related-key characteristics for PIPO-128 with new key schedules and present the results. We also present some results for PIPO-256.

### 4.1 MILP model for PIPO

Generating Linear Inequalities of S-box. Yadav and Kumar [17] proposed the MILES tool to minimize the number of linear inequalities for large S-boxes. Minimizing the number of inequalities directly affects the efficiency of MILP model. Thus, we utilize the MILES tool to generate linear inequalities of the PIPO S-box. As described in [17], we obtain 4474 linear inequalities for the Sbox, and 35792 inequalities are needed for the SL of one round of PIPO.

Variables and Constraints. We represent the difference of all cells in each round as a set of binary variables $x_{i}$. Each variable $x_{i}$ can take on values 0 or 1 , signifying inactive and active bits, respectively. The binary variables $x_{0}, x_{1}, \cdots, x_{63}$ represent a 64 -bit plaintext difference, and the difference for the next round state is updated as $x_{64}, x_{65}, \cdots, x_{127}$, and so on. To reduce the number of variables in the MILP model, the output bits of SL in the first round are set to the variables in the next round with $\mathrm{RL}^{-1}$ applied, and the process is repeated for each subsequent round. This process in the first round is represented as follows:

Here, we construct linear inequalities based on the input and output variables of SL. Since SL is applied column-wise, the linear inequalities are also constructed in such a manner.

To search a related-key differential characteristics, we represent the difference of the master key as a set of binary variables $k_{i}$. For PIPO-128, its 128 -bit key is represented by $k_{0}, k_{1}, \cdots, k_{127}$ and for PIPO-256, the 256-bit key is represented by $k_{0}, k_{1}, \cdots, k_{255}$.

In our model, the XOR operation of the difference is used for XORing the internal state and the key to generate a new internal state. $x_{i n}$ and $k_{i n}$ are the state bit and corresponding key bit, respectively, and $x_{\text {out }}$ is the corresponding output bit. The following inequalities are used to describe the XOR operation:

$$
\left\{\begin{array}{l}
x_{\text {in }}+k_{\text {in }}-x_{\text {out }} \geq 0 \\
x_{\text {in }}-k_{\text {in }}+x_{\text {out }} \geq 0 \\
-x_{\text {in }}+k_{\text {in }}+x_{o u t} \geq 0 \\
x_{\text {in }}+k_{\text {in }}+x_{\text {out }} \leq 2
\end{array}\right.
$$

In addition, we use the following set of the inequalities to check the number of active S-boxes of a characteristic:
$\left\{\begin{array}{l}x_{64 \cdot(r-1)+i}+x_{64 \cdot(r-1)+i+8}+x_{64 \cdot(r-1)+i+16}+x_{64 \cdot(r-1)+i+24} \\ +x_{64 \cdot(r-1)+i+32}+x_{64 \cdot(r-1)+i+40}+x_{64 \cdot(r-1)+i+48}+x_{64 \cdot(r-1)+i+56}-a_{(r, i)} \geq 0, \\ a_{(r, i)}-x_{64 \cdot(r-1)+i} \geq 0, \\ a_{(r, i)}-x_{64 \cdot(r-1)+i+8} \geq 0, \\ a_{(r, i)}-x_{64 \cdot(r-1)+i+16} \geq 0, \\ a_{(r, i)}-x_{64 \cdot(r-1)+i+24} \geq 0, \\ a_{(r, i)}-x_{64 \cdot(r-1)+i+32} \geq 0, \\ a_{(r, i)}-x_{64 \cdot(r-1)+i+40} \geq 0, \\ a_{(r, i)}-x_{64 \cdot(r-1)+i+48} \geq 0, \\ a_{(r, i)}-x_{64 \cdot(r-1)+i+56} \geq 0,\end{array}\right.$
where $a_{(r, i)}$ denotes whether the $i$-th column from the right is active.

Objective Function. Our goal is to minimize the number of active S-boxes of a characteristic. Thus, when finding a r-round characteristic, our objective function is

$$
\text { Minimize } \sum_{\text {Round 1 }} a_{(1, i)}+\sum_{\text {Round 2 }} a_{(2, i)}+\cdots+\sum_{\text {Round } r} a_{(r, i)} .
$$

### 4.2 Results

We apply our MILP model to PIPO-128 with KSO, KS1, and KS2. Due to the large search space, we only compare these results up to 10 rounds of PIPO-128. In the case of KS2 in round 10, we were unable to prove that this is the best result since the MILP solver did not terminate within a reasonable amount of time. We imposed a one-month time constraint for this case and ran the solver. Our results are summarized in Table 4.

We can observe that in rounds 1 to 2 , the results for three key schedules are identical since the first two key states are the same as $K_{0}$ and $K_{1}$. The change occurs starting from round 5 , which is due to the differential diffusion resulting from additional operations on key states. In particular, in KSO, there are rounds where active S-boxes do not exist every two rounds, whereas, in KS1 and KS2, after round 3 , there is at least one active S-box in each round. In comparing KSO and KS1, the difference in the number of active S-boxes begins to appear from round 9 , and considering the results up to round 10 , this difference is expected to increase as the number of rounds increases. This difference is more pronounced when comparing KSO and KS2. Furthermore, even if the number of active Sboxes in round 10 of KS2 may not be optimal, we need to consider at least three additional active S-boxes to reach a full-round PIPO. Thus, by adopting KS2 as the key schedule for the tweaked version of PIPO, we expect that there will be no related-key differential characteristics with a probability higher than $2^{-64}$.

Table 4. Comparison of related-key differential characteristics for PIPO-128 according to KS0, KS1, and KS2

| Round | KS0 |  | KS1 |  | KS2 |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | $\#($ Active S-box) | $-\log _{2} p$ | $\#$ (Active S-box) | $-\log _{2} p$ | $\#($ Active S-box $)$ | $-\log _{2} p$ |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 1 | 4 | 1 | 4 | 1 | 4 |
| 4 | 2 | 8 | 2 | 8 | 2 | 8 |
| 5 | 2 | 8 | 3 | 13 | 3 | 13 |
| 6 | 3 | 12 | 4 | 22.415 | 4 | 23 |
| 7 | 3 | 12 | 5 | 29 | 5 | 30 |
| 8 | 4 | 16 | 6 | 33.415 | 6 | 36.415 |
| 9 | 4 | 16 | 7 | 38.415 | 8 | 46.415 |
| 10 | 5 | 20 | 8 | 47.415 | $11^{*}$ | 60.830 |

KS0 represents the original key schedule of PIPO-128.
*Number of active S-boxes are not confirmed to be optimal.

On the Results for PIPO-256. We try to apply the approach of the key schedule KS1 to PIPO-256, and we refer to it as KS1*. That is, we simply rotate each row register within each of the four key states by 1 in the upward direction. In the same way as with PIPO-128, four key states are input as $M K_{0}, M K_{1}$, $M K_{2}$, and $M K_{3}$ in the first four rounds. As a result, we see that even when KS1* is adopted for PIPO-256, we can achieve better related-key security than the original key schedule (see Table 5).

We also attempted to apply the approach of KS2 to PIPO-256. However, due to the larger search space, the MILP solver did not termininate after 14 rounds. Furthermore, the results are either the same as or inferior to those obtained using KS1. Thus, we only present the results adopting KS1*.

## 5 Implementations

In this section, we compare our implementation results with the original PIPO128 and other lightweight block ciphers. We used Atmel Studio 6.2 and compiled all implementations with optimization level 3. Our target processor was an ATmega128 running at 8 MHz , as in [11]. Since we could not find a reference assembly code for PIPO-128, we developed the code and analyzed it for a fair comparison. We also adopted a metric to measure overall performance on low-end devices, RANK, which is calculated as

$$
R A N K=\left(10^{6} / C P B\right) /(R O M+2 \times R A M)
$$

where the code size represents ROM. Table 6 compares results for PIPO-128 on 8bit AVR environment according to key schedules. Results for other block ciphers can be found in [11].

Table 5. Comparison of related-key differential characteristics for PIPO-256 according to KSO*, KS1*

| Round | KSO $^{*}$ |  | KS1 $^{*}$ |  |
| :---: | :---: | :---: | :---: | :---: |
|  | $\#$ (Active S-box) | $-\log _{2} p$ | $\#$ (Active S-box) | $-\log _{2} p$ |
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 |
| 5 | 1 | 4 | 1 | 4 |
| 6 | 1 | 4 | 1 | 4 |
| 7 | 1 | 4 | 1 | 4 |
| 8 | 2 | 8 | 2 | 8 |
| 9 | 2 | 8 | 3 | 13 |
| 10 | 2 | 8 | 4 | 20.415 |
| 11 | 2 | 8 | 4 | 22 |
| 12 | 3 | 12 | 6 | 31.415 |
| 13 | 3 | 12 | 8 | 39 |
| 14 | 3 | 12 | 10 | 51.245 |
| 15 | 3 | 12 | 11 | 60.660 |
| 16 | 4 | 16 | 12 | 67 |

*We refer to the original key schedule of PIPO-256 as KSO*.

Table 6. Comparison of PIPO-128 on 8-bit AVR according to key schedules with other lightweight block ciphers

| Block cipher | Code size <br> (bytes) | RAM <br> (bytes) | Execution time <br> (cycles per byte) | RANK |
| ---: | ---: | ---: | ---: | ---: |
| PIPO-64/128(KSO) | 354 | 31 | 197 | 12.09 |
| SIMON-64/128 [3] | 290 | 24 | 253 | 11.69 |
| PIPO-64/128(KS1) | 354 | 31 | 249 | 8.85 |
| PIPO-64/128(KS2) | 354 | 31 | 251 | 8.78 |
| RoadRunneR-64/128 [2] | 196 | 24 | 477 | 8.59 |
| RECTANGLE-64/128 [19] | 466 | 204 | 403 | 2.84 |
| PRIDE-64/128 [1] | 650 | 47 | 969 | 1.39 |
| SKINNY-64/128 [4] | 502 | 187 | 877 | 1.30 |
| PRESENT-64/128 [8] | 660 | 280 | 1,349 | 0.61 |
| CRAFT-64/128 [5] | 894 | 243 | 1,504 | 0.48 |

We also implemented PIPO-256 with KSO* and KS1* in the same environment. Both cases require the same code size and RAM: 354 bytes of code and 47 bytes of RAM. With regard to the execution time, PIPO-256 with KSO* requires 253 CPB, whereas with KS1*, it requires 321 CPB. Therefore, the RANK metrics for them are 8.82 and 6.95 , respectively.

## 6 Conclusion

In this paper, we presented two new key schedules, KS1 and KS2, for the PIPO block cipher, aiming to enhance PIPO's related-key security. By applying KS1 and KS2 to PIPO-128, we achieved better related-key security compared to original PIPO128. We also applied KS1 to PIPO-256 and obtained interesting results regarding security. We obtained comparative implementation results in an 8-bit AVR environment by completely excluding nonlinear operators and only applying computation within a single register or from one register to another. The significance of this study lies in enhancing related-key security of PIPO without significantly increasing the implementation cost.

## References

1. Albrecht, M.R., Driessen, B., Kavun, E.B., Leander, G., Paar, C., Yalçin, T.: Block ciphers - focus on the linear layer (feat. PRIDE). In: CRYPTO 2014. Lecture Notes in Computer Science, vol. 8616, pp. 57-76. Springer (2014), https://doi.org/10. 1007/978-3-662-44371-2_4
2. Baysal, A., Sahin, S.: Roadrunner: A small and fast bitslice block cipher for low cost 8 -bit processors. In: LightSec 2015. Lecture Notes in Computer Science, vol. 9542, pp. 58-76. Springer (2015), https://doi.org/10.1007/978-3-319-29078-2_4
3. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The simon and speck block ciphers on AVR 8-bit microcontrollers. In: LightSec 2014. Lecture Notes in Computer Science, vol. 8898, pp. 3-20. Springer (2014), https://doi.org/10.1007/978-3-319-16363-5_1
4. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: CRYPTO 2016. Lecture Notes in Computer Science, vol. 9815, pp. 123-153. Springer (2016), https://doi.org/10.1007/978-3-662-53008-5_5
5. Beierle, C., Leander, G., Moradi, A., Rasoolzadeh, S.: CRAFT: lightweight tweakable block cipher with efficient protection against DFA attacks. IACR Trans. Symmetric Cryptol. 2019(1), 5-45 (2019), https://doi.org/10.13154/tosc.v2019. i1.5-45
6. Biham, E.: New types of cryptanalytic attacks using related keys. J. Cryptol. 7(4), 229-246 (1994), https://doi.org/10.1007/BF00203965
7. Corporation, A.: Atmega128(1) datasheet (Visited on October 4, 2023), www. microchip.com/wwwproducts/en/ATmega128
8. Engels, S., Kavun, E.B., Paar, C., Yalçin, T., Mihajloska, H.: A non-linear/linear instruction set extension for lightweight ciphers. In: 21st IEEE Symposium on Computer Arithmetic, ARITH 2013, Austin, TX, USA, April 7-10, 2013. pp. 6775. IEEE Computer Society (2013), https://doi.org/10.1109/ARITH.2013.36
9. Kelsey, J., Schneier, B., Wagner, D.A.: Key-schedule cryptanalysis of idea, gdes, gost, safer, and triple-des. In: CRYPTO '96. Lecture Notes in Computer Science, vol. 1109, pp. 237-251. Springer (1996), https://doi.org/10.1007/ 3-540-68697-5_19
10. Kelsey, J., Schneier, B., Wagner, D.A.: Related-key cryptanalysis of 3-way, bihamdes, cast, des-x, newdes, rc2, and TEA. In: ICICS'97. Lecture Notes in Computer Science, vol. 1334, pp. 233-246. Springer (1997), https://doi.org/10.1007/ BFb0028479
11. Kim, H., Jeon, Y., Kim, G., Kim, J., Sim, B., Han, D., Seo, H., Kim, S., Hong, S., Sung, J., Hong, D.: PIPO: A lightweight block cipher with efficient higherorder masking software implementations. In: ICISC 2020. Lecture Notes in Computer Science, vol. 12593, pp. 99-122. Springer (2020), https://doi. org/10.1007/ 978-3-030-68890-5_6
12. Kim, H., Jeon, Y., Kim, G., Kim, J., Sim, B., Han, D., Seo, H., Kim, S., Hong, S., Sung, J., Hong, D.: A new method for designing lightweight s-boxes with high differential and linear branch numbers, and its application. IEEE Access 9, 150592150607 (2021), https://doi.org/10.1109/ACCESS . 2021. 3126008
13. Knudsen, L.R.: Cryptanalysis of LOKI91. In: AUSCRYPT '92. Lecture Notes in Computer Science, vol. 718, pp. 196-208. Springer (1992), https://doi.org/10. 1007/3-540-57220-1_62
14. Sun, L., Wang, M.: Sok: Modeling for large s-boxes oriented to differential probabilities and linear correlations. IACR Transactions on Symmetric Cryptology 2023(1), 111-151 (Mar 2023), https://tosc.iacr.org/index.php/ToSC/ article/view/10310
15. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: Application to simon, present, lblock, DES(L) and other bit-oriented block ciphers. In: ASIACRYPT 2014. Lecture Notes in Computer Science, vol. 8873, pp. 158-178. Springer (2014), https://doi.org/10.1007/978-3-662-45611-8_9
16. TTAK.KO-12.0382: 64-bit Block Cipher PIPO. Telecommunications Technology Association of Korea (2022), https://www.tta.or.kr/tta/ttaSearchView.do? key $=77 \& r e p=1 \&$ searchStandardNo=TTAK.KO-12.0382\&searchCate=TTAS
17. Yadav, T., Kumar, M.: Modeling large s-box in MILP and a (related-key) differential attack on full round PIPO-64/128. In: SPACE 2022. Lecture Notes in Computer Science, vol. 13783, pp. 3-27. Springer (2022), https://doi.org/10. 1007/978-3-031-22829-2_1
18. ZDNet: New xbox security cracked by linux fans (Visited on October 4, 2023), https://www.zdnet.com/article/ new-xbox-security-cracked-by-linux-fans/
19. Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., Verbauwhede, I.: RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. Sci. China Inf. Sci. 58(12), 1-15 (2015), https://doi.org/10.1007/s11432-015-5459-7

# Optimized Quantum Implementation of SEED 

Yujin Oh, Kyungbae Jang, Yujin Yang, and Hwajeong Seo<br>Division of IT Convergence Engineering, Hansung University, Seoul, South Korea<br>oyj0922@gmail.com, starj1023@gmail.com, yujin.yang34@gmail.com, hwajeong84@gmail.com


#### Abstract

With the advancement of quantum computers, it has been demonstrated that Grover's algorithm enables a potential reduction in the complexity of symmetric key cryptographic attacks to the square root. This raises increasing challenges in considering symmetric key cryptography as secure. In order to establish secure post-quantum cryptographic systems, there is a need for quantum post-quantum security evaluations of cryptographic algorithms. Consequently, NIST is estimating the strength of post-quantum security, driving active research in quantum cryptographic analysis for the establishment of secure post-quantum cryptographic systems. In this regard, this paper presents a depth-optimized quantum circuit implementation for SEED, a symmetric key encryption algorithm included in the Korean Cryptographic Module Validation Program (KCMVP). Building upon our implementation, we conduct a thorough assessment of the post-quantum security for SEED. Our implementation for SEED represents the first quantum circuit implementation for this cipher.


Keywords: Quantum Circuit • SEED • Korean Block Cipher • Grover Algorithm.

## 1 Introduction

Quantum computers are the new and upcoming computing paradigm which are based on quantum mechanical principles (such as superposition and entanglement), and will be able to solve certain classes of problems significantly faster than the classical computers. Quantum computers are being developed by many top-tier companies and research institutions.

The introduction of the Shor algorithm [1], which is known for its ability to solve the integer factorization problem and the discrete logarithm problem in polynomial time, poses significant risks to public-key cryptography designed based on these problems. Similarly, the Grover search algorithm [2], known for its ability to reduce the complexity of data search by a square root factor, can have a significant impact on the security of symmetric key cryptography.

NIST has proposed criteria for estimating the quantum attack complexity on the AES family and a parameter called MAXDEPTH, which represents the maximum circuit depth that a quantum computer can execute, in its evaluation
criteria document for post-quantum cryptography standardization $[3,4]$. Both of these aspects need to be considered to evaluate the quantum security strength of a cipher. Detailed explanations on these topics will be provided in Section 2.4, 5.

Based on these NIST criteria, continuous efforts have been made to estimate the complexity of Grover's key search for symmetric-key ciphers and evaluate post-quantum security [ $5,6,7$ ]. In addition to AES, research has also been conducted on estimating quantum resources for well-known lightweight block ciphers such as SPECK, GIFT, and PRESENT [8, 9, 10], as well as lightweight block ciphers selected as finalists in the Lightweight Cryptography (LWC) competition, including SPARKLE [11,12] and ASCON [13, 14].

In this paper, we propose an optimized quantum circuits for SEED, which is a symmetric key encryption algorithms included as validation subjects in the Korean Cryptographic Module Validation Program (KCMVP). Since these cryptographic algorithms are widely used in cryptographic modules in Korea, it is of great importance to estimate quantum resources and measure the quantum security strength of these ciphers. Using the proposed quantum circuit as a basis, we assess the post-quantum security strength of SEED in accordance with NIST criteria.

### 1.1 Our Contribution

The contribution in this paper is manifold and can be summarized as follows:

1. Quantum Circuit Implementation of SEED. We demonstrate the first implementation of a quantum circuit for SEED, which is the one of Korean cipher.
2. Low-Depth Implementation of SEED. In our quantum circuit implementation of SEED, we focus to optimize a low Toffoli depth and full depth. We implement the Itoh-Tsujii algorithm for S-box optimization. For the implementation, we utilize the WISA'22 quantum multiplication, and a squaring based on PLU factorization. Further, we enhance the efficiency of depth optimization by using an optimal quantum adder(which is called CDKM adder) and implementing parallelization for applicable components.
3. Post-quantum Security Assessment of SEED. We estimate the cost of Grover's key search using an our implemented quantum circuit for SEED in order to assess the quantum security of SEED. During this assessment, we compare the estimated cost of Grover's key search for SEED with the security levels defined by NIST.

## 2 Preliminaries

### 2.1 SEED Block Cipher

SEED is a block cipher of Feistel structure operates on 128 -bit block and 128 -bit key. It consists of 16 rounds and each round has a round function $F$.

A 128 -bit block is divided into 64 -bit blocks, and the right 64 -bit block $\left(R_{0}\right)$ serves as the input to the $F$ function with 64 -bit round key. The output of $F$ function is XORed to the left 64-bit block $\left(L_{0}\right)$. The overall structure of SEED cipher is shown in Figure 1.


Fig. 1: Overall structure of SEED cipher.
$\boldsymbol{F}$ Function The input of $F$ function (Figure 2) is 64 -bit block and 64 -bit round key $R K_{i}=\left(K_{i, 0}, K_{i, 1}\right)$. The 64 -bit block is divided into two 32 -bit blocks $(C, D)$ and each block is XORed with the round key. The $F$ function consists of XOR operations $(\oplus)$, modular additions $(\boxplus)$, and $G$ functions.


Fig. 2: Process of the $F$ function.

4 Yujin Oh, Kyungbae Jang, Yujin Yang, and Hwajeong Seo
$G$ Function The 32 -bit input block of the $G$ function (Figure 3) is divided into 8 -bit blocks ( $X_{0-3}$ ) and each block becomes input for the S -boxes.

To compute the output of an S-box, it involves exponentiation of $x \in \mathbb{F}_{2^{8}} /\left(x^{8}+\right.$ $x^{6}+x^{5}+x+1$ ), matrix-vector multiplication, and XORing a single constant. Specifically, two distinct S-boxes ( $S_{1}$ and $S_{2}$ ) are employed, each using its own corresponding set of matrices $\left(A^{(1)}\right.$ or $\left.A^{(2)}\right)$, exponentiation values ( $x^{247}$ or $x^{251}$ ), and constant values ( 169 or 56 ), which are as follows:

$$
\begin{equation*}
S_{1}(x)=A^{(1)} \cdot x^{247} \oplus 169, \quad S_{2}(x)=A^{(2)} \cdot x^{251} \oplus 56 \tag{1}
\end{equation*}
$$

The output values of the S-boxes are ANDed (\&) with the constants $m_{0-3}$, and the results of these AND operations are XORed with each other to compute the final output (i.e., $Z_{0-3}$ ).


Fig. 3: Process of the $G$ function.

Key Schedule In the key schedule (Figure 4), the 128-bit key is divided into four blocks $\left(A\|B\| C \| D\right.$, where $\|$ denotes concatenation), and key constnt values $\left(K C_{i}\right)$ are utilized. Additionally, operations such as shift ( $\gg \lll)$, modular addition, modular subtraction ( $\boxminus$ ), and $G$ function are applied.


Fig. 4: Process of the key schedule

### 2.2 Quantum Gates

This section describes commonly used quantum gates (Figure 5) for implementing quantum circuits of block ciphers (note that this is not an exhaustive list of all possible gates that can be used).

The X gate acts like a NOT operation on a classical computer, reversing the state of the qubit that goes through it. The Swap gate exchanges the states of two qubits. The CNOT gate behaves like an XOR operation on a classical computer. In $\operatorname{CNOT}(a, b)$, the input qubit $a$ is the control qubit, and $b$ is the target qubit. When the control qubit $a$ is in the state 1 , the target qubit $b$ is flipped. As a result, the value of $a \oplus b$ is stored in the qubit $b$ (i.e., $b=a \oplus b$ ), while the state of qubit $a$ remains unchanged. The Toffoli gate, represented as Toffoli $(a, b, c)$, acts like an AND operation on a classical computer. It requires three input qubits, with the first two qubits ( $a$ and $b$ ) serving as control qubits. Only when both control qubits are in the state 1 , the target qubit $c$ is flipped. The result of the operation $a \& b$ is XORed with the qubit $c$ (i.e., $c=c \oplus(a \&$ $b)$ ), while the states of qubits $a$ and $b$ are preserved.

### 2.3 Grover's Key Search

Grover's algorithm searches for a specific data from an unsorted set of $N$ with a search complexity of $O(\sqrt{N})$. In cryptography, for an encryption scheme that uses a $k$-bit key, a classical computer requires a search of $O\left(2^{k}\right)$ complexity for exhaustive key search. However, using Grover's algorithm, a quantum computer can perform this search with a complexity of only $O\left(\sqrt{2^{k}}\right)$, which is reduced by a square root. In this section, we divide the progress of Grover's key search into


Fig. 5: Quantum gates: X (left top), Swap (right top), CNOT (left bottom) and Toffoli (right bottom) gates.
three stages: Input Setting, Oracle, and Diffusion Operator, and describe them as follows.

1. Input Setting: Prepare a $k$-qubit key in a superposition state using Hadamard gates. In this case, equal amplitudes are generated for all $2^{k}$ possible states.

$$
\begin{equation*}
H^{\otimes k}|0\rangle^{\otimes k}=|\psi\rangle=\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)=\frac{1}{2^{k / 2}} \sum_{x=0}^{2^{k}-1}|x\rangle \tag{2}
\end{equation*}
$$

2. In the oracle, the target encryption algorithm(Enc) is implemented through a quantum circuit. This circuit encrypts a known plaintext $(p)$ in a superposition state using a pre-prepared key (as set in the input setting), producing ciphertexts for every possible key value. Subsequently, these generated ciphertexts are compared with the known ciphertexts (performed in $f(x)$ ). Upon discovering a match (i.e., when $f(x)=1$ in Equation (3)), the sign of the desired key state to be recovered is negated (i.e., $(-1)^{f(x)}$ in Equation (4)). Finally, the implemented quantum circuit reverses the generated ciphertexts back to the known plaintext for the next iteration.

$$
\begin{gather*}
f(x)=\left\{\begin{array}{l}
1 \text { if } E n c_{\text {key }}(p)=c \\
0 \text { if } E n c_{\text {key }}(p) \neq c
\end{array}\right.  \tag{3}\\
U_{f}(|\psi\rangle|-\rangle)=\frac{1}{2^{k / 2}} \sum_{x=0}^{2^{k}-1}(-1)^{f(x)}|x\rangle|-\rangle \tag{4}
\end{gather*}
$$

3. The Diffusion Operator serves to amplify the amplitude of the target key state indicated by the oracle, identifying it by flipping the sign of said amplitude to negative. The quantum circuit for the diffusion operator is typically straightforward and does not require any special techniques to implement. Additionally, the overhead of the diffusion operator is usually negligible compared to the oracle, and therefore it is generally ignored when estimating the cost of Grover's algorithm [5,7,15]. Lastly, the Grover's algorithm provides a high probability of measuring the solution key by performing a sufficient number of iterations of the oracle and the diffusion operator to amplify the amplitude of the target key state.

### 2.4 NIST Security Criteria

NIST establishes security levels and estimates the required resources for block cipher and hash function attack costs for post-quantum security [3]. The estimates provided by NIST for the security levels defined and the number of classical and quantum gates for the attacks are as follows:

- Level 1: Any attempt to breach the defined security standards should necessitate computational capabilities equal to or greater than those required to perform a key search on a 128-bit key block cipher, such as AES128. $\left(\mathbf{2}^{170} \rightarrow \mathbf{2}^{157}\right)$.
- Level 3: Any attempt to breach the defined security standards should necessitate computational capabilities equal to or greater than those required to perform a key search on a 192-bit key block cipher, such as AES192. $\left(\mathbf{2}^{233} \rightarrow \mathbf{2}^{221}\right)$.
- Level 5: Any attempt to breach the defined security standards should necessitate computational capabilities equal to or greater than those required to perform a key search on a 256 -bit key block cipher, such as AES256. $\left(\mathbf{2}^{298} \rightarrow \mathbf{2}^{\mathbf{2 8 5}}\right)$.

Level 1, 3, and 5 are based on the Grover's key search cost for AES, while Level 2 and 4 rely on the collision attack cost for SHA3. Additionally, for Levels 2 and 4, estimates are provided only for classical gates, not quantum attacks. In our implementation of SEED, which is a symmetric key cipher, we primarily focus on Levels 1,3 , and 5 .

NIST sets the Grover's key search cost for AES-128, 192, and 256 based on the quantum circuits implemented by Grassl et al. [5], resulting in Levels 1, 3, and 5. During the execution of the Grover's key search, the number of gates and depth continue to increase, while the number of qubits remains constant. Therefore, the estimates provided by NIST are derived from the product of the total gates and total depth of the quantum circuit, excluding the number of qubits(AES-128, 192, and 256 as $\mathbf{2}^{\mathbf{1 7 0}}, \mathbf{2}^{\mathbf{2 3 3}}, \mathbf{2}^{\mathbf{2 9 8}}$, respectively).

The estimates for Grover's key search on the quantum circuit from [5], which NIST used as a basis for setting security levels, are notably high. Subsequent efforts to optimize AES quantum circuits have led to a reduction in the cost of quantum attacks. In 2019, Jaques et al. presented optimized quantum circuits for AES at Eurocrypt '20 [16]. Based on this, NIST redefines the quantum attack costs for AES-128, 192, and 256 as $\mathbf{2}^{\mathbf{1 5 7}}, \mathbf{2}^{\mathbf{2 2 1}}, \mathbf{2}^{\mathbf{2 8 5}}$, respectively [4].

Moreover, NIST proposes a restriction on circuit depth known as MAXDEPTH. This restriction stems from the challenge of executing highly prolonged sequential computations. In other words, it arises from the challenge of prolonged calculations due to sequential repetitions of quantum circuits in Grover's key search (especially in the Grover oracle). The MAXDEPTH specified by NIST is as follows. $\left(2^{40}<\right.$ $2^{64}<2^{96}$ )

## 3 Proposed Quantum Implementation of SEED

In this section, we present our optimized quantum circuit implementation of SEED. Our optimization goal in implementation is to minimize the depth while allowing a reasonable number of qubits.

### 3.1 Implementation of S-box

In quantum computers, the utilization of look-up table-based methods for implementing S-boxes is not appropriate. Thus, we employ quantum gates to implement the S-boxes based on Boolean expression of Equation 1. We use $x^{247}$ or $x^{251}$ in the S-box implementation, and these values can be expressed using primitive polynomials in $\mathbb{F}_{2^{8}} /\left(x^{8}+x^{6}+x^{5}+x+1\right)$ as follows: The S-boxes are in $\operatorname{GF}\left(2^{8}\right)$, so they can be modified with inversion as follows:

$$
\begin{align*}
& \left(x^{-1}\right)^{8} \equiv x^{247} \quad \bmod p(x) \\
& \left(x^{-1}\right)^{4} \equiv x^{251} \quad \bmod p(x)  \tag{5}\\
& p(x)=x^{8}+x^{6}+x^{5}+x+1
\end{align*}
$$

We can obtain the value by multiplying the inverse by the square. And then, following the Itoh Tsujii inversion algorithm [17], the $x^{-1}$ can be computed:

$$
\begin{equation*}
x^{-1}=x^{254}=\left(\left(x \cdot x^{2}\right) \cdot\left(x \cdot x^{2}\right)^{4} \cdot\left(x \cdot x^{2}\right)^{16} \cdot x^{64}\right)^{2} \tag{6}
\end{equation*}
$$

To compute the inversion of $x$, squaring and multiplication are used (as shown in Equation 6). In squaring, modular reduction can be employed PLU factorization because it is a linear operation. By using PLU factorization, it can be implemented without allocating additional ancilla qubits (i.e., in-place), using only the CNOT gates. Upon applying the PLU factorization, we obtain the following:

$$
\left(\begin{array}{lllllllll}
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0  \tag{7}\\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{array}\right)=\left(\begin{array}{lllllllll}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{array}\right) \cdot\left(\begin{array}{cccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1
\end{array}\right) \cdot\left(\begin{array}{llllllll}
1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right)
$$

These three matrices consist of a permutation matrix, a lower triangular matrix, and an upper triangular matrix, respectively. Figure 6 demonstrates the implementation of quantum circuit of squaring using only CNOT gates, utilizing these three matrices.


Fig. 6: Squaring in $\mathbb{F}_{2^{8}} /\left(x^{8}+x^{6}+x^{5}+x+1\right)$

For implementing multiplication in quantum, we adopt the method proposed in [18], which employs the Karatsuba multiplication instead of schoolbook multiplication [19]. The Karatsuba algorithm, when applied in the context of quantum computers, can lead to a reduction in the number of Toffoli gates (as it decrease the number of AND operations). This efficiency makes it a valuable technique for quantum computing.

In [18], a special Karatsuba algorithm is used, which enables the quantum multiplication with a Toffoli depth of one. By applying the Karatsuba algorithm recursively, all the AND operations for multiplication become independent. Additionally, by allocating more ancilla qubits, it becomes possible to operate all Toffoli gates in parallel, leading to a Toffoli depth of one.

Actually, allocating additional ancilla qubits is a known overhead in their method [18]. However, it is important to note that their method is more effective when used in conjunction with other operations rather than as a stand-alone multiplication. The authors of [18] mention that the ancilla qubits allocated for multiplication can be initialized (i.e., clean state) using reverse operations. This means that if it is not a stand-alone multiplication, the ancilla qubits can be reused in ongoing operations.

In Equation 6, multiple multiplications are performed to compute the inverse of the input $x$. Indeed, the method proposed in [18] is well-suited for implementing quantum circuits for inversion. Concretely, in our implementation, ancilla qubits are allocated only once for the initial multiplication $\left(x \cdot x^{2}\right)$, and for subsequent multiplications, the initialized ancilla qubits are reused without incurring any additional cost.

As a result, we successfully optimize the number of qubits and the Toffolirelated metrics such as, the number of Toffoli gates, Toffoli depth, and full depth ${ }^{1}$.

Using these methods of squaring and multiplication, we can obtain the exponentiation values ( $x^{247}$ and $x^{251}$ ). And then, we compute the multiplication of the exponentiation values ( $x^{247}$ and $x^{251}$ ) and the matrices $\left(A^{(1)}\right.$ and $\left.A^{(2)}\right)$. Since the matrices $A^{(1)}$ and $A^{(2)}$ are constant, the matrix-vector multiplication (classical-quantum) can be implemented in-place without requiring additional

[^1]qubits. We apply the PLU factorization to the matrices $A^{(1)}$ and $A^{(2)}$, similar to how we implemented the quantum circuit for squaring (Equation 7).

### 3.2 Implementation of $G$ function

In the $G$ function, four S-boxes (two $S 1$ and two $S 2$ ) are used, and the implementation of these S-boxes follows the method described in Section 3.1. Each S-box requires 38 ancilla qubits, which can be initialized using reverse operations, enabling their reuse. Therefore, if the four S-boxes are implemented sequentially, the number of ancilla qubits can be saved by using only 38 of them. However, in this case, the depth of the circuit increases due to the sequential operations. Thus, considering the trade-off, we implement the four S-boxes in parallel to reduce the circuit depth. This is achieved by allocating a total of $152(38 \times 4)$ ancilla qubits at first. Additionally, these ancilla qubits are initialized (i.e., returning to 0 ), allowing the 4 sets of ancilla qubits to be reused in the G function of the next round.

### 3.3 Implementation of Key Schedule

Algorithm 1 describes the proposed quantum circuit implementation of the key schedule. In the key schedule, two 32 -qubit subkeys ( $K_{i, 0}$ and $K_{i, 1}$ ) are generated. To reduce the circuit depth, the implementation is parallelized by operating two processes simultaneously. For this, we allocate two sets of 152 ancilla qubits to implement two $G$ functions in parallel. Also, for parallel processing, the operations with KeyConstant values of quantum state also need to be implemented in parallel. To enable parallel processing, two pairs of qubits $(32 \times 2)$ are allocated to store the KeyConstant values (using on $K_{i, 0}, K_{i, 1}$ respectively) in our implementation.

Due to the different KeyConstant values used in each round, it is necessary to allocate and store new qubits every time. Instead of allocating new qubits in each round, we utilize reverse operations to initialize and reuse the qubits. The reverse operation for the KeyConstant of quantum state involves only X gates, which have a trivial overhead on the circuit depth. Thanks to this approach, we can effectively parallelize the quantum circuit for the key schedule, resulting in a reduced circuit depth while using a reasonable number of qubits.

For implementing addition in quantum, we utilize the CDKM adder [20], an enhanced version of the quantum ripple-carry adder, which is implemented using X, CNOT, and Toffoli gates. The CDKM adder proves to be effective for $n$-qubit addition when $n \geq 4$, making it a suitable choice for SEED, where $n=8$. This adder requires only one ancilla qubit and optimizes the circuit depth. Specifically, it utilizes one ancilla qubit, $(2 n-3)$ Toffoli gates, $(5 n-7)$ CNOT gates, $(2 n-6)$ X gates, and achieves a circuit depth of $(2 n+3)$.

In Shift operation, it can be implemented using swap gates, but in our approach, we utilize logical swaps that change the index of qubits, avoiding the use of quantum gates.

```
Algorithm 1: Quantum circuit implementation of SEED Key Schedule.
Input: \(A, B, C, D, c_{0}, c_{1}\) ancilla \(_{0}\), ancilla \(a_{1}\)
Output: \(k e y_{0}, k e y_{1}, C, D\)
//Each operation in parallel.
    for \(0 \leq i \leq 16\) do
        \(K C \_Q_{0} \leftarrow\) Constant_XOR \(\left(K C[i], K C \_Q_{0}\right)\)
        \(K C_{-} Q_{1} \leftarrow\) Constant_XOR \(\left(K C[i], K C_{-} Q_{1}\right)\)
        \(C_{2} \leftarrow\) allocate new 32 qubits
        \(D_{2} \leftarrow\) allocate new 32 qubits
        \(C_{2} \leftarrow \operatorname{Copy32}\left(C, C_{2}\right)\)
        \(D_{2} \leftarrow \operatorname{Copy32}\left(D, D_{2}\right)\)
        \(C_{2} \leftarrow \operatorname{CDKM}\left(A, C_{2}, c_{0}\right)\)
        \(D_{2} \leftarrow\) CDKM_minus \(\left(B, D_{2}, c_{1}\right)\)
        \(C_{2} \leftarrow\) CDKM_minus \(\left(K C \_Q_{0}, C_{2}, c_{0}\right)\)
        \(D_{2} \leftarrow \operatorname{CDKM}\left(K C_{-} Q_{1}, D_{2}, c_{1}\right)\)
        \(k e y_{0} \leftarrow G\) function \(\left(C_{2}\right.\), ancilla \(\left.a_{0}\right)\)
        \(k e y_{1} \leftarrow G\) function \(\left(D_{2}\right.\), ancilla \(\left.{ }_{1}\right)\)
        //Initialize qubitsthrough reverse to reuse.
        \(K C \_Q_{0} \leftarrow\) Constant_XOR \(\left(K C[i], K C \_Q_{0}\right)\)
        \(K C-Q_{1} \leftarrow\) Constant_XOR \(\left(K C[i], K C \_Q_{1}\right)\)
        if i \(\% 2==0\) then
            \(\operatorname{RightShift}(A, B) \quad \triangleright\) logical Swap
        else
            LeftShift \((C, D) \quad \triangleright\) logical Swap
        return \(k e y_{0}, k e y_{1}, C, D\)
```


## 4 Performance of the Proposed Quantum Circuits

In this part, we present the performance of our SEED quantum circuit implementation. Our proposed quantum circuits of cryptographys are implemented using the ProjectQ tool provided by IBM. ProjectQ provides ClassicalSimulator, which can simulate simple quantum gates mentioned in Section 2.2, and ResourceCounter, which can measure circuit resources, as internal libraries. ClassicalSimulator has the advantage of providing enough quantum resources to run our proposed quantum circuit. Real quantum computers still provide limited quantum resources that are not sufficient to run cryptography. Therefore, the circuits are run through the simulator provided by ProjectQ and the quantum resources are measured.

Table 1 and 2 show the quantum resources required to implement our SEED quantum circuits. Table 1 provides a comprehensive analysis of quantum resources

Table 1: Required quantum resources for SEED quantum circuit implementation

| Cipher | \#X | \#CNOT | \#Toffoli | Toffoli depth | \#Qubit | Depth | TD- $M$ cost |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| SEED | 8116 | 409,520 | 41,392 | 321 | 41,496 | 11,837 | $13,320,216$ |

Table 2: Required decomposed quantum resources for SEED quantum circuit implementation

| Cipher | \#Clifford | $\# T$ | $T$-depth | \#Qubit | Full depth | $F D$ - $M$ cost |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| SEED | 748,740 | 289,680 | 1,284 | 41,496 | 34,566 | $1,434,350,736$ |

at the NCT (NOT, CNOT, Toffoli) level. The Toffoli gate can be decomposed into 8 Clifford gates and 7 T gates and Table 2 presents the decomposed quantum resource costs for the quantum circuit implementation of SEED. Additionally, our implementation focuses on optimizing the circuit depth while considering the trade-off for using qubit, and we also perform metrics to evaluate these trade-offs such as Toffoli depth $\times$ qubit count $(T D \times M)$ and full depth $\times$ qubit count $(F D \times M)$.

## 5 Evaluation of Grover's Search Complexity

We adopt the methodology detailed in Section 2.3 to estimate the cost of Grover's key search for SEED. Grover's search can be estimated based on our implemented SEED quantum circuit. Since the overhead of the diffusion operator can be considered insignificant compared to the oracle when most of the quantum resources are used for implementing the target cipher in the quantum circuit, it can be disregarded.

Additionally, the Grover oracle is comprised of two consecutive executions of the SEED quantum circuit. The first one constitutes the encryption circuit, while the second one is the reverse operation of encryption circuit to return back to the state prior to encryption. Therefore, the oracle requires twice the cost of implementing a quantum circuit, not including of qubits. The number of iterations of Grover key search for $k$-bit key length is about $\sqrt{2^{k}}$. In [21], Grover's key search algorithm was analyzed in detail and the optimal number of iterations was suggested to be $\left\lfloor\frac{\pi}{4} \sqrt{2^{k}}\right\rfloor$. In conclusion, including Grover iterations, the Grover's key search cost for SEED is approximately Table $2 \times 2 \times\left\lfloor\frac{\pi}{4} \sqrt{2^{k}}\right\rfloor$, as shown in Table 3.

## 6 Conclusion

We can assess the post-quantum security of SEED based on the cost of Grover's key search obtained earlier (in Section 5). In 2016, NIST defined post-quantum security levels by considering the estimated costs of Grover's key search attacks on AES-128, 192, and 256. Nevertheless, with the declining costs of AES attacks,

Table 3: Cost of the Grover's key search for SEED

| Cipher | Total gates | Total depth | Cost <br> (complexity) | \#Qubit | $T D$ - $M$ cost | $F D-M$ cost |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| SEED | $1.559 \cdot 2^{84}$ | $1.657 \cdot 2^{79}$ | $1.291 \cdot 2^{164}$ | 41,497 | $1.246 \cdot 2^{88}$ | $1.049 \cdot 2^{95}$ |

NIST revised the cost assessments to align with the respective security levels in 2019.

According to Table 3, the Grover's key search attack cost for SEED is calculated to be $1.291 \cdot 2^{164}$. This leads to the assessment that SEED attains post-quantum security Level 1.

In summary, this paper presents the first implementation of a quantum circuit for SEED. We focus on optimizing Toffoli and full depths utilizing parallelization and optimized multiplication, squaring and an adder. By analyzing the cost of Grover's key search attack, we confirm that SEED achieves post-quantum security Level 1. Furthermore, we provide $T D \times M$ and $F D \times M$ costs to consider the trade-off between depth and qubits.

## 7 Acknowledgment

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT).(No. RS-2023-00277994, Quantum Circuit Depth Optimization for ARIA, SEED, LEA, HIGHT, and LSH of KCMVP Domestic Cryptographic Algorithms, 80\%) and this work was supported by Institute of Information \& communications Technology Planning \& Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00627, Development of Lightweight BIoT technology for Highly Constrained Devices, $20 \%$ ).

## References

1. P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in Proceedings 35th Annual Symposium on Foundations of Computer Science, pp. 124-134, 1994. 1
2. L. K. Grover, "A fast quantum mechanical algorithm for database search," in Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996 (G. L. Miller, ed.), pp. 212-219, ACM, 1996. 1
3. NIST., "Submission requirements and evaluation criteria for the postquantum cryptography standardization process," 2016. https://csrc. nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/ call-for-proposals-final-dec-2016.pdf. 2, 7

14 Yujin Oh, Kyungbae Jang, Yujin Yang, and Hwajeong Seo
4. NIST., "Call for additional digital signature schemes for the post-quantum cryptography standardization process," 2022. https://csrc.nist.gov/csrc/media/ Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022. pdf. 2, 7
5. M. Grassl, B. Langenberg, M. Roetteler, and R. Steinwandt, "Applying Grover's algorithm to AES: quantum resource estimates," 2015. 2, 6, 7
6. S. Jaques, M. Naehrig, M. Roetteler, and F. Virdia, "Implementing grover oracles for quantum key search on aes and lowmc." Cryptology ePrint Archive, Report 2019/1146, 2019. https://eprint.iacr.org/2019/1146. 2
7. K. Jang, A. Baksi, G. Song, H. Kim, H. Seo, and A. Chattopadhyay, "Quantum analysis of aes," Cryptology ePrint Archive, 2022. 2, 6
8. K. Jang, G. Song, H. Kim, H. Kwon, H. Kim, and H. Seo, "Efficient implementation of present and gift on quantum computers," Applied Sciences, vol. 11, no. 11, p. $4776,2021.2$
9. K. Jang, A. Baksi, H. Kim, H. Seo, and A. Chattopadhyay, "Improved quantum analysis of speck and lowmc (full version)," Cryptology ePrint Archive, 2022. 2
10. R. Anand, A. Maitra, and S. Mukhopadhyay, "Evaluation of quantum cryptanalysis on speck," in Progress in Cryptology - INDOCRYPT 2020 (K. Bhargavan, E. Oswald, and M. Prabhakaran, eds.), (Cham), pp. 395-413, Springer International Publishing, 2020. 2
11. Y. Yang, K. Jang, H. Kim, G. Song, and H. Seo, "Grover on sparkle," in International Conference on Information Security Applications, pp. 44-59, Springer, 2022. 2
12. A. Jagielski and K. Kanciak, "Quantum resource estimation for a nist lwc call finalist," Quantum Information and Computation, vol. 22, no. 13\&14, pp. 1132-1143, 2022. 2
13. S. Roy, A. Baksi, and A. Chattopadhyay, "Quantum implementation of ascon linear layer," Cryptology ePrint Archive, 2023. 2
14. Y. Oh, K. Jang, A. Baksi, and H. Seo, "Depth-optimized implementation of ascon quantum circuit," Cryptology ePrint Archive, 2023. 2
15. S. Jaques, M. Naehrig, M. Roetteler, and F. Virdia, "Implementing grover oracles for quantum key search on aes and lowmc." Cryptology ePrint Archive, Report 2019/1146, 2019. https://eprint.iacr.org/2019/1146. 6
16. S. Jaques, M. Naehrig, M. Roetteler, and F. Virdia, "Implementing grover oracles for quantum key search on aes and lowmc," in Advances in Cryptology-EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II 30, pp. 280-310, Springer, 2020. 7
17. T. Itoh and S. Tsujii, "A fast algorithm for computing multiplicative inverses in gf (2m) using normal bases," Information and computation, vol. 78, no. 3, pp. 171-177, 1988. 8
18. K. Jang, W. Kim, S. Lim, Y. Kang, Y. Yang, and H. Seo, "Optimized implementation of quantum binary field multiplication with toffoli depth one," in International Conference on Information Security Applications, pp. 251-264, Springer, 2022. 9
19. D. Cheung, D. Maslov, J. Mathew, and D. K. Pradhan, "On the design and optimization of a quantum polynomial-time attack on elliptic curve cryptography," in Workshop on Quantum Computation, Communication, and Cryptography, pp. 96104, Springer, 2008. 9
20. S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, "A new quantum ripple-carry addition circuit," arXiv preprint quant-ph/0410184, 2004. 10
21. M. Boyer, G. Brassard, P. Høyer, and A. Tapp, "Tight bounds on quantum searching," Fortschritte der Physik, vol. 46, p. 493-505, Jun 1998. 12

# Depth-Optimized Quantum Implementation of ARIA 

Yujin Yang, Kyungbae Jang, Yujin Oh, and Hwajeong Seo<br>Division of IT Convergence Engineering, Hansung University, Seoul, South Korea<br>yujin. yang34@gmail.com, starj1023@gmail.com, oyj0922@gmail.com, hwajeong84@gmail.com


#### Abstract

The advancement of large-scale quantum computers poses a threat to the security of current encryption systems. In particular, symmetric-key cryptography significantly is impacted by general attacks using the Grover's search algorithm. In recent years, studies have been presented to estimate the complexity of Grover's key search for symmetrickey ciphers and assess post-quantum security. In this paper, we propose a depth-optimized quantum circuit implementation for ARIA, which is a symmetric key cipher included as a validation target the Korean Cryptographic Module Validation Program (KCMVP). Our quantum circuit implementation for ARIA improves the depth by more than $88.8 \%$ and Toffoli depth by more than $98.7 \%$ compared to the implementation presented in Chauhan et al.'s SPACE'20 paper. Finally, we present the cost of Grover's key search for our circuit and evaluate the post-quantum security strength of ARIA according to relevant evaluation criteria provided NIST.


Keywords: Depth-Optimized Quantum Circuit • Korean Block Ciphers - ARIA • Grover's Search Algorithm.

## 1 Introduction

Quantum computers, built upon principles of quantum mechanics like quantum superposition and entanglement, have the capability to solve specific problems at a faster rate compared to classical computers. As a result, many companies and research institutions are concentrating on quantum computer development. However, it is known that the advancement of large-scale quantum computers has the potential to pose a threat to the security of current cryptographic systems. In particular, symmetric-key cryptography can be significantly compromised by general attacks using the Grover's search algorithm, which can reduce the data search complexity. As a result, in recent years, studies have been presented to estimate the complexity of recovering secret keys in existing symmetric-key ciphers using the Grover's search algorithm and evaluate post-quantum security based on these findings $[8,10,11,14,15,22,25]$.

ARIA is a symmetric-key cryptography algorithm optimized for ultra-light environments and hardware implementation, and is included as a validation
target in the Korean Cryptographic Module Validation Program (KCMVP). This means that ARIA is widely used in verified cryptographic modules, so it is very important to measure ARIA's quantum security strength for future preparedness against emerging threats. Fortunately, there is already a study that measured the quantum security strength of ARIA in 2020 [2]. However, since [2] primarily focuses on qubit optimization, there is also a need for research that addresses the recent emphasis on optimizing depth.

In a document guiding evaluation criteria for post-quantum cryptography standardization, NIST provided a criteria for estimating quantum attack complexity and proposed a parameter called MAXDEPTH, which refers to the maximum circuit depth that a quantum computer can execute. In order to evaluate the strength of quantum security, not only the quantum attack complexity but also the MAXDEPTH related to execution must be considered. Further elaboration on this topic can be found in Sections 2.4 and 4.

The paper is structured as follows. Section 2 offers the background for this paper. Section 2.1 provides an introduction to ARIA. In Section 2.2, the quantum gates utilized to implement quantum circuits are covered. In Section 2.3 Grover's key search is examined because it relates to measuring quantum resources, and in Section 2.4, NIST post-quantum security and MAXDEPTH are covered because they are crucial for estimating security strength. Following this, in Sections 3, the design of quantum circuits for ARIA is suggested, drawing upon the information presented in Section 2. Section 4.2 presents the cost of Grover's key search for our circuit and evaluates ARIA's post-quantum security strength based on the estimates. Lastly, Section 5 delves into the summarizing conclusions and outlines potential directions for future research.

### 1.1 Our Contribution

This paper makes the following contributions:

1. Low depth quantum implementation of ARIA. In our implementation of the ARIA quantum circuit, our main focus is minimizing the Toffoli depth and ensuring full depth. We achieve a reduction in Toffoli depth and full depth through various techniques for optimization.
2. Various techniques for optimization. We utilize various techniques for optimization to reduce the depth. For optimizing binary field operations, we choose a multiplication optimizer that implements the Karatsuba algorithm in parallel and a squaring method using linear layer optimization methods (PLU factorization, XZLBZ). Furthermore, we enhance implementing parallel processing for applicable components.
3. Evaluation of post-quantum security. We estimate the resources required for implementing quantum circuits for ARIA. The resource estimation for the ARIA quantum circuit also includes the comparison with previous research. Furthermore, we evaluate the quantum security of ARIA by estimating the cost of Grover's key search based on the implemented quantum circuit and comparing them with the security levels provided by NIST.

## 2 Background

### 2.1 ARIA Block Cipher

ARIA [17], which stands for Academy, Research Institute, and Agency, is a Korean symmetric key block cipher jointly developed by the three organizations mentioned above. Since the adoption of ARIA as a national standard encryption algorithm in 2004, it has been widely used for secure communication and data protection. Especially, ARIA holds significance as symmetric key ciphers included in the validation subjects of the KCMVP. ARIA has an interface similar to AES, a symmetric key block cipher standard, because its designers considered the design principles of AES during its development. It has an Involutional Substitution-Permutation Network (ISPN) structure optimized for lightweight hardware implementation. The input/output size of ARIA is fixed at 128 -bit, and only the key size is different as 128,192 , and 256 -bit.

Round Function The round function is made of three main operations: $A d$ dRoundKey, Substitution layer, and Diffusion layer.

In the AddRoundKey, the round key suitable for each round is XORed to intermediate state.

In the Substitution layer, the input 128 -bit state is divided into 8 -bit units, and substitutions are performed using the S-boxes. ARIA employs a total of four S-boxes $\left(S_{1}, S_{1}^{-1}, S_{2}, S_{2}^{-1}\right)$, which include the inverse S-boxes. The $S_{1}, S_{1}^{-1}$ are identical to the ones used in AES, and the $S_{2}, S_{2}^{-1}$ are newly designed S-boxes specifically for ARIA. These S-boxes used in ARIA are generated by applying an affine transformation to the functions $x^{-1}$ and $x^{247}$ over $G F\left(2^{8}\right)$. The S-boxes $S_{1}(x), S_{2}(x)$ are obtained by performing multiplication between $8 \times 8$ non-singular matrix ( $\mathbf{A}$ or $\mathbf{B}$ ) and the function $\left(x^{-1}\right.$ or $x^{247}$ ), followed by XOR with $8 \times 1$ vector. This can be expressed as follows:

$$
\begin{aligned}
& S_{1}(x)=\mathbf{A} \cdot x^{-1} \oplus[1,1,0,0,0,1,1,0]^{\mathrm{T}}, \\
& S_{2}(x)=\mathbf{B} \cdot x^{247} \oplus[0,1,0,0,0,1,1,1]^{\mathrm{T}}
\end{aligned}
$$

$$
\text { where } \quad \mathbf{A}=\left(\begin{array}{llllllll}
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1  \tag{1}\\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1
\end{array}\right) \quad \text { and } \quad \mathbf{B}=\left(\begin{array}{llllllll}
0 & 1 & 0 & 1 & 1 & 1 & 1 & 1
\end{array}\right)
$$

ARIA features two types of S-box layers consisting of four S-boxes. Type 1 comprises four 32 -bit sets consisting of $S_{1}, S_{2}, S_{1}^{-1}$, and $S_{2}^{-1}$ in this order. Since the two types are the inverse relationship to each other, Type 2 is the inverse of

Type 1 (i.e., Type1 ${ }^{-1}=$ Type2). Type 1 is used for odd rounds and Type 2 for even rounds in the round function. The two types of S-box layers in ARIA are shown in Figure 1.

(a) Type 1

(b) Type 2

Fig. 1: Two types of S-box layers in ARIA

The Diffusion layer performs byte-wise matrix multiplication by multiplying the given $16 \times 16$ involution binary matrix with the output of the substitution layer. The involution binary matrix does not require a separate implementation of the inverse matrix during the decryption process, as its inverse matrix is the same as itself.

The detailed composition of the round function differs depending on whether the round is odd, even, or final. The main difference between odd and even rounds lies in the type of the S-box layer used: odd rounds use a Type 1, whereas even rounds use a Type 2. In the final round, the diffusion step is omitted and the AddRoundKey is performed once more. A brief outline of the round function of ARIA is shown in Figure 2.


Fig. 2: Brief outline of round function of ARIA.

Key Schedule In the key initialization step (Figure 3), 128-bit initial constants $W_{0}, W_{1}, W_{2}$, and $W_{3}$ are generated as essential components for generating a round key. During this step, the round functions $F_{o}$ and $F_{e}$ are utilized.

$$
\begin{equation*}
K L\|K R=M K\| 0 \cdots 0 \tag{2}
\end{equation*}
$$

Equation 2 represents the formula used to generate the input values $K L$ and $K R$ in the key initialization step. This equation is derived from the master key $M K$. Since the concatenated result of $K L$ and $K R$, which are each 128-bit, is fixed to 256 -bit (i.e., $K L \| K R$ ), if $M K$ is smaller than 256 , padding is performed to match the size by filling the insufficient bits with 0 s. The 128 -bit initial round constant keys $C K_{1,2,3}$ are the 128 -bit constant values of the rational part of $\pi^{-1}$. The order of using the 128 -bit initial round constant keys $C K_{1,2,3}$ depends on the length of $M K$. Figure 3 shows the key initialization step.


Fig. 3: Key Initialization of ARIA

In the key generation phase, a round key is generated and used as the key for each round. The round keys $e k_{1 \sim 17}$ are obtained by applying rotations ( $\ll, \gg$ ) and XOR operations to the initial constants $W_{0 \sim 3}$ generated during the key initialization step.

The round key in all ARIA instances has a size of 128 bits. The number of rounds for ARIA-128, 192, and 256 are 12, 14, and 16, respectively. Additionally,
an extra round key is used in the AddRoundKey operation for the final round, resulting in a total of 13,15 , and 17 round keys for ARIA-128, 192, and 256, respectively. The round keys $e k_{i}$ are generated as follows:

$$
\begin{array}{ll}
e k_{1}=\left(W_{0}\right) \oplus\left(W_{1} \ggg 19\right), & e k_{2}=\left(W_{1}\right) \oplus\left(W_{2} \ggg 19\right) \\
e k_{3}=\left(W_{2}\right) \oplus\left(W_{3} \ggg 19\right), & e k_{4}=\left(W_{0} \ggg 19\right) \oplus\left(W_{3}\right) \\
e k_{5}=\left(W_{0}\right) \oplus\left(W_{1} \ggg 31\right), & e k_{6}=\left(W_{1}\right) \oplus\left(W_{2} \ggg 31\right) \\
e k_{7}=\left(W_{2}\right) \oplus\left(W_{3} \ggg 31\right), & e k_{8}=\left(W_{0} \ggg 31\right) \oplus\left(W_{3}\right) \\
e k_{9}=\left(W_{0}\right) \oplus\left(W_{1} \ll 61\right), & e k_{10}=\left(W_{1}\right) \oplus\left(W_{2} \ll 61\right) \\
e k_{11}=\left(W_{2}\right) \oplus\left(W_{3} \ll 61\right), & e k_{12}=\left(W_{0} \ll 61\right) \oplus\left(W_{3}\right) \\
e k_{13}=\left(W_{0}\right) \oplus\left(W_{1} \ll 31\right), & e k_{14}=\left(W_{1}\right) \oplus\left(W_{2} \ll 31\right) \\
e k_{15}=\left(W_{2}\right) \oplus\left(W_{3} \ll 31\right), & e k_{16}=\left(W_{0} \ll 31\right) \oplus\left(W_{3}\right) \\
e k_{17}=\left(W_{0}\right) \oplus\left(W_{1} \ll 19\right) &
\end{array}
$$

### 2.2 Quantum Gates

Since in the quantum computer environment they do not provide logic gates such as AND, OR, and XOR, quantum gates are used as replacements for logic gates. This section describes commonly used quantum gates (Figure 4) for implementing quantum circuits of block ciphers (note that this is not an exhaustive list of all possible gates that can be used).

The X gate acts like a NOT operation on a classical computer, reversing the state of the qubit that goes through it. The Swap gate exchanges the states of two qubits. The CNOT gate behaves like an XOR operation on a classical computer. In $\operatorname{CNOT}(a, b)$, the input qubit $a$ is the control qubit, and $b$ is the target qubit. When the control qubit $a$ is in the state 1 , the target qubit $b$ is flipped. As a result, the value of $a \oplus b$ is stored in the qubit $b$ (i.e., $b=a \oplus b$ ), while the state of qubit $a$ remains unchanged. The Toffoli gate, represented as Toffoli $(a, b, c)$, acts like an AND operation on a classical computer. It requires three input qubits, with the first two qubits ( $a$ and $b$ ) serving as control qubits. Only when both control qubits are in the state 1 , the target qubit $c$ is flipped. The result of the operation $a \& b$ is XORed with the qubit $c$ (i.e., $c=c \oplus(a \&$ $b)$ ), while the states of qubits $a$ and $b$ are preserved. The Toffoli gate consists of 8 Clifford gates and $7 T$ gates. The $T$-count of the standard Toffoli gate [18] is 7 and the $T$-depth is 6 . Many studies are reporting the implementation of Toffoli gate circuits with minimized depth and $T$-depth $[1,7,16,21,23]$.

### 2.3 Grover's Key Search

Grover's search algorithm is a quantum algorithm that enables rapid searching for specific data within an unstructured database set $N$, reducing the search complexity from $O(N)$ to $O(\sqrt{N})$. When applied to an $n$-bit secret key search in symmetric key encryption, it reduces the search complexity from $O\left(2^{n}\right)$ resulting from a brute-force attack to $O\left(2^{n / 2}\right)$, halving the security level in theory. Grover's key search algorithm operates in three sequential steps as follows:


Fig. 4: Quantum gates: X (left top), Swap (right top), CNOT (left bottom) and Toffoli (right bottom) gates.

1. Initialization: Input the $n$-qubit key into the Hadamard gate to create a superposition of states $|\psi\rangle$ in which all $2^{n}$ computational basis states have equal amplitudes.

$$
\begin{gather*}
H|0\rangle=\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)  \tag{4}\\
|\psi\rangle=(H|0\rangle)^{\otimes n}=\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)^{\otimes n}=\frac{1}{\sqrt{2^{n}}} \sum_{x=0}^{2^{n}-1}|x\rangle \tag{5}
\end{gather*}
$$

2. Oracle Operator: The quantum circuit for the target cipher encrypts the known plaintext using keys (prepared keys) generated through a superposition of states in the Oracle and produces ciphertext for all key values. Within the Oracle operator ( $U_{f}$ ), the function $f(x)$ in Equation 6 compares the ciphertext generated by the circuit to the known ciphertext. The function $f(x)$ returns 0 if the generated ciphertext and the known ciphertext do not match and 1 if they do. When a match is identified, the state of the corresponding key in Equation 7, i.e., its amplitude, becomes negative because $f(x)$ is equal to 1 . If no match is found, $(-1)^{0}$ equals 1 , so the amplitude remains positive.

$$
\begin{gather*}
f(x)=\left\{\begin{array}{l}
1 \text { if } E n c_{\text {key }}(p)=c \\
0 \text { if } E n c_{\text {key }}(p) \neq c
\end{array}\right.  \tag{6}\\
U_{f}(|\psi\rangle|-\rangle)=\frac{1}{\sqrt{2^{n}}} \sum_{x=0}^{2^{n}-1}(-1)^{f(x)}|x\rangle|-\rangle \tag{7}
\end{gather*}
$$

3. Diffusion Operator: The diffusion operator $(D)$ serves the purpose of transforming a key state (target key state) with a negative amplitude into a symmetric state. This transformation involves computing the average value of all key states $(|s\rangle)$ and then subtracting this average value from each key state element $(I)$. During the second step, if the amplitude of the key state is initially negative, subtracting a negative number results in a positive value, thereby amplifying only the amplitude of that value.

$$
\begin{equation*}
D=2|s\rangle\langle s|-I \tag{8}
\end{equation*}
$$

In order to increase the probability of measuring the solution key, steps 2 and 3 must be repeated sufficiently. In general, when the number of repetitions is $\frac{\pi}{4} \sqrt{2^{k}}$, it has the highest measurement probability.

### 2.4 NIST Post-quantum Security and MAXDEPTH

In order to analyze the algorithms submitted during the post-quantum cryptography standardization, NIST provided security standards based on the security strength range specified in the existing NIST standard for symmetric cryptography in a related document [19, 20]. This post-quantum security baseline is based on the complexity of quantum attacks against AES and SHA- $2 / 3$ variants. The following is a summary of the criteria for estimating the complexity of quantum attacks provided in NIST's document [20] :

- Level 1: Any attempt to compromise the applicable security definition should demand computational resources that are equal to or exceed the resources needed to conduct a key search on a 128 -bit key block cipher, such as AES-128.
- Level 3: Any attempt to compromise the applicable security definition should demand computational resources that are equal to or exceed the resources needed to conduct a key search on a 192-bit key block cipher, such as AES-192.
- Level 5: Any attempt to compromise the applicable security definition should demand computational resources that are equal to or exceed the resources needed to conduct a key search on a 256 -bit key block cipher, such as AES-256.

Grover's search algorithm is recognized as one of the most efficient quantum attacks for targeting symmetric key ciphers, and NIST also acknowledges this fact. The difficulty presented by attacks at Levels 1,3 , and 5 is assessed according to the cost needed for Grover's key search on AES-128, 192, and 256, respectively. This cost is determined by multiplying the total gate count by the depth of Grover's key search circuit. Through studies published over the past few years that optimized AES quantum circuits to reduce Grover's key search costs, NIST has defined the costs for Levels 1,3 , and 5 as $\mathbf{2}^{\mathbf{1 5 7}}, \mathbf{2}^{\mathbf{2 2 1}}$, and $\mathbf{2}^{\mathbf{2 8 5}}$, respectively in their recent document [20] by citing the costs of depth-optimized quantum circuit implementations for AES [15] presented by Jaques et al. at Eurocrypt'20.

It should be mentioned that the quantum circuit implementation by Jaques et al. [15] has a few programming-related problems. Nevertheless, Jang et al. addressed and examined these issues in their research [11], showing that the cost values mentioned in [15] can be roughly achieved using their optimized AES quantum circuits. As far as our current understanding goes, the most notable outcomes are detailed in [11] (Level 1,3 , and 5 cost $\mathbf{2}^{157}, \mathbf{2}^{\mathbf{1 9 2}}, \mathbf{2}^{\mathbf{2 7 4}}$ ).

Additionally, we must also consider an approach called MAXDEPTH. NIST introduced a parameter called MAXDEPTH, which signifies the maximum circuit depth the quantum computer is able to execute, as an excessively large depth can
lead to execution challenges in terms of time. The depth limits (i.e. MAXDEPTH) for quantum attacks provided by NIST range as follows: (no more than $\mathbf{2}^{\mathbf{4 0}}, \mathbf{2}^{\mathbf{6 4}}$, $2^{96}$ ).

## 3 Proposed Quantum Implementation of ARIA

This section describes our optimized quantum circuit implementation of ARIA. We compare the results of the previous work [2], which implemented ARIA as a quantum circuit, and examine the optimized components.

### 3.1 Implementation of S-box

In classical computers, the S-box of most block ciphers, including AES, employs a predefined look-up table. However, in a quantum computing environment, it is more efficient to implement the S-box using multiplicative inversion and affine transformation, primarily because of the limited number of qubits [5].

While the tool LIGHTER-R [6] efficiently constructs quantum circuits based on existing S-boxes, it has a limitation in its applicability, as it can only be used with 4-bit S-boxes, making it unsuitable for ARIA's S-box. The recent studies $[4,26]$ aim to enhance LIGHTER-R tools to build quantum circuits for S-boxes that are previously beyond its reach. However, since these tools also concentrate on 4 -bit S-boxes, they cannot be employed to implement quantum circuits for ARIA's S-box. Therefore, it is necessary to obtain the multiplicative inverse and perform the affine transformation to implement the quantum circuit of ARIA's S-box.

In order to find ARIA's S-box $S_{1}$ and $S_{2}$, The inverse of $x$ (i.e., $x^{-1}$ ) and the exponentiation value $x^{247}$ of Equation 1 in Section 2.1 must be obtained. $x^{247}$ in $S_{2}$ can be expressed as follows using the primitive polynomial $m(x)$ in the environment of $G F\left(2^{8}\right)$ :

$$
\begin{align*}
& x^{247} \equiv\left(x^{-1}\right)^{8} \equiv\left(\left(\left(x^{-1}\right)^{2}\right)^{2}\right)^{2} \quad \bmod m(x) \\
& m(x)=x^{8}+x^{4}+x^{3}+x+1 \tag{9}
\end{align*}
$$

Likewise, the multiplicative inverse of x in the environment of $G F\left(2^{8}\right)$ is equal to $x^{254}$ The multiplicative inverse can be efficiently obtained using the Itoh-Tsujii algorithm [9]. Therefore, by applying the Itoh-Tsuji algorithm, it can be expressed as an expression consisting of square and multiplication as follows:

$$
\begin{equation*}
x^{-1}=x^{254}=\left(\left(x \cdot x^{2}\right) \cdot\left(x \cdot x^{2}\right)^{4} \cdot\left(x \cdot x^{2}\right)^{16} \cdot x^{64}\right)^{2} \tag{10}
\end{equation*}
$$

In order to increase the operation speed, the squaring operation is generally performed by converting the irreducible polynomial having linearity through modular reduction into a matrix form. Since this corresponds to a linear operation, it can be implemented as an in-place structure using only the XOR operation by using the XZLBZ [24], a heuristic search algorithm based on factorization in binary matrices.

The squaring operation in ARIA is implemented using CNOT gates and SWAP gates through modular reduction and XZLBZ [24]. This implementation utilizes 10 CNOT gates and has a circuit depth of 7 . Figure 5 depicts the quantum circuit for the squaring operation in ARIA.


Fig. 5: Quantum circuit implementation for Squaring in $\mathbb{F}_{2^{8}} /\left(x^{8}+x^{4}+x^{3}+x+1\right)$

For multiplication, Jang et al.'s Toffoli-depth optimized Karatsuba quantum multiplication [13], first announced at WISA'22, is used. By employing the Karatsuba algorithm, which is known for reducing the number of multiplications, the number of Toffoli gates required for a multiplication can be reduced. Jang et al.'s multiplication applies the Karatsuba algorithm recursively to perform all multiplications, i.e., AND operations, independently. In order to achieve a Toffoli depth of 1 , more ancilla qubits are allocated to execute Toffoli gates in parallel. This method is only used for multiplications between quantum-quantum values.

Compared to the previous quantum implementation of ARIA [2], squaring uses the same method, so the resources used are the same, but the multiplication operation differs. In [2], the authors employed the schoolbook multiplication method [3]. In contrast, in our work, by adopting the Toffoli-depth optimized Karatsuba multiplication [13], we achieve a significant reduction in quantum resources. Table 1 compares the quantum resources required for multiplication by adopting different methods [3,13]. In Table 1, we can see that overall quantum resources have been reduced, and, in particular, Toffoli-depth have been optimized.

Table 1: Quantum resources required for multiplication.

| Source | \#Clifford | \#T | Toffoli depth | Full depth |
| :---: | :---: | :---: | :---: | :---: |
| CMMP [2] | 435 | 448 | 28 | 195 |
| J++ [13] | 390 | 189 | 1 | 28 |

※: The multiplication size $n$ is 8 .

After obtaining the exponentiation values, matrix-vector multiplication between the exponentiation and the matrix is computed by applying the XZLBZ methods because it involves the product of classical and quantum values. Multiplying vector would have originally required applying 8 CNOT gates, but by taking advantage of the fact that the given vector is a constant, resources are saved by applying X gates only to the positions where inversion is necessary.

### 3.2 Implementation of Diffusion Layer

ARIA's diffusion function $A: G F\left(2^{8}\right)^{16} \rightarrow G F\left(2^{8}\right)^{16}$ is expressed as a $16 \times 16$ binary matrix multiplication. Since one element of the binary matrix is a byte, in order to multiply with the input bit, the byte must be converted to a bit unit and the calculation proceeded. To do so, the calculation proceeds assuming that the element 0 in the matrix represents an $8 \times 8$ zero matrix, and the element 1 in the matrix represents an $8 \times 8$ identity matrix. For implementing matrix-vector multiplication in quantum, we can use linear layer optimization methods (i.e. PLU Factorization, Gauss-Jordan elimination etc.) [2] employed PLU factorization. In contrast, we applied XZLBZ [24] to optimize the implementation of the linear layer for increased efficiency. Table 2 compares the quantum resources used in the implementation of the Diffusion layer between [2] and our approach. In the case of [2], since 96 CNOT operations are required per byte, a total of $768(=96 \times 8)$ CNOT gates are used. In contrast, for XZLBZ, since 47 CNOT operations are required per byte, $376(=47 \times 8)$ CNOT gates are used in total. Consequently, Table 2 demonstrates a reduction of $51.04 \%$ and $45.16 \%$ in CNOT gates and depths, respectively, while maintaining the same number of qubits.

Table 2: Quantum resources required for Diffusion layer.

| Source | \#CNOT | qubit | Depth |
| :---: | :---: | :---: | :---: |
| CMMP [2] | 768 | 128 | 31 |
| XZLBZ [24] | 376 | 128 | 17 |

### 3.3 Implementation of Key Schedule

In the key initialization phase, the 128 -qubit $W_{1}, W_{2}$, and $W_{3}$ are generated using round functions. Since $K_{L}$ is used only for the generation of $W_{0}$, instead of allocating new qubits for $W_{0}, K_{L}$ is utilized as a substitute, resulting in a reduction of 128 qubits. In addition, when performing the XOR operation of KR and $W_{1 \sim 3}$, since KR is a constant, the X gate are applied to W 1 only when the bit of KR is 1 . By replacing the CNOT gates with cheaper X gates, the number of gates and gate cost are reduced. In contrast, our implementation in the key initialization stage employs 192 X gates and 87544 CNOT gates, leading
to a reduction of approximately $49 \%$ in X gates and about $45 \%$ in CNOT gates compared to [2].

In the key generation stage, a round key $e k$ used as an encryption key for each round is generated using $W_{0 \sim 3}$. If $W_{0}$ is used in the generation of $e k$, we reduce the gate cost by applying the X gates instead of the CNOT gates as in the generation of $W_{0}$.

Since the value of $e k$ is different for each round, new qubits must be allocated and stored each time. However, instead of allocating new qubits for ek every round, we initialize and reuse the qubits by performing a reverse operation on the round key generation at the end of every round. Since the reverse operation on key generation, which is related to CNOT gates and X gates, has little effect on the depth, it is more efficient to perform the reverse operation than to allocate 128 ancilla qubits every round.

```
Algorithm 1: Quantum circuit implementation of key schedule for ARIA.
Input: master key \(M K\), key length \(l\), vector \(a, b\), ancilla qubit anc, round number \(r\)
Output: round key \(e k\)
    \(W_{1} \leftarrow F_{o}(M K[: 128], a, b, a n c) \quad \triangleright M K[: 128]\) is \(K L\)
    Constant_XOR \(\left(W_{1}[l-128: 128], M K[l-128: l]\right) \triangleright M K[l-128: l]\) is \(K R\)
    \(W_{2} \leftarrow F_{e}\left(W_{1}, a, b, a n c\right)\)
    \(W_{2} \leftarrow \operatorname{CNOT128}\left(M K[: 128], W_{2}\right)\)
    \(W_{3} \leftarrow F_{o}\left(W_{2}, a, b, a n c\right)\)
    \(W_{3} \leftarrow \operatorname{CNOT} 128\left(W_{1}, W_{3}\right)\)
    num \(=[19,31,67,97,109] \quad \triangleright\) Key Generation
    for \(i \leftarrow 0\) to \(r\) do
        if \(i=0(\bmod 4)\) then
            Constant_XOR ( \(e k, M K[: 128])\)
        else
            \(e k \leftarrow \operatorname{CNOT128}\left(W_{(i \% 4)}, e k\right)\)
        \(e k \leftarrow \operatorname{CNOT} 128\left(W_{(i+1) \% 4} \ggg n u m[i \% 4], e k\right)\)
    return \(e k\)
```


## 4 Evaluation

In this section, we estimate and analyze the quantum circuit resources for ARIA. The proposed quantum circuits cannot yet be implemented in large-scale quantum computers. Therefore, we use ProjectQ, a quantum programming tool, on a classical computer instead of real quantum computer to implement and simulate
quantum circuits. A large number of qubits can be simulated using ProjectQ's own library, ClassicalSimulator, which is restricted to simple quantum gates (such as X, SWAP, CNOT, and Toffoli). With the aid of this functionality, the ClassicalSimulator is able to test the implementation of a quantum circuit by classically computing the output for a particular input. For the estimation of quantum resources, another internal library called ResourceCounter is needed. ResourceCounter solely counts quantum gates and circuit depth, doesn't run quantum circuits, in contrast to ClassicalSimulator.

### 4.1 Performance of the Proposed Quantum Circuit

Table 3 and 4 represent the quantum resources required to implement our proposed quantum circuits for ARIA. These tables compare the quantum resources between the quantum circuit proposed by Chauhan et al. [2] and our proposed quantum circuit. Table 3 shows quantum resources for ARIA at the NCT (NOT, CNOT, Toffoli) gate level, while Table 4 presents quantum resources for ARIA at the Clifford+T level, achieved by decomposing the Toffoli gate. In [2], the decomposed quantum resources were not explicitly provided, so the quantum resources in Table 4 are extrapolated based on the information provided in the paper [2]. Furthermore, our implementation places a primary emphasis on circuit depth optimization while carefully considering the balance with qubit utilization. We conduct assessments that encompass circuit complexity metrics, such as $T D-M$ cost and $F D-M$ cost, where $T D-M$ cost represents the multiplication of Toffoli depth ( $T D$ ) and the number of qubits $(M)$, while $F D-M$ cost signifies the multiplication of Full depth $(F D)$ and the number of qubits $(M)$.

Table 3: Required quantum resources for ARIA quantum circuit implementation

| Cipher | Source | \#X | \#CNOT | \#Toffoli | Toffoli depth | \#Qubit | Depth | $T D-M$ cost |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| ARIA-128 | CS [2] | 1,595 | 231,124 | 157,696 | 4,312 | 1,560 | 9,260 | $6,726,720$ |
|  | This work | 1,408 | 272,392 | 25,920 | 60 | 29,216 | 3,091 | $1,752,960$ |
| ARIA-192 | CS [2] | 1,851 | 273,264 | 183,368 | 5,096 | 1,560 | 10,948 | $7,949,760$ |
|  | This work | 1,624 | 315,144 | 29,376 | 68 | 32,928 | 3,776 | $2,239,104$ |
| ARIA-256 | CS [2] | 2,171 | 325,352 | 222,208 | 6,076 | 1,688 | 13,054 | $10,256,288$ |
|  | This work | 1,856 | 352,408 | 32,832 | 76 | 36,640 | 4,229 | $2,784,640$ |

### 4.2 Evaluation of Grover's Search Complexity

In this section, we evaluate the quantum security of ARIA by estimating the cost of Grover's key search for this algorithm. As described in Section 2.3, the overhead of the diffusion operator can be considered insignificant compared to the overhead of the oracle, so it is disregarded when estimating the cost of the

Table 4: Required decomposed quantum resources for ARIA quantum circuit implementation

| Cipher | Source | $\#$ Cliford | $\# T$ | $T$-depth | $\#$ Qubit | Full depth |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| ARIA-128 | CS $[2]^{\diamond}$ | $1,494,287$ | $1,103,872$ | 17,248 | 1,560 | 37,882 |
|  | This work | 481,160 | 181,440 | 240 | 29,216 | 4,241 |
| ARIA-192 | CS $[2]^{\diamond}$ | $1,742,059$ | $1,283,576$ | 20,376 | 1,560 | 44,774 |
|  | This work | 551,776 | 205,632 | 272 | 32,928 | 5,083 |
| ARIA-256 | CS $[2]^{\diamond}$ | $2,105,187$ | $1,555,456$ | 24,304 | 1,688 | 51,666 |
|  | This work | 616,920 | 229,824 | 304 | 36,640 | 5,693 |

$\diamond$ Extrapolated result

Grover's key search. Therefore, the optimal number of iterations for Grover's key search for a cipher using a $k$-bit key is approximately $\left\lfloor\frac{\pi}{4} \sqrt{2^{k}}\right\rfloor$.

According to [15], finding a unique key requires $r$ plaintext-ciphertext pairs, where $r$ needs to be at least $\lceil$ key size/block size $\rceil$. To calculate the quantum resources required for Grover's key search in the block cipher, the decomposed quantum resources need to be multiplied by $2, r$, and $\left\lfloor\frac{\pi}{4} \sqrt{2^{k}}\right\rfloor$.

In the case of ARIA with the key size of 192 or 256 bits, the value of $r$ is 2 , indicating that the multiplication by $r$ cannot be omitted. Therefore, the Grover's key search cost for ARIA is approximately Table $4 \times r \times 2 \times\left\lfloor\frac{\pi}{4} \sqrt{2^{k}}\right\rfloor$ (see Table 5).

Table 5: Cost of the Grover's key search for ARIA

| Cipher | Source | Total gates | Full depth | Cost <br> (complexity) | \#Qubit | $T D-M$ cost |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| ARIA-128 | CS [2] | $1.946 \cdot 2^{85}$ | $1.816 \cdot 2^{79}$ | $1.767 \cdot 2^{165}$ | 1,561 | $1.26 \cdot 2^{86}$ |
|  | This work | $1.985 \cdot 2^{83}$ | $1.626 \cdot 2^{76}$ | $1.614 \cdot 2^{160}$ | 29,217 | $1.313 \cdot 2^{84}$ |
| ARIA-192 | CS [2] | $1.133 \cdot 2^{119}$ | $1.073 \cdot 2^{113}$ | $1.216 \cdot 2^{232}$ | 3,121 | $1.489 \cdot 2^{118}$ |
|  | This work | $1.135 \cdot 2^{117}$ | $1.949 \cdot 2^{109}$ | $1.106 \cdot 2^{227}$ | 65,857 | $1.677 \cdot 2^{116}$ |
| ARIA-256 | CS [2] | $1.627 \cdot 2^{150}$ | $1.238 \cdot 2^{145}$ | $1.007 \cdot 2^{296}$ | 3,377 | $1.921 \cdot 2^{150}$ |
|  | This work | $1.268 \cdot 2^{149}$ | $1.092 \cdot 2^{142}$ | $1.385 \cdot 2^{291}$ | 73,281 | $1.043 \cdot 2^{149}$ |

Cost is an indicator that can be compared with the security criteria provided by NIST. After comparing with the quantum attack cost ( $\mathbf{2}^{\mathbf{1 5 7}}, \mathbf{2}^{\mathbf{2 2 1}}$, and $\mathbf{2}^{\mathbf{2 8 5}}$ ) described in Section 2.4, it can be confirmed that all instances of ARIA attain the suitable level of security for their respective key sizes. We conduct evaluations, including metrics such as $T D-M$ cost, where $T D-M$ cost represents the multiplication of Toffoli depth $(T D)$ and qubit $\operatorname{count}(M)$, to assess these trade-offs.

To take NIST's MAXDEPTH (mentioned in Section 2.4) into account, one cannot disregard parallelization. When comparing Full depth $(F D)$ and NIST MAXDEPTH in Table 5, only ARIA-128 meets the MAXDEPTH requirement (ARIA-128 < $2^{96}$ ). If the full depth $(F D)$ exceeds MAXDEPTH, as in the case of ARIA-192 and ARIA-256, reducing $F D$ by $F D /$ MAXDEPTH requires Grover instances to operate in parallel by a factor of $F D^{2} /$ MAXDEPTH $^{2}$. In this scenario, while MAXDEPTH can be decreased, $M$ increases by a factor of $F D^{2} /$ MAXDEPTH $^{2}$, resulting in a final value of $\left(F D^{2} /\right.$ MAXDEPTH $\left.^{2}\right) \times M$. Ultimately, $F D^{2}-M$ represents the cost of $F D-M$, considering parallelization for Grover search. Similar to $F D^{2}-M, F D^{2}-M$ also denotes the cost of $T D-M$, considering parallelization for Grover search. However, according to [12,15], parallelization of Grover's key search is highly inefficient; therefore, , instead of directly imposing a MAXDEPTH limit on the cost, the focus is on minimizing the costs of relevant metrics (e.g., $F D^{2}-M, T D^{2}-M$ ).

## 5 Conclusion

In this paper, we propose optimized quantum circuit for ARIA, focusing on circuit depth optimization. We utilize various techniques such as optimized multiplication and squaring methods in binary fields, along with parallelization, to reduce both Toffoli and full depths while ensuring a reasonable number of qubits. As a result, our quantum circuit implementation for ARIA achieves the depth improvement of over $88.8 \%$ and Toffoli depth by more than $98.7 \%$ compared to the implementation proposed in Chauhan et al.'s SPACE'20 paper [2]. Based on our quantum circuits, we estimate the quantum resources and the cost of Grover's attacks for the proposed circuit. We then evaluate the security strength based on the criteria provided by NIST. We demonstrate that ARIA achieves post-quantum security levels 1,3 , and 5 , respectively, for all key sizes: 128, 192, and 256 bits (according to the recent standards [20]). Additionally, we have shown that only ARIA-128 satisfies the MAXDEPTH limit.

Our future plan involves optimizing ARIA's quantum circuits further, with greater consideration for the MAXDEPTH limit.

## 6 Acknowledgment

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT).(No. RS-2023-00277994, Quantum Circuit Depth Optimization for ARIA, SEED, LEA, HIGHT, and LSH of KCMVP Domestic Cryptographic Algorithms, 80\%) and this work was supported by Institute of Information \& communications Technology Planning \& Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00627, Development of Lightweight BIoT technology for Highly Constrained Devices, $20 \%$ ).

## References

1. Amy, M., Maslov, D., Mosca, M., Roetteler, M., Roetteler, M.: A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 32(6) (Jun 2013) 818-830 6
2. Chauhan, A.K., Sanadhya, S.K.: Quantum resource estimates of grover's key search on aria. In: Security, Privacy, and Applied Cryptography Engineering: 10th International Conference, SPACE 2020, Kolkata, India, December 17-21, 2020, Proceedings 10, Springer (2020) 238-258 2, 9, 10, 11, 12, 13, 14, 15
3. Cheung, D., Maslov, D., Mathew, J., Pradhan, D.K.: On the design and optimization of a quantum polynomial-time attack on elliptic curve cryptography. In: Workshop on Quantum Computation, Communication, and Cryptography, Springer (2008) 96-104 10
4. Chun, M., Baksi, A., Chattopadhyay, A.: Dorcis: Depth optimized quantum implementation of substitution boxes. Cryptology ePrint Archive (2023) 9
5. Chung, D., Lee, S., Choi, D., Lee, J.: Alternative tower field construction for quantum implementation of the aes s-box. IEEE Transactions on Computers 71(10) (2021) 2553-2564 9
6. Dasu, V.A., Baksi, A., Sarkar, S., Chattopadhyay, A.: Lighter-r: optimized reversible circuit implementation for sboxes. In: 2019 32nd IEEE International System-on-Chip Conference (SOCC), IEEE (2019) 260-265 9
7. Fedorov, A., Steffen, L., Baur, M., da Silva, M.P., Wallraff, A.: Implementation of a Toffoli gate with superconducting circuits. Nature 481(7380) (2012) 170-172 6
8. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying Grover's algorithm to AES: quantum resource estimates (2015) 1
9. Itoh, T., Tsujii, S.: A fast algorithm for computing multiplicative inverses in gf (2m) using normal bases. Information and computation 78(3) (1988) 171-177 9
10. Jang, K., Baksi, A., Kim, H., Seo, H., Chattopadhyay, A.: Improved quantum analysis of speck and lowmc (full version). Cryptology ePrint Archive (2022) 1
11. Jang, K., Baksi, A., Song, G., Kim, H., Seo, H., Chattopadhyay, A.: Quantum analysis of aes. Cryptology ePrint Archive (2022) 1, 8
12. Jang, K., Kim, D., Oh, Y., Lim, S., Yang, Y., Kim, H., Seo, H.: Quantum implementation of aim: Aiming for low-depth. Cryptology ePrint Archive (2023) 15
13. Jang, K., Kim, W., Lim, S., Kang, Y., Yang, Y., Seo, H.: Optimized implementation of quantum binary field multiplication with toffoli depth one. In: International Conference on Information Security Applications, Springer (2022) 251-264 10
14. Jang, K., Song, G., Kim, H., Kwon, H., Kim, H., Seo, H.: Efficient implementation of present and gift on quantum computers. Applied Sciences 11(11) (2021) 47761
15. Jaques, S., Naehrig, M., Roetteler, M., Virdia, F.: Implementing grover oracles for quantum key search on aes and lowmc. In: Advances in Cryptology-EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II 30, Springer (2020) 280-310 1, 8, 14, 15
16. Jones, C.: Low-overhead constructions for the fault-tolerant toffoli gate. Physical Review A 87(2) (2013) 0223286
17. Kwon, D., Kim, J., Park, S., Sung, S.H., Sohn, Y., Song, J.H., Yeom, Y., Yoon, E.J., Lee, S., Lee, J., et al.: New block cipher: Aria. In: Information Security and Cryptology-ICISC 2003: 6th International Conference, Seoul, Korea, November 27-28, 2003. Revised Papers 6, Springer (2004) 432-445 3
18. Nielsen, M.A., Chuang, I.: Quantum computation and quantum information (2002) 6
19. NIST.: Submission requirements and evaluation criteria for the postquantum cryptography standardization process (2016) https://csrc. nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/ call-for-proposals-final-dec-2016.pdf. 8
20. NIST.: Call for additional digital signature schemes for the post-quantum cryptography standardization process (2022) https://csrc.nist.gov/csrc/media/ Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022. pdf. 8, 15
21. Ralph, T., Resch, K., Gilchrist, A.: Efficient Toffoli gates using qudits. Physical Review A 75(2) (2007) 0223136
22. Roy, S., Baksi, A., Chattopadhyay, A.: Quantum implementation of ascon linear layer. Cryptology ePrint Archive (2023) 1
23. Selinger, P.: Quantum circuits of t-depth one. Physical Review A 87(4) (2013) 0423026
24. Xiang, Z., Zeng, X., Lin, D., Bao, Z., Zhang, S.: Optimizing implementations of linear layers. IACR Transactions on Symmetric Cryptology (2020) 120-145 9, 10, 11
25. Yang, Y., Jang, K., Kim, H., Song, G., Seo, H.: Grover on sparkle. In: International Conference on Information Security Applications, Springer (2022) 44-59 1
26. Yongjin, J., Baek, S., Kim, J.: A novel framework to construct quantum circuits of s-boxes: Applications to 4-bit s-boxes. (2023) 9

# Finding Shortest Vector using Quantum NV Sieve on Grover 

Hyunji Kim ${ }^{1}$, Kyoungbae Jang ${ }^{1}$, Yujin Oh ${ }^{1}$, Woojin Seok ${ }^{2}$, Wonhuck Lee ${ }^{2}$, Kwangil $\mathrm{Bae}^{2}$, Ilkwon Sohn ${ }^{2}$, and Hwajeong Seo ${ }^{1}$<br>${ }^{1}$ IT Department, Hansung University<br>${ }^{2}$ Korea Institute of Science and Technology Information (KISTI)<br>\{khj1594012,starj1023, oyj0922\}@gmail.com,\{wjseok, livezone, d2estiny, kibae\}@kisti.re.kr, hwajeong84@gmail.com


#### Abstract

Quantum computers, especially those with over 10,000 qubits, pose a potential threat to current public key cryptography systems like RSA and ECC due to Shor's algorithms. Grover's search algorithm is another quantum algorithm that could significantly impact current cryptography, offering a quantum advantage in searching unsorted data. Therefore, with the advancement of quantum computers, it is crucial to analyze potential quantum threats. While many works focus on Grover's attacks in symmetric key cryptography, there has been no research on the practical implementation of the quantum approach for lattice-based cryptography. Currently, only theoretical analyses involve the application of Grover's search to various Sieve algorithms. In this work, for the first time, we present a quantum NV Sieve implementation to solve SVP, posing a threat to lattice-based cryptography. Additionally, we implement the extended version of the quantum NV Sieve (i.e., the dimension and rank of the lattice vector). Our extended implementation could be instrumental in extending the upper limit of SVP (currently, determining the upper limit of SVP is a vital factor). Lastly, we estimate the quantum resources required for each specific implementation and the application of Grover's search. In conclusion, our research lays the groundwork for the quantum NV Sieve to challenge lattice-based cryptography. In the future, we aim to conduct various experiments concerning the extended implementation and Grover's search.


Keywords: Shortest Vector Problem • Lattice based cryptography . Quantum NV Sieve • Quantum attack • Grover's search.

## 1 Introduction

As outlined in IBM's roadmap ${ }^{1}$, if a stable quantum computer with more than 10,000 qubits is developed, public key algorithms (such as Rivest, Shamir, Adle-

[^2]man (RSA) and Elliptic curve cryptography (ECC)) may be decrypted within polynomial time through Shor algorithm [1].

Additionally, If a search count of $O\left(2^{k}\right)$ on a classical computer is required, Grover's algorithm can find results with a maximum of $O\left(\sqrt{2^{n}}\right)$ searches.

As quantum computers developed, the current cryptography system is under threat. Therefore, migration to a secure cryptography system and analysis of potential quantum attacks are very important issues.

Among the categories of post-quantum cryptography, there are lattice-based ciphers (e.g. LWE (Learning with Error)). Currently, much research has been conducted on estimating the cost of Grover attacks on symmetric key cryptography $[2,3,4,5,6]$.

However, research on practical quantum attacks on lattice-based cryptography is lacking. As mentioned earlier, to establish a secure post-quantum security system, it is crucial to analyze potential quantum attacks on various cryptographic methods. Therefore, in this paper, we propose a quantum implementation for NV Sieve that can solve SVP (Shortest Vector Problem) for latticebased cryptography. In addition, we present an implementation considering the dimension and rank expansion of the lattice and estimate the quantum cost for an attack through quantum NV Sieve.

### 1.1 Our Contributions

1. For the first time in our knowledge, Quantum NV Sieve implementation to solve SVP
There is theoretical research that applies Grover's search to Sieve algorithms to solve SVP [7]. However, as far as we know, there is no implementation for these yet. In this work, we implement NV Sieve, an attack that can threaten lattice-based cryptosystems by solving SVP, as a quantum circuit. Through this, an oracle that can be applied to Grover's search is created.
2. Extension implementation considering multiple conditions (dimension, rank) of lattice-based cryptography
In addition to the basic NV Sieve implementation, we present an extended implementation that takes into account the dimension and rank of the lattice. Our extended implementation can help raise the SVP upper limit that NV Sieve can solve.
3. Resource estimation for Quantum NV Sieve logic and Grover's search
Grover's search algorithm has an advantage that can compute all possibilities at once. By applying Grover's search to NV Sieve, a solution that satisfies the condition can be found with quantum advantage. This approach requires an oracle, and our implementation can be used as an oracle for Grover's search. In this work, we estimate the quantum cost for each case-specific implementation.
Based on our quantum circuits, we estimate the required quantum resources for Grover's search (on NV Sieve). This is affected by quantum resources
and the number of iterations. We get the appropriate iteration of Grover's search and also get the quantum cost of Grover's search attack. ${ }^{2}$

### 1.2 Organization of the paper

The remainder of this paper is organized as follows. In Section 2, classical NV Sieve, SVP (Shortest Vector Problem), and background for quantum implementation are described. In Section 3, the implementation of the quantum NV Sieve is proposed. Section 4 demonstrates the results of the experiment and further discussion about that. Finally, Section 5 concludes the paper.

## 2 Prerequisites

### 2.1 Lattice

Lattice Lattice $(L)$ is a set of points made up of a linear combination of basis vectors $(B)$. Since it is made up of points, there can be more than one shortest vector (e.g. $x,-x \in L$ ). Equation 1 represent a lattice, and $x$ is an integer in Equation 1, and $\left(b_{1}, \ldots, b_{n}\right)$ means the basis vector.

$$
\begin{equation*}
L\left(b_{1}, \ldots, b_{n}\right)=\sum_{i=1}^{n}\left(x_{i} \cdot b_{i}, x_{i} \in Z\right) \tag{1}
\end{equation*}
$$

Basis As noted earlier, the lattice is based on basis vectors. A basis vector $(B)$ is a set of vectors that can constitute all lattice points. The vector (arrow sign) in Figure 1 represents the basis in the lattice. Each vector $\left(b_{i}\right)$ constituting the basis vector has a length of $m$ and consists of a total of $n$ components. Here, the length of each vector and the number of vectors constituting the basis vector, respectively, are called $\operatorname{Dimension}(m)$ and $\operatorname{Rank}(n)$. Generally, a full-rank lattice is used $(m=n)$.

Here, the basis vector consisting of one lattice is not unique. As shown in Figure 1, the basis vectors on a lattice with the same lattice points are different. If a lattice is created with a vector created by multiplying one basis vector by another, the two basis vectors create the same lattice.

However, these basis vectors have a good basis and a bad basis. A good basis is generally composed of a short vector, and a bad basis is created by multiplying the good basis by a matrix such as an unimodular matrix ${ }^{3}$ several times. Therefore, finding a bad basis from a good basis is easy because only matrix multiplication several times is required. However, in the opposite case, finding a good basis from a bad basis becomes a very difficult task. This can be seen as similar to generating a public key from a private key in public key cryptography. (i.e. obtaining a private key by factorizing a very large public key

[^3]

Fig. 1: Two different basis vectors generating the same lattice.
into prime factors.) Similarly, in lattice-based cryptography, a bad basis is used as the public key, and a good basis is used as the private key. Here, the good basis and the bad basis are basis vectors that generate the same lattice. Constructing the public and private keys in this way makes it difficult to decrypt messages in lattice-based encryption.

### 2.2 Shortest Vector Problem (SVP)

SVP, known as the basic problem of lattice-based cryptography, is the problem of finding the shortest vector on a lattice that is not a zero vector. Miklo's Ajtai [8] revealed that SVP is an NP-hard problem. In addition, it was later revealed that it had almost the same level of difficulty as the Closest Vector Problem (CVP), which is another lattice-based problem. SVP is a problem of finding the shortest vector by using the basis of the lattice as input. However, the solution is not always unique because one vector can have an opposite vector with the same size.

When a bad basis vector is used as input, the difficulty of solving the SVP increases. If a good basis is used as an input, there is a high possibility that the shortest vector will be included in the already input good basis. If a bad basis is used, the opposite scenario occurs. Additionally, as the rank of the lattice (the number of vectors constituting the lattice) increases, it becomes more difficult to solve.

The lattice-based cryptography is generally used when the rank is 500 or higher. Therefore, solving lattice-based cryptography is a very challenging work. Furthermore, as mentioned earlier, one can easily derive a bad basis (public key) from a good basis (private key). However, it is difficult to find a good basis (private key) from a bad basis (public key) due to information asymmetry. Thus, solving lattice-based cryptography is challenging due to its reliance on one-wayness (the computation in one direction is straightforward but difficult in the reverse direction).

In this way, lattice-based cryptography is based on lattice problems (SVP, CVP, etc.), and the security level of lattice-based cryptography is based on the difficulty of solving the lattice problem. For example, RSA's security strength is based on the difficulty of prime factorization. In other words, lattice-based cryptography is designed by utilizing one-wayness such as information asymmetry.

To solve such lattice-based cryptography, the lattice problem must be solved. Solving SVP, a representative lattice problem, lattice-based cryptosystems such as LWE can be threatened.

Algorithms to solve SVP Several algorithms, such as AKS and Sieve, have been proposed to solve the lattice problem, which underpins lattice-based cryptography. However, these algorithms generally target low-dimensional lattices with a rank of about $50 \sim 60$. There are also algorithms that target highdimensional lattices, but finding the shortest vector in a high-dimensional lattice is a very difficult problem. Therefore, there's a need for an approximate algorithm that can reduce the problem from a high-dimensional lattice to a lowdimensional one. As a result, to solve SVP, an exact algorithm that accurately finds the shortest vector in the low-dimensional lattice is needed and important.

Approximate algorithms that reduce high-dimensional to low-dimensional lattice (e.g., Lenstra, Lenstra, and Lovász (LLL) [9], block Korkine-Zolotarev (BKZ) [10]) have also been widely studied. Also, it is efficient in high-dimension lattices. However, as shown in Figure 2, the method for finding exactly short vectors belongs to the exact algorithm, and the best practical and theoretical SVP solution should be accurate and efficient in low dimensions. Therefore, for now, it is important to take an approach that accurately solves SVP in low dimensions. It is then important to determine the upper limit (highest dimension of lattice) that can be solved.


Fig. 2: Flow chart of approximate and exact algorithms for solving SVP.

### 2.3 Survey on the exact algorithms for SVP

Well-known exact algorithms include AKS [11] and NV Sieve [12]. AKS is the most famous early exact algorithm, but it has the disadvantage of using many parameters and having high time and space complexity. Moreover, due to the absence of optimal parameters, actual implementation, or analysis, it is deemed an impractical algorithm. Subsequently, NV Sieve, an exact algorithm, was introduced to address these limitations of AKS. It offers benefits such as reduced time and space complexity, practicality, and the possibility for actual implementation and evaluation. Additionally, building upon the NV Sieve algorithm,
several Sieve algorithms, including the List Sieve and Gaussian Sieve, have been presented [13,14,15, 16, 17].

However, only the theoretical complexity of the Sieve algorithm on quantum computers (using Grover's search) has been calculated [7]. There is no practical implementation or analysis for this.

### 2.4 NV Sieve algorithm

Reasons and overview for selecting the NV Sieve algorithm NV Sieve is more practical and efficient than AKS and serves as the foundation for numerous Sieve algorithms. So, in our work, NV Sieve is selected as an exact algorithm for solving the SVP problem. Although there are algorithms with lower time and space complexity than NV Sieve, quantum computing can incur significant costs when implementing algorithms that require additional procedures. Of course, a simple algorithm is not necessarily efficient when executed on a quantum computer.

```
Algorithm 1: NV Sieve algorithm for finding short lattice vectors
Input: An reduced basis \((B)\) in lattice \((L)\) using the LLL algorithm, a sieve factor \(\gamma\)
\(\left(\frac{2}{3}<\gamma<1\right), S\) is an empty set, and a number \(N\)
Output: A non-zero short vector of \(L\)
    for \(i=1\) to \(N\) do
        \(S \leftarrow\) Sampling \(B\) using sampling algorithm
    end for
    Remove all zero vectors from \(S\).
    \(S_{0} \leftarrow S\)
    Repeat
        \(S_{0} \leftarrow S\)
        \(S \leftarrow\) latticesieve \((S, \gamma R)\) using Algorithm 2.
        Remove all zero vectors from \(S\).
    until \(S\) becomes an empty set.
    Return \(v_{0} \in S_{0}\) such that \(\left\|v_{0}\right\|=\min \|v\|, v \in S_{0}\)
```

Details of NV Sieve algorithm Algorithm 1 briefly shows the main process of NV Sieve. The goal of NV Sieve is to find the shortest vector excluding zero vectors while losing as few vectors as possible. The input is the basis vector of the lattice reduced through the approximate algorithm (i.e., LLL), and the output is the shortest vector, not the zero vector. As mentioned earlier, the shortest vector may not be one. In addition, $\gamma R$, the sieve factor, is a geometric element in the range of $\frac{2}{3}<\gamma R<1$, and the closer it is to 1 , the better. The reduction range of the lattice, which will be explained later, is determined by the corresponding sieve factor.

The overall structure is as follows. First, a set $S$ is generated by randomly sampling from the basis received as input. Next, the zero vector is removed from $S$ to generate $S_{0}$, and then the latticesieve is repeatedly performed with $S$ and $\gamma$ as input. After this, the output vectors with zero vectors removed are stored in $S_{0}$, and the process is repeated until $S$ becomes an empty set. Finally, it is completed by returning the shortest vector among the vectors belonging to $S_{0}$.

```
Algorithm 2: The latticesieve algorithm in NV Sieve
Input: A subset \(S\) in \(L\) and sieve factor \(\gamma(0.666<\gamma<1)\)
Output: \(S^{\prime}\) (Short enough vector, not zero vector)
    Initialize \(C, S^{\prime}\) to empty set.
    \(R \leftarrow \max _{v \in S}\|v\|\)
    for \(v \in S\) do
        if \(\|v\| \leq \gamma R\) then
            \(S^{\prime} \leftarrow S^{\prime} \cup\{v\}\)
        else
            if \(\exists c \in C\|v-c\| \leq \gamma R\) then
                \(S^{\prime} \leftarrow S^{\prime} \cup\{v-c\}\)
            else
                    \(C \leftarrow C \cup\{v\}\)
            end if
        end if
    end for
    return \(S^{\prime}\)
```



Fig. 3: The core logic in NV Sieve (See line 7 in Algorithm 2).

Algorithm 2 shows the lattice sieve algorithm in NV Sieve and shows the detailed process. This sieve algorithm is the core logic of NV Sieve, and its purpose is as follows.

- In order to minimize the loss for short vectors, a point on the lattice called $c$ is randomly selected. $c$ is a sufficient number of points on the lattice belonging to $\gamma R<x<R$ and belongs to the yellow area in Figure 3.
- The search range $(\gamma R)$ is reduced by the sieve factor $\gamma$ to obtain a vector shorter. Here, $R$ means the maximum length among the vectors belonging to the vector set received as input.

The core logic of the NV sieve mentioned earlier in more detail is as follows.

1. First, initialize $C$ and $S^{\prime}$. Afterward, vectors with a length shorter than $\gamma R$ are stored in $S^{\prime}$. ( $S^{\prime}$ is used to store vectors within the $\gamma R$ range.)
2. However, there will be vectors longer than $\gamma R$. For this, the process as in line 7 is performed to minimize loss for short vectors on the lattice, which is the goal of NV Sieve.
A vector longer than $\gamma R$ is subtracted from a point on the lattice called $c$. If the result is shorter than $\gamma R$, then it is stored in $S^{\prime}$. If the length is longer than $\gamma R$, it is stored in $C$. In other words, when the vector after subtraction starts from $O$ (origin point), if it is within the range of $\gamma R$, it is stored in $S^{\prime}$.
3. Finally, by returning $S^{\prime}$, vectors with a length shorter than $\gamma R$ are selected. By performing this process repeatedly, sufficiently short vectors are obtained, and the shortest vector among them is found.

Important factors related to the complexity The parts that affect the complexity of NV Sieve's algorithm are as follows. The first part is measuring the number of points in $c$. There are a sufficiently large number of points on the lattice, and we need to find a point $c$ that can be used to create a vector with a length shorter than $\gamma R$. Therefore, it is important to find the number of $c$. Next, as the size of the initially given vector set $S$ increases, complexity increases. As the rank of $S$ increases, the number of $c$ also increases because $c$ is also a vector on the lattice and a subset of $S$. This is related to the complexity related to the number of $c$ mentioned above. Additionally, as mentioned earlier, lattice problems with large ranks are difficult to solve, so the size of the target basis vector set affects the complexity of the algorithm.

### 2.5 Grover's search algorithm

Grover's search algorithm is a quantum search algorithm for tasks with $n$-bit complexity and has $O\left(\sqrt{2^{n}}\right)$ of complexity $\left(O\left(2^{n}\right)\right.$ for classical). The data ( $n$ bit) for the target of the search must exist in a state of quantum superposition, so given by:

$$
\begin{equation*}
H^{\otimes n}|0\rangle^{\otimes n}(|\psi\rangle)=\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)=\frac{1}{2^{n / 2}} \sum_{x=0}^{2^{n}-1}|x\rangle \tag{2}
\end{equation*}
$$

Thanks to quantum advantage, all search targets are computed simultaneously as a probability.

Grover's algorithm consists of two modules: Oracle and Diffusion operator. Oracle is a quantum circuit that implements logic that can return a solution to the problem to be solved. Then it returns a solution by inverting the decision qubit at the end of the circuit as follows.

$$
f(x)=\left\{\begin{array}{l}
1 \text { if } \text { Oracle }_{\psi(n)}=\text { Solution }  \tag{3}\\
0 \text { if } \text { Oracle }_{\psi(n)} \neq \text { Solution }
\end{array}\right.
$$

Afterwards, the probability of the returned solution is amplified through the diffusion operator. By repeating this process, the probability of observing the correct solution is increased. The number of such repetitions is expressed as Grover iteration. The most important thing in Grover's search is the optimal implementation of the quantum circuit that designs the oracle.

The diffusion operator has a fixed implementation method and is often excluded from resource estimation $[5,18]$ because the overhead is so small that it is negligible. Therefore, the final efficiency is determined depending on the quantum circuit in the oracle.

### 2.6 Quantum Circuit

Qubits A qubit (quantum bit) is the basic unit of computation in a quantum computer and can have probabilities of 0 and 1 at the same time (superposition). So, $2^{n}$ states can be expressed with $n$ qubits. Additionally, qubits exist in a superposition state and are calculated, but are determined as a single classical value the moment they are measured. In quantum computing, classical bits are used to store the results of measuring the state of the qubit.

Quantum Gates Quantum gates operate as logical gates in quantum circuits. By applying a quantum gate to a qubit, the state of the qubit can be controlled. There are several quantum gates (see Figure 4). Each gate can be used to configure superposition, entanglement, invert, and copy, and can be utilized to perform various operations such as addition and multiplication.

## 3 Quantum NV Sieve for solving SVP

### 3.1 System Overview

According to the results of theoretical calculations, Quantum NV Sieve with Grover's search is expected to have less time complexity than classical NV Sieve ( $l o g_{2}^{0.415}$ to $l o g_{2}^{0.312}$ ) [7]. However, no implementation is presented. To the best of our knowledge, our work presents the first implementation of various cases of the NV Sieve algorithm for solving SVP using quantum circuits. However, given the current state of quantum computers in the Noisy Intermediate-Scale Quantum

10 Authors Suppressed Due to Excessive Length
|0) $\mathrm{H}-\frac{|0\rangle+|1\rangle}{\sqrt{2}}$
|1

$\frac{|0\rangle-|1\rangle}{\sqrt{2}}$


CNOT / CX gate

Hadamard gate (H)


Toffoli / CCX gate


Measure gate

Fig. 4: Quantum gates.
(NISQ) era and the challenges encountered during implementation, achieving results akin to the theoretical complexity remains challenging. Starting with our work, we plan to further improve our approach, which we remain for our future work.

As noted earlier, solving SVP, the fundamental problem of lattice-based encryption, can threaten grid-based encryption systems (e.g., LWE). Furthermore, among several algorithms, the Exact algorithm, that accurately finds short vectors, is an important part of the process of solving lattice problems.


Fig. 5: Overview of Quantum NV Sieve.

We implemented the NV Sieve algorithm, which solves the SVP problem among several lattice problems (e.g., SVP, CVP, etc.), on a quantum computer.

Figure 5 shows the overview of the quantum NV Sieve algorithm. In other words, this is the overall relationship between the quantum NV Sieve we present and the configuration diagram for solving SVP on a quantum computer.

First, since Grover's search must be applied, the logic of the NV Sieve (oracle) must be implemented using the quantum circuit. In other words, since the purpose of NV Sieve is to find the short vector $c$ that satisfies the condition ( $\|v-c\| \leq \gamma R$ ), the NV Sieve logic for searching $c$ must be implemented as an oracle. Then, Grover's search algorithm should be performed on the implemented oracle. The factors that determine the performance of NV Sieve in classical are finding the number of large numbers of $c$ and the corresponding computational and memory complexity. However, when using quantum NV Sieve, it is possible to calculate numerous cases for $c$ at once. Therefore, there are advantages in terms of computational and memory complexity.

Meanwhile, $v$, which is not the search target but is a vector on the lattice, needs to be loaded from quantum memory. However, it is difficult to access actual QRAM (Quantum RAM). In addition, many studies, such as [19], are conducted on the premise that queries can be made to QRAM. Therefore, in this implementation, QRAM is implemented as a very simple quantum circuit (Explicit QRAM: data is written directly to the quantum circuit, and the value is loaded from the corresponding memory qubit).

### 3.2 Implementation of NV Sieve on Quantum Circuit

We implement/design the quantum circuit for line 7 in Algorithm 2. It operates classically except where quantum NV sieve algorithms are used. In other words, quantum is applied to operations on a sufficiently large number of $c$. In a classical computer, we need to know how many $c$ there are and perform a size comparison on all $c$. However, in the implementation of the quantum NV sieve, a size comparison is performed on all cases of $c$ at once. Details are described in Algorithm 3.

The overall steps in Algorithm 3 are as follows:

1. Data load from explicit QRAM (line 3): It is difficult to actually access QRAM. Therefore, we implement a simple explicit QRAM on a quantum circuit. This is actually close to QROM (Quantum Read-only Memory) because it can only read data to be used.
2. Prepare $c$ in superposition state (line 4~5): Apply the Hadamard gate to $c$, Grover's search target, and prepare it in a superposition state. Since $v$ is not a search target, it doesn't make it a superposition state.
3. Prepare $\left(s q r_{-} r R\right)^{2}$ (line 6): Prepare the squared $\gamma R$.
4. Overflow handling (line 8~15): To handle overflow that occurs during the calculation process, the highest bit of the data qubit is copied to the highest qubit. Through this, data expressed in 2-qubits is made to have the same value even when converted to 3 -qubits.
5. Complement function for signed vector (line 17~18): For data involving signed vectors, the complement operation is utilized to repurpose
```
Algorithm 3: The quantum NV Sieve on the quantum circuit.
Input: Quantum circuit \((Q N V)\), A subset \(S\) in \(L\) and sieve factor \(\gamma\left(\frac{2}{3}<\gamma<1\right)\)
Output: \(c_{0}, c_{1}\)
    Initiate quantum registers and classical registers. \(\triangleright\) carry, qflag, sqr_result, etc.
    // Input setting (Each vector is allocated 3 qubits to address the overflow)
    \(v_{0}, v_{1} \leftarrow\) Data load from memory qubits
    QNV.Hadamard (cco)
    QNV.Hadamard \(\left(c_{1}\right)\)
    \(Q N V . x\left(s q r \_r R[i]\right) \quad \triangleright 0 \leq i<6\)
    7: // To address the overflow of target qubits
    8: vflag \([0] \leftarrow Q N V . c x\left(v_{0}[1], v f l a g[0]\right)\)
    \(9: v_{0}[2] \leftarrow Q N V . c x\left(v f l a g[0], v_{0}[2]\right)\)
10: cflag \([0] \leftarrow Q N V . c x\left(c_{0}[1]\right.\), cflag \(\left.[0]\right)\)
    \(: c_{0}[2] \leftarrow Q N V . c x\left(c f l a g[0], c_{0}[2]\right)\)
12: \(v f l a g[1] \leftarrow Q N V . c x\left(v_{1}[1], v f l a g[1]\right)\)
13: \(v_{1}[2] \leftarrow Q N V . c x\left(v f l a g[1], v_{1}[2]\right)\)
14: cflag[1] \(\leftarrow Q N V . c x\left(c_{1}[1]\right.\), cflag[1])
15: \(c_{1}[2] \leftarrow Q N V . c x\left(c f l a g[1], c_{1}[2]\right)\)
16: // Two's complement for subtraction using adder
17: \(c_{0} \leftarrow\) Two's complement \(\left(Q N V, c_{0}, q f l a g 0\right.\), zero \()\)
18: \(c_{1} \leftarrow\) Two's complement \(\left(Q N V, c_{1}\right.\), qflag1, zero \()\)
19: //v+ \(\bar{c}\)
20: \(c_{0} \leftarrow \operatorname{Addition}\left(Q N V, v_{0}, c_{0}\right.\), carry \()\)
21: \(c_{1} \leftarrow \operatorname{Addition}\left(Q N V, v_{1}, c_{1}\right.\), carry \()\)
22: // Two's complement for correct squaring
23: \(c_{0} \leftarrow\) Two's complement_negative \(\left(Q N V, c_{0}, q f l a g 2\right.\), carry, zero)
24: \(c_{1} \leftarrow\) Two's complement_negative \(\left(Q N V, c_{1}, q f l a g 3\right.\), carry, zero)
25: // Duplicating qubit for squaring
26: dup_c \(c_{0} \leftarrow Q N V . c x\left(c_{0}, d u p_{-} c_{0}\right)\)
27: dup_c \(c_{1} \leftarrow Q N V . c x\left(c_{1}, d u p_{-} c_{1}\right)\)
28: // Squaring elements of vectors
29: sqr_result \([2] \leftarrow \operatorname{Squaring}\left(Q N V, c_{0}\right.\), dup_c \(c_{0}\), sqr_result \([0]\), sqr_result \([1]\), sqr_result \([2]\), carry, 6\()\)
30: sqr_result \([5] \leftarrow \operatorname{Squaring}\left(Q N V, c_{1}\right.\), dup_c \(c_{1}\), sqr_result \([3]\), sqr_result[4], sqr_result \([5]\), carry, 6\()\)
31: // Addition for squared results to calculate the size of the vector
32: sqr_result \([5] \leftarrow \operatorname{Addition}(Q N V\), sqr_result[2], sqr_result[5], carry, 6)
// Two's complement for subtraction using adder
sqr_result \([5] \leftarrow\) Two's complement_6bit(QNV, sqr_result[5], qflag4, carry, zero, zero1, zero2)
// Size comparison between \((r R)^{2}\) and \((\|v-c\|)^{2} \quad \triangleright\left((r R)^{2}-(\|v-c\|)^{2}\right)\)
sqr_result \([5] \leftarrow\) Addition \((Q N V\), sqr_rR, sqr_result \([5]\), carry, 6\() \quad \triangleright\) No square root
return \(c_{0}, c_{1}\)
```

the adder as a subtractor. When comparing vector magnitudes at the conclusion of the quantum circuit, the complement operation is currently applied solely to positive vectors.
6. Three-qubit addition (line 20~21): For vector elements $(v+\bar{c})$, a 3 -qubit ripple carry adder is applied between $v$ and the complements of $c$.
7. Apply complement function for 3 -qubits to ensure correct square value (line $\mathbf{2 3} \sim \mathbf{2 4}$ ): In the complement system, $11_{2}$ is -1 , but if the complement operation for negative numbers is not performed before the square operation, $11_{2}$ is recognized as 3 . Then, the result is 9 . Therefore the complement operation must be applied for the correct result of squaring.
8. Duplicate the target qubits for squaring (line 26~27): In a quantum circuit, performing calculations on identical qubits is not feasible; therefore, the value must be copied to a different qubit.
9. Squaring each element to calculate the size of the vector (line $\mathbf{2 9 \sim 3 0}$ ): The size of a vector is the root of the sum of the squares of each element. However, in our oracle only size comparison between $\gamma R$ and $\|v-c\|$ is required. Therefore, the root operation is removed in our approach. So we only need the squaring operation of the vector at this stage.
10. 6-qubit addition of each element of the vector after squaring (line 32): To calculate the size of a vector, a square operation is required for each element.
11. 6-qubit complement for positive values (line 34): The value after squaring is naturally a positive value. However, as in the previous part of the quantum circuit, we perform a complement operation to use the adder as a subtractor. Here, since it is the value after squaring a 3 -qubit vector, a complement operation on 6 -qubits must be performed.
12. Size comparison through 6-qubit addition for $(\gamma R)^{2}$ and ( $\overline{\|v-c\|^{2}}$ ) (line 36): As mentioned earlier, the size of the vector can be obtained by performing the root operation. However, in our method, since the only purpose is size comparison, the root operation is not performed.
13. Check the MSB (Most Significant Bit): We have to check the MSB of the result value performed in step 13. If MSB is $0,(\gamma R)^{2}$ is larger than $\overline{\|v-c\|^{2}}$. Therefore, MSB of 0 means that $\|v-c\|^{2}$ is a short vector that falls within the range of the condition. Therefore, we can add vector $v-c$ to the list that stores short vectors (Classical).
Conversely, if MSB is 1 , it means that it is a negative sign, which means that it is a vector that does not satisfy the condition. Therefore, it is not added to the short vector list.

## Implementation details for core functions in Quantum NV Sieve

- Data load and input Setting $\left(v, c,(\gamma R)^{2}\right)$ : In our implementation, we use a simple QRAM structure. After allocating a memory qubit for value storage, the values are stored in the corresponding memory qubit. Afterward, the cx gate is used to read the values stored in the memory qubit, and the values are loaded into the input vector $v$. In other words, it is copying values
from quantum memory to input qubits for the oracle. Additionally, Grover's search is repeated for each $v$, and $v$ is not a search target, so it is not prepared in a superposition state.
Figure 6 shows the input setting process for $c$, the search target. What must be found through Grover's search algorithm is the $c$ value that satisfies the condition, and the $v-c$ vector at that time must be returned. Therefore, the Hadamard gate is applied to all qubits for $c$, generating a superposition state with the same probability of 0 and 1 .
Next, a process is needed to set $(\gamma R)^{2}$ required for the conditional expression. This applies the x gate to the qubit to express 1 (the same as setting the $v$ value). However, the $\gamma R$ is determined in each iteration. So, in our implementation, its squaring value is calculated in a classical method and then set as input. Therefore, since it is a square value for 2 qubit data, 4 qubits are allocated.


Fig. 6: Preparation $c\left(c_{0}\right.$ and $\left.c_{1}\right)$ and $(\gamma R)^{2}\left(s q r \_r R\right)$.

- Overflow handling: In this work, we will cover cases where overflow may occur during the NV Sieve calculation process. When the dimension is 2, there are cases where 2-qubits are exceeded during the calculation. Therefore, the calculation of NV Sieve is performed by upscaling to 3 -qubit. Figure 7 shows the quantum circuit for the overflow handling process. For example, if the dimension is 2 , data can be represented by two qubits. Therefore, the value of the second qubit (with an index of qubit is 1 ) is copied to the qflag. Afterward, upscaling is completed by copying the qflag to the highest qubit that is set to 0 . Through this process, the value expressed through 3 qubits can be expressed equally with 2 qubits.
- Two's Complement (2-qubit, 4-qubit, positive and negative cases): Figure 8 shows the quantum circuit for 2's complement for positive values. As mentioned earlier, an additional qubit (ancilla qubit, qflag) is needed as a control qubit. When the target qubit to which the complement will be applied is $c$, the MSB is $c[1]$ (lowest qubit). Therefore, the value of the lower qubit is copied to the control qubit through the cx gate. Here, bit inversion and addition of LSB and 1 must be performed only when the value


Fig. 7: Upscaling quantum circuit to handle the overflow.
is positive. However, if MSB is zero, the value of the control qubit is 0 , so the value of the control qubit must be inverted. But, after applying the x gate to qflag, the value of the control qubit is 1 , so complement logic is performed. After bit inversion, to add the value of 1 to the LSB, create a new qubit array, input qflag as the lowest bit first, and then append the value of 0 . Afterwards, addition is performed through a 2 qubit adder between the 1's complement ( 2 qubits) and the new qubit array (2 qubits).

The quantum circuit for 2's complement for the negative values is performed to calculate the correct squaring on the signed data. This uses control qubits like two's complement when positive. However, for negative numbers, the MSB itself is 1 , so there is no need to apply the x gate to qflag (omit the x gate for $q$ flag). Therefore, the bit is inverted through the cx gate without additional work. Afterward, the process for adding 1 to LSB is also performed in the same way.

The 2's complement quantum circuit for 4 qubits is similar to the 2-qubit complement quantum circuit, which is performed only when the number is positive. However, since it is 4 qubits, the MSB is $c[3]$ (Only the index of MSB is different). Therefore, after copying the value to $q f l a g$, apply the x gate to invert all bits. Afterwards, a new qubit with the state of $[0,0,0,1]$ is assigned and a 4-qubit addition is performed. Through this, 2 's complement operations on 4 qubits can be performed.


Fig. 8: Two's complement quantum circuit for a positive value (3-qubit).

- Addition: Addition is a very important and basic operation among quantum circuit operations. In this implementation, Ripple-Carry Adders such
as 3 -qubit and 4 -qubit adders are used. These are the method proposed in Cuccaro's paper [20].
- Squaring: The square operation is necessary to find the size of the vector. An integer square operation is performed on the value converted to a positive number through the 2 's complement. Figure 9 depicts the square operation. The squaring is equivalent to multiplying the same value, so $a$ and $b$ are the same value. However, operations using the same qubit repeatedly are impossible in quantum circuits. In other words, as shown in Figure 9, the value $a$ must be copied to another qubit (b) through the cx gate. Additionally, performing a multiplication on 3 qubits affects up to 6 qubits, so two 6 -qubit arrays ( $a b$ and $b a$ ) are created to store the result.

The process is as follows. First, multiply $a$ and $b$, which represent the same value, like integer multiplication. However, all elements are qubits and therefore have a value of 0 or 1 . If even one element is 0 , the result value is 0 , and only if both elements are 1 , the result value is 1 . These operations correspond to the ccx (Toffoli) gate. Therefore, the ccx gate is applied to all elements of $a$ and $b$. Afterward, the results are saved in an appropriate location. Here, the location where the calculation results are saved gray circle in each array. In addition, the results of 6 -qubit addition are stored in Second and Third array in Figure 9. However, since First, Second and Third are 6qubit arrays, the top three qubits of First are set to 0, and the remainders of Second are set to 0 . Finally, the square operation is completed by applying a 6 -qubit adder to First, Second and Third. The adder used is CDKM adder, an in-place ripple carry adder, so the final result value is stored in Third.


Fig. 9: The integer squaring for 3 -qubits.

### 3.3 Implementation for dimension expansion

Increasing dimension means that the range of bits that can be expressed by each element of the vector increases. In other words, operations on 2 qubits must be changed to operations on $n(n<2)$ qubits. In our work, we implement the case
where the dimension is increased to 4 . This indicates that our implementation is scalable in terms of dimensionality. In this case, the 3 -qubit adder must become a 5 -qubit adder, and the 3 -qubit two's complement must become a 5 -qubit two's complement. Therefore, in accordance with this increased data range, the range of functions for calculations must also be expanded.

### 3.4 Implementation for rank expansion

Even if the rank of the input vector increases, the formula for calculating the size of the vector remains the same. Therefore, it is implemented by allocating additional qubits as needed depending on the number of extended rank. Neither the type nor the scope of the operation used changes. The same operation is performed on the elements of the new vector. In the case of the addition, it can be implemented by adding another vector to the result of adding two vectors. Hence, our implementation offers scalability as the rank of the input lattice vector increases.

## 4 Evaluation

### 4.1 Experiment Environment

Our implementation utilized Qiskit ${ }^{4}$, a quantum computing platform. The cloud platform provides IBM's real hardware and simulators. Additionally, programming can be possible using Python and Qiskit's grammar, allowing access to the quantum computing environment. We use the 'matrix_product_state' simulator, which can provide relatively large-scale qubits.

### 4.2 Result of Quantum NV Sieve

Table 1 shows the results of each step of our implementation for quantum NV Sieve. The complement expression of $x$ is $\bar{x}$, and the abbreviation of the previous step is sometimes used in the next step to prevent the output term from becoming long. On the other hand, we present results for Default, Ex_DIM, and Ex_RANK. The extension to dimension (Ex-DIM) increases the length of the vector ( $v_{0}=$ $\{0,1\}$ to $v_{0}=\{0,0,0,1\}$ ). The extension to rank (Ex_RANK) increases the number of elements $\left(V=\left\{v_{0}, v_{1}\right\}\right.$ to $\left.V=\left\{v_{0}, \ldots, v_{n}\right\}\right)$.

Through the result of quantum NV Sieve logic, we present a scalable implementation that takes into account various situations on the lattice. Correct values are output at all steps. This allows us to verify the suitability of our quantum NV Sieve for practical implementation. Furthermore, this extended implementation can help raise the SVP upper limit that NV Sieve can solve. In our work, we confirmed that the NV Sieve algorithm operates accurately on a quantum circuit. Based on our work, we can expect that the possibility of solving the larger problem will increase as the scale of quantum computers expands.

[^4]18 Authors Suppressed Due to Excessive Length

Table 1: Results from each step of quantum NV Sieve to check whether it has been implemented correctly. (Default: 2-dimension and 2-rank, Ex_DIM: 4dimension and 2-rank, Ex_RANK: 2-dimension and 3-rank)

| Output | Default | Ex_DIM | Ex_RANK |
| :---: | :---: | :---: | :---: |
| $v_{0}$ | 000 | 00111 | 000 |
| $v_{1}$ | 001 | 00011 | 001 |
| $v_{2}$ | None | None | 001 |
| $c_{0}$ | 001 | 11001 | 001 |
| $c_{1}$ | 000 | 00101 | 001 |
| $c_{2}$ | None | None | 111 |
| $(\gamma R)^{2}$ | 000001 | 0000000001 | 000001 |
| $\overline{c_{0}}$ (when positive) | 111 | 11001 | 111 |
| $\overline{c_{1}}$ (when positive) | 000 | 11011 | 111 |
| $\overline{c_{2}}$ (when positive) | None | None | 111 |
| $v_{0}+\overline{c_{0}}:\left(v c_{0}\right)$ | 111 | 00000 | 111 |
| $v_{1}+\overline{c_{1}}:\left(v c_{1}\right)$ | 001 | 11110 | 000 |
| $v_{2}+\overline{c_{2}}:\left(v c_{2}\right)$ | None | None | 000 |
| $\left(v c_{0}\right)^{2}$ | 001 | 0000000000 | 001 |
| $\left(v c_{1}\right)^{2}$ | 001 | 0000000100 | 000 |
| $\left(v c_{2}\right)^{2}$ | None | None | 000 |
| $\left(v c_{0}\right)^{2}+\left(v c_{1}\right)^{2}+\left(v c_{2}\right)^{2}:\left(S u m_{v} c\right)$ | 000010 | 0000000100 | 000001 |
| $\overline{S u m_{v} \mathrm{c}}$ | 111110 | 1111111100 | 111111 |
| $\gamma R+\overline{S u m_{v} c}$ | 111111 | 1111111101 | 000000 |
| MSB | 1 | 1 | 0 |
| Shots | 100 |  |  |

### 4.3 Resource Estimation of Quantum NV Sieve

Table 2 shows the resource estimation of quantum NV Sieve. Since this is a resource estimate for Oracle, the result also includes resources for reverse operation. Contrary to the traditional Grover's search that identifies a single solution, the NV Sieve yields multiple outcomes. That is, it may produce multiple short vectors meeting the condition, with probabilities varying based on the number of shots. Therefore, determining the correct Grover's iteration is a very important issue.

The required quantum resources increase as the rank and dimension of the target vector increase. Even if the dimension is doubled, the total quantum cost increases by about 8.38 times, and even if the rank increases by just one, the total cost increases by about 1.98 times. However, a real lattice will have larger dimensions and ranks. Therefore, if the dimension and rank increase simultaneously, the quantum cost of the quantum NV Sieve is expected to increase enormously.

$$
\begin{equation*}
2 \cdot \# \text { gates } \cdot F D \cdot \text { iter } \tag{4}
\end{equation*}
$$

Additionally, when applying Grover, the total number of gates (\#gates) mentioned in Table 2 must be multiplied by full depth ( $F D$ ). Then, we need to multiply by 2 (reverse operation) and multiply by the number of Grover's iterations (iter). In other words, the formula for calculating Grover's attack cost is as shown in Equation 4. That is, in addition to quantum resources (i.e. the number of gates and circuit depth), Grover's iteration affects the attack cost. Table 3 is calculated from Table 2 and Equation 4. Table 3 shows the required quantum resources for Grover's search on NV Sieve. The number of qubits in every case increases by 1 because of the decision qubit. And, we get the appropriate iteration for these cases. Therefore, we calculate Grover's search cost on NV Sieve (Default,Ex_RANK and Ex_DIM).

Table 2: Resource Estimation of Quantum NV Sieve oracle.

| Case | \#CNOT | \#1qCliff | \#T | T-depth | full depth | \#Qubit |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Default | 291 | 69 | 124 | 396 | 1126 | 74 |
| Ex_RANK | 420 | 90 | 181 | 576 | 1631 | 105 |
| Ex_DIM | 685 | 224 | 296 | 878 | 2342 | 179 |

Table 3: Required quantum resources for Grover's search on NV Sieve.

| Case | Total gates | Full depth | T-depth | Quantum cost | \#Qubit |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Default | 972 | 2259 | 792 | 2195748 | 75 |
| Ex_RANK | 1403 | 3271 | 1152 | 4589213 | 106 |
| Ex_DIM | 2436 | 4714 | 1756 | 11483304 | 180 |

※: The appropriate iteration is 1 .

### 4.4 Further discussion

According to our implementation mentioned above, it is expected that quantum gain can be obtained through Grover's search. Of course, there will certainly be implementation challenges as follows. Also, in the current quantum computing environment, it is believed that there will be many difficulties from an implementation perspective to derive results similar to the theoretically proposed complexity of the quantum NV Sieve.

- Grover's iteration: Since iteration affects the cost, finding an iteration for a problem that has multiple solutions is the most important challenge in the practical implementation of Quantum NV Sieve. In this work, we get the proper iteration that ensures that only the correct answer is derived. We are conducting experiments on other cases (other extended implementations), and the results will be published in future research.
- Increase the upper limit: The important thing to solve the current SVP is to accurately find short vectors and increase the upper limit of the dimension that can be solved. In other words, the Sieve algorithm belongs to the exact algorithm, and it is important to solve it accurately starting from low dimensions. Therefore, we should start experimenting with low dimensions and ranks, as we do now, and then work our way up to higher limits.
- Optimizing the oracle circuit: In order to improve the efficiency of the quantum NV Sieve and maximize the benefits that arise from applying quantum, it is thought that optimal implementation of the oracle will be important. In other words, it appears that the optimal implementation of the oracle (NV Sieve quantum circuit), which determines the efficiency of quantum costs in Grover's search, must be progressed to solve SVP on a higherdimensional and rank lattice and obtain greater quantum advantages.
- NISQ era: As the resource estimation results indicate, quite a bit of attack cost is required despite the small dimensions and rank. Therefore, it is believed that there will be limitations in allowing general users to treat lattice vectors with higher rank and dimension. In other words, it is thought that solving SVP for high dimensions ( $50 \sim 60$ dimensions) such as classical is difficult for now.


## 5 Conclusion

In conclusion, there are quantum threats to traditional cryptographic systems, especially as quantum computing technology advances. While the most of research has focused on the potential impact of Grover's algorithm on symmetric key cryptography, the field of quantum attacks on lattice-based cryptography on Grover's search remains underexplored.

To address this gap and solve SVP on quantum computers, our work introduces a practical implementation of Quantum NV Sieve, designed to solve the SVP for hacking lattice-based cryptography. This implementation is an oracle that is a vital component of Grover's search algorithm. Furthermore, our work extends the Quantum NV Sieve implementation to handle various conditions (i.e., expansion of dimensions and rank of the lattice) thereby increasing its applicability and impact.

We estimate the quantum resources required for each case-specific implementation (oracle) and predict the cost of Grover's attacks when applied in conjunction with their Quantum NV Sieve. Like this, in a rapidly evolving quantum field, our research addresses the new potential quantum threats practically.

In our future work, we plan to find the correct Grover's iteration on other extended cases in the condition that there are multiple solutions, and successfully sieve the short vectors.

## 6 Acknowledgment

This work is funded by Utilizing Quantum Computing for Quantum Cryptography Confidentiality Analysis (N-23-NM-CU03-S01).

## References

1. P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in Proceedings 35th annual symposium on foundations of computer science, pp. 124-134, Ieee, 1994. 2
2. K. Jang, S. Choi, H. Kwon, H. Kim, J. Park, and H. Seo, "Grover on korean block ciphers," Applied Sciences, vol. 10, no. 18, p. 6407, 2020. 2
3. K. Jang, S. Choi, H. Kwon, and H. Seo, "Grover on speck: quantum resource estimates," Cryptology ePrint Archive, 2020. 2
4. K. Jang, G. Song, H. Kim, H. Kwon, H. Kim, and H. Seo, "Efficient implementation of present and gift on quantum computers," Applied Sciences, vol. 11, no. 11, p. $4776,2021.2$
5. K. Jang, A. Baksi, H. Kim, G. Song, H. Seo, and A. Chattopadhyay, "Quantum analysis of aes," Cryptology ePrint Archive, 2022. 2, 9
6. M. Rahman and G. Paul, "Grover on katan: Quantum resource estimation," IEEE Transactions on Quantum Engineering, vol. 3, pp. 1-9, 2022. 2
7. T. Laarhoven, M. Mosca, and J. Van De Pol, "Finding shortest lattice vectors faster using quantum search," Designs, Codes and Cryptography, vol. 77, pp. 375-400, 2015. 2, 6, 9

22 Authors Suppressed Due to Excessive Length
8. M. Ajtai, "The shortest vector problem in 12 is np-hard for randomized reductions," in Proceedings of the thirtieth annual ACM symposium on Theory of computing, pp. 10-19, 1998. 4
9. P. Q. Nguyen and B. Vallée, The LLL algorithm. Springer, 2010. 5
10. C.-P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," Mathematical programming, vol. 66, pp. 181-199, 1994. 5
11. M. Ajtai, R. Kumar, and D. Sivakumar, "A sieve algorithm for the shortest lattice vector problem," in Proceedings of the thirty-third annual ACM symposium on Theory of computing, pp. 601-610, 2001. 5
12. P. Q. Nguyen and T. Vidick, "Sieve algorithms for the shortest vector problem are practical," Journal of Mathematical Cryptology, vol. 2, no. 2, pp. 181-207, 2008. 5
13. X. Wang, M. Liu, C. Tian, and J. Bi, "Improved nguyen-vidick heuristic sieve algorithm for shortest vector problem," in Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, pp. 1-9, 2011. 6
14. F. Zhang, Y. Pan, and G. Hu, "A three-level sieve algorithm for the shortest vector problem," in International Conference on Selected Areas in Cryptography, pp. 2947, Springer, 2013. 6
15. T. Laarhoven, "Sieving for shortest vectors in lattices using angular localitysensitive hashing," in Advances in Cryptology-CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I 35, pp. 3-22, Springer, 2015. 6
16. A. Becker, N. Gama, and A. Joux, "Speeding-up lattice sieving without increasing the memory, using sub-quadratic nearest neighbor search," Cryptology ePrint Archive, 2015. 6
17. D. Micciancio and P. Voulgaris, "Faster exponential time algorithms for the shortest vector problem," in Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms, pp. 1468-1480, SIAM, 2010. 6
18. S. Jaques, M. Naehrig, M. Roetteler, and F. Virdia, "Implementing grover oracles for quantum key search on aes and lowmc," in Advances in CryptologyEUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II 30, pp. 280-310, Springer, 2020. 9
19. M. Kaplan, G. Leurent, A. Leverrier, and M. Naya-Plasencia, "Quantum differential and linear cryptanalysis," arXiv preprint arXiv:1510.05836, 2015. 11
20. S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, "A new quantum ripple-carry addition circuit," arXiv preprint quant-ph/0410184, 2004. 16

# Extended Attacks on ECDSA with Noisy Multiple Bit Nonce Leakages 

Shunsuke Osaki ${ }^{1}{ }^{[0000-0002-9221-8095]}$ and Noboru<br>Kunihiro ${ }^{1[0000-0003-1822-7476]}$<br>University of Tsukuba, Ibaraki, Japan<br>s2220571@u.tsukuba.ac.jp<br>kunihiro@cs.tsukuba.ac.jp


#### Abstract

It is well known that in ECDSA signatures, the secret key can be recovered if more than a certain number of tuples of random nonce partial information, corresponding message hash values, and signatures are leaked. There exist two established methods for recovering a secret key, namely lattice-based attack and Fourier analysis-based attack. When using the Fourier analysis-based attack, the number of signatures required for the attack can be evaluated through a precise calculation of the modular bias even if the leaked nonce contains errors. Previous works have focused on two cases: error-free cases and the case for the first MSB has errors among all of the nonce leakage. In this study, we extend the technique to the noisy multiple bits case to calculate the precise value of the modular bias for the case that multiple bits (say, $l$ bits from MSB) have errors. Aranha et al. (ACM CCS 2020) introduced a linear programming problem with parameters to evaluate the number of signatures, time, and memory required for a Fourier analysis-based attack. They also employed a SageMath module to optimize the number of signatures and time required for the attack. Furthermore, we show by experiments that 131 -bit ECDSA is vulnerable when the first MSB of the nonce is leaked without error and when 2 MSBs are leaked with an error rate 0.1 each, which implies that total error rate is about 0.19. We then show that the latter case requires less signatures to recover the secret key.


Keywords: ECDSA • Fourier analysis-based attack • Side-channel attack

## 1 Introduction

The Elliptic Curve Digital Signature Algorithm (ECDSA) is a digital signature algorithm that utilizes elliptic curves. It is widely used in various systems such as SSH, SSL/TLS, Bitcoin, and others. Therefore, evaluating the potential for leakage of secret information and the effect it may have on the overall security of a system is critical.

A nonce (Number used only ONCE) is secret information that is randomly generated during the signing process. However, it is possible to leak nonces
through side-channel attacks. An attack is reduced to the Hidden Number Problem (HNP) if a certain number of pairs of nonce partial bits, corresponding message hash values, and signatures are available [5]. Lattice-based and Fourier-analysis-based attacks are known as methods that solve HNPs.

A lattice-based attack can find a secret key with a relatively small number of signatures if the MSBs of the nonce are known without errors. If the secret key is 160 -bit and the 2 bits in a nonce is leaked [1][7][9]; if the secret key is 256 -bit and the 3 bits in a nonce is leaked [1][9]; or if the secret key is 384 -bit and the 4 bits in a nonce is leaked [1][9], then several dozens to several thousands of signatures can be used to recover the secret key in a few minutes to hours. Lattice-based attacks require more than 2 bits of nonce information without errors but do not require many signatures.

In a Fourier analysis-based attack, recovering the secret key is possible when the MSBs of the nonce are known without errors. If the key length is 192-bit [3] or 256 -bit [11], the signatures can be solved with a 1 or 2 bits leak with small errors, respectively. It was reported that several hundreds of millions of signatures and several days were required to solve the problem using workstations and clusters in those cases. In addition, the attack can also be successful if more MSBs are obtained with errors, but it requires many signatures, computational cost and time.

Aranha et al. [3] found vulnerabilities in OpenSSL 1.0.2 and 1.1.0, etc., against side-channel attacks that leak the MSB of ECDSA nonce, and used these vulnerabilities in their attacks. They estimated the number of signatures and costs of time, and memory of an attack when the 1 bit nonce is leaked with errors by estimating the modular bias. The number of signatures, cost of time, and memory required for the attack are also obtained by using the 4 -list sum algorithm for linear combination, which is critical in Fourier analysis-based attacks. They then reduced the problem of optimizing the number of signatures to a linear programming problem and solved it using the Mixed Integer Linear Program module of SageMath to optimize the number of signatures, costs of memory, and time required for the attack [10].

### 1.1 Our contributions

In this paper, we estimate the number of signatures, costs of time, and memory required for an attack in the case of multiple bits by estimating the modular bias when multiple MSBs with errors are obtained. In previous studies, modular bias has only been formulated for MSB leakage with errors or multiple bit leakage without errors. We have successfully generalized the formulation of the module bias. This allows us to estimate the modular bias in any case and to obtain an estimate of the number of signatures needed to recover a secret key.

We also focus on changes to the number of signatures when the error rate changes. Then, the optimal parameters are selected based on the evaluation of the number of obtained signatures. We extend their optimization program with a generalized modular bias to find the number of signatures required to recover the secret key. We also perform an actual attack against 131-bit ECDSA and
confirm that it is possible to recover the secret key. Furthermore, we show from both theoretical analysis for modular bias and experiment that the secret key is successfully recovered with fewer signatures when each of the 2 bits is leaked with an error rate of 0.1 than when the nonce is leaked with 1 bit without error.

## 2 Preliminaries

### 2.1 ECDSA signature generation algorithm

The set of solutions $(x, y) \in \mathbb{F} \times \mathbb{F}$ of an elliptic curve $E$ defined over a field $F$ with an infinity point $O$ is a commutative group derived from the chord-and-tangent rule.

The signature generation algorithm of the ECDSA is shown in Algorithm 1. The secret key sk is $\lambda$-bit. The secret information (i.e., nonce $k$ ) is randomly generated in the first line of Algorithm 1. In this study, we consider the case in which the MSBs of $k$ are leaked.

```
Algorithm 1 ECDSA signature generation
Input: prime number \(q\), secret key sk \(\in \mathbb{Z}_{q}\), message msg \(\in\{0,1\}^{*}\), base point on
    elliptic curve \(G\), and cryptographic hash function \(H:\{0,1\}^{*} \rightarrow \mathbb{Z}_{q}\)
Output: valid signatures ( \(r, s\) )
    \(k\) is chosen at random from \(\mathbb{Z}_{q}\)
    \(R=\left(r_{x}, r_{y}\right) \leftarrow k G ; r \leftarrow r_{x} \bmod q\)
    \(s \equiv(H(\mathrm{msg})+r \cdot \mathrm{sk}) / k \bmod q\)
    return \((r, s)\)
```


### 2.2 Hidden number problem with errors

The function $\operatorname{MSB}_{n}(x)$ returns the top $n$ bits of $x$ for a positive integer $x$. Let $b$ be a positive integer, $\{0,1\}^{b}$ be a fixed distribution on $\chi_{b}$, and the error bit sequence $e$ be sampled from $\chi_{b}$. The probabilistic algorithm $\operatorname{EMSB}_{\chi_{b}}(x)$ takes $x, b$ as input and returns $\operatorname{MSB}_{b}(x) \oplus e$. For each $i=1, \ldots, M$, let $z_{i}$ be $z_{i} \equiv k_{i}-h_{i} \cdot$ sk $\bmod q$ and $h_{i}, k_{i}$ be uniform random values on $\mathbb{Z}_{q}$. The HNP is the problem of finding sk that satisfies the aforementioned equations given the $h_{i}, z_{i}, \operatorname{EMSB}_{\chi_{b}}\left(k_{i}\right)$ obtained for each $i=1, \ldots, M$.

The ECDSA signature $(r, s)$ is generated according to Algorithm 1, nonce $k \in \mathbb{Z}_{q}$ is chosen uniformly at random, and $s \equiv(H(\mathrm{msg})+r \cdot \mathrm{sk}) / k(\bmod q)$ is satisfied. This yields the following equation.

$$
H(\mathrm{msg}) / s \equiv k-(r / s) \cdot \mathrm{sk} \bmod q
$$

If the MSBs of $k$ are obtained, we obtain an instance of HNP as $z \equiv H(\mathrm{msg}) / s$ $(\bmod q)$ and $h \equiv r / s(\bmod q)$,

4 S. Osaki et al.

### 2.3 Bias function and sample bias

We follow the idea of [3] and first show definitions of bias function and sample bias.

Definition 1. Let $\boldsymbol{K}$ be a random variable over $\mathbb{Z}_{q}$. The modulus bias $B_{q}(\boldsymbol{K})$ is defined as

$$
B_{q}(\boldsymbol{K})=\boldsymbol{E}[\exp ((2 \pi \boldsymbol{K} / q) \mathrm{i})]
$$

Let $\boldsymbol{E}(\boldsymbol{K})$ denote the expected value of random variable $\boldsymbol{K}$ and let i be an imaginary unit. In the same way, the sample bias of the set of points $K=\left\{k_{i}\right\}_{i=1}^{M}$ in $\mathbb{Z}_{q}$ is defined as

$$
\begin{equation*}
B_{q}(K)=\frac{1}{M} \sum_{i=1}^{M} \exp \left(\left(2 \pi k_{i} / q\right) \mathrm{i}\right) \tag{1}
\end{equation*}
$$

By fast Fourier transform (FFT), the computational complexity is $O(M \log M)$. For some positive integer $l$, let the higher $l$ bits of $\boldsymbol{K}$ be fixed to a certain constant, and the remaining $(\lambda-l)$ bits be random. The following equation is given in [11].

$$
\begin{equation*}
\lim _{q \rightarrow \infty}\left|B_{q}(\boldsymbol{K})\right|=\frac{2^{l}}{\pi} \cdot \sin \left(\frac{\pi}{2^{l}}\right) \tag{2}
\end{equation*}
$$

If no bits are fixed, its absolute value of sample bias is estimated as $1 / \sqrt{M}$. In addition, we can easily see that $\lim _{l \rightarrow \infty} \lim _{q \rightarrow \infty}\left|B_{q}(\boldsymbol{K})\right|=1$ from Equation (2).

The following lemma is given in [3].
Lemma 1. Suppose that the random variable $\boldsymbol{K}$ follows the following distribution on $\mathbb{Z}_{q}$ for $b \in\{0,1\}$, all $\varepsilon \in[0,1 / 2]$ and even $q>0$.

$$
\begin{cases}\operatorname{Pr}\left[\boldsymbol{K}=k_{i}\right]=(1-b) \cdot \frac{1-\varepsilon}{q / 2}+b \cdot \frac{\varepsilon}{q / 2} & \text { if } 0 \leq k_{i}<q / 2 \\ \operatorname{Pr}\left[\boldsymbol{K}=k_{i}\right]=b \cdot \frac{1-\varepsilon}{q / 2}+(1-b) \cdot \frac{\varepsilon}{q / 2} & \text { if } \quad q / 2 \leq k_{i}<q\end{cases}
$$

Letting $\boldsymbol{K}_{b}$ be a uniform distribution over $[b q / 2,(b+1) q / 2)$, the modular bias of $\boldsymbol{K}$ is given by

$$
\begin{equation*}
B_{q}(\boldsymbol{K})=(1-2 \varepsilon) B_{q}\left(\boldsymbol{K}_{b}\right) \tag{3}
\end{equation*}
$$

It can be easily verified that $\left|B_{q}\left(\boldsymbol{K}_{0}\right)\right|=\left|B_{q}\left(\boldsymbol{K}_{1}\right)\right|$. Note that Equation (3) considers only 1 bit leakage. The absolute value of $B_{q}(\boldsymbol{K})$ is given by

$$
\begin{equation*}
\left|B_{q}(\boldsymbol{K})\right|=(1-2 \varepsilon) \cdot \frac{2}{\pi} \sin \frac{\pi}{2} \tag{4}
\end{equation*}
$$

### 2.4 Fourier analysis-based attack

Bleichenbacher introduced Fourier analysis based attack in [4]. First, we consider a naive search method to obtain the secret key sk using the bias function, which
is shown in Algorithm 2. Let $M$ be the number of signatures obtained. In the case in which the input sample $\left\{\left(z_{i}, h_{i}\right)\right\}_{i=1}^{M}$ is biased $K_{i}$, we randomly select a candidate secret key $w \in \mathbb{Z}_{q}$ and then calculate $K_{w}=\left\{z_{i}+h_{i} w \bmod q\right\}_{i=1}^{M}$. Next, we compute $\left|B_{q}\left(K_{w}\right)\right|$ under Equation (1). If $w=\mathrm{sk}$, then $K_{w}$ is biased and $\left|B_{q}\left(K_{w}\right)\right|$ has the peak. Then finding the correct key sk is possible. However, this method is inefficient because it must search $w$ in all $\mathbb{Z}_{q}$.

```
Algorithm 2 Naive search
Input: \(\left(h_{i}, z_{i}\right)_{i=1}^{M}\) : HNP samples over \(\mathbb{Z}_{q}\)
Output: Correct secret key sk
    // Select a candidate \(w\) for the secret key.
    for \(w=1\) to \(q-1\) do
        Calculate \(K_{w}=\left\{z_{i}+h_{i} w \bmod q\right\}_{i=1}^{M}\).
        Calculate \(\left|B_{q}\left(K_{w}\right)\right|\).
    end for
    return \(w\) which maximizes \(\left|B_{q}\left(K_{w}\right)\right|\).
```

De Mulder et al. [8] and Aranha et al. [2] proposed a method to efficiently search for a secret key without performing an exhaustive search. Their methods perform a linear combination of input samples to satisfy $h_{j}^{\prime}<L_{\mathrm{FFT}}$ until $M^{\prime}$ samples are obtained. Consequently, a new linear combined sample $\left\{\left(h_{j}^{\prime}, z_{j}^{\prime}\right)\right\}_{j=1}^{M^{\prime}}$ is generated. The width of the peak $w$ is extended from 1 to approximately $q / L_{\mathrm{FFT}}$, showing that recovering the higher $\log L_{\mathrm{FFT}}$ bits of the secret key is possible. In a Fourier analysis-based attack, the entire secret key is recovered by repeating this process.

Let $\lambda^{\prime}$ be the number of already recovered bits in sk. At the first step of Fourier analysis-based attack, $\lambda^{\prime}=\log L_{\mathrm{FFT}}$. Letting the higher $\lambda^{\prime}$ bits of sk be $\mathrm{sk}_{\mathrm{hi}}$ and the unknown lower $\left(\lambda-\lambda^{\prime}\right)$ bits be sk ${ }_{\mathrm{lo}}$, sk can be expressed as $\mathrm{sk}=2^{\lambda-\lambda^{\prime}} \mathrm{sk}_{\mathrm{hi}}+\mathrm{sk}_{\mathrm{lo}}$. Thus, the new HNP formula for the case in which the higher $\lambda^{\prime}$ bits of sk has already been recovered can be rewritten as

$$
\begin{aligned}
k & \equiv z+h \cdot\left(2^{\lambda-\lambda^{\prime}} \mathrm{sk}_{\mathrm{hi}}+\mathrm{sk}_{\mathrm{lo}_{\mathrm{o}}}\right) \bmod q \\
k & \equiv z+h \cdot 2^{\lambda-\lambda^{\prime}} \mathrm{sk}_{\mathrm{hi}}+h \cdot \mathrm{sk}_{\mathrm{lo}_{\mathrm{o}}} \bmod q \\
k & \equiv \hat{z}+h \cdot \mathrm{sk}_{\mathrm{lo}_{\mathrm{o}}} \bmod q
\end{aligned}
$$

where $\hat{z}=z+h \cdot 2^{\lambda-\lambda^{\prime}}$ sk $_{\text {hi }}$. Thus, we obtain the new HNP samples $\left\{\left(\hat{z}_{i}, h_{i}\right)\right\}_{i=1}^{M}$. When the Fourier analysis-based attack is repeated, $\lambda^{\prime}$ increases. The $\hat{z}$ is updated in each repetition and, finally, the whole of sk can be recovered.

Algorithm 3 shows Bleichenbacher's attack framework for a Fourier analysisbased attack. The range reduction phase of the algorithm considers two constraints on linear combinations for efficient key searches, namely, small and sparse linear combinations.

```
Algorithm 3 Bleichenbacher's attack framework
Input: \(\left\{\left(h_{i}, z_{i}\right)\right\}_{i=1}^{M}\) : Sample of HNP over \(\mathbb{Z}_{q} . M^{\prime}\) : Number of linear combinations to
    find. \(L_{\mathrm{FFT}}\) : FFT table size.
Output: \(\mathrm{MSB}(\mathrm{sk})_{\log L_{\mathrm{FFT}}}\).
    Range reduction
    For all \(j \in\left[1, M^{\prime}\right]\), the coefficients are \(\omega_{i, j} \in\{-1,0,1\}\), and the linear combination
    pairs are denoted as \(\left(h_{j}^{\prime}, z_{j}^{\prime}\right)=\left(\sum_{i} \omega_{i, j} h_{i}, \sum_{i} \omega_{i, j} z_{i}\right)\). In this case, we generate \(M^{\prime}\)
    sample \(\left\{\left(h_{j}^{\prime}, z_{j}^{\prime}\right)\right\}_{j=1}^{M^{\prime}}\) that satisfies the following two conditions.
```

    (1) Small: \(0 \leq h_{j}^{\prime}<L_{\mathrm{FFT}}\).
    (2) Sparse : \(\left|B_{q}(K)\right|^{\Omega_{j}} \gg 1 / \sqrt{M^{\prime}}\), where \(\Omega_{j}:=\sum_{i}\left|\omega_{i, j}\right|\) for all \(j \in\left[1, M^{\prime}\right]\).
    
## Bias computation

$Z:=\left(Z_{0}, \ldots Z_{L_{\mathrm{FFT}}-1}\right) \leftarrow(0, \ldots, 0)$
for $j=1$ to $M^{\prime}$ do
$Z_{h_{j}^{\prime}} \leftarrow Z_{h_{j}^{\prime}}+\exp \left(\left(2 \pi z_{j}^{\prime} / q\right) \mathrm{i}\right)$
end for
Let $w_{i}=i q / L_{\mathrm{FFT}},\left\{B_{q}\left(K_{w_{i}}\right)\right\}_{i=0}^{L_{\mathrm{FFT}}-1} \leftarrow \operatorname{FFT}(Z)$
$=\left(B_{q}\left(K_{w_{0}}\right), B_{q}\left(K_{w_{1}}\right), \ldots, B_{q}\left(K_{w_{L_{\mathrm{FFT}}-1}}\right)\right)$.
Find $i$ that maximizes $\left|B_{q}\left(K_{w_{i}}\right)\right|$.
return $\mathrm{MSB}\left(w_{i}\right)_{\log L_{\mathrm{FFT}}}$.

In the small linear combination constraint, it should be satisfied that $\omega_{i, j} \in$ $\{-1,0,1\}$ and $h_{j}^{\prime}=\sum_{i=1}^{M} \omega_{i, j} h_{i}<L_{\mathrm{FFT}}$. This constraint is used to reduce the search range by linear combinations. To enable $h_{j}^{\prime}$ to be smaller, we can take linear combinations with a greater number of $h_{i}$ (i.e., a fewer number of $\omega_{i, j}=0$ ). The fewer the number of linear combinations, the smaller $L_{\text {FFT }}$ becomes, and thus the width of the peak, $q / L_{\text {FFT }}$ increase. However, if too many linear combinations are taken, the peak value decreases exponentially. Although the original peak value is $\left|B_{q}(\boldsymbol{K})\right|$, the peak bias after linear combinations is $\left|B_{q}(\boldsymbol{K})\right|^{\Omega_{j}}$, due to constraint, which exponentially decreases if we take $\Omega_{j}$ linear combinations. If the peak value is sufficiently larger than the average of the noise $1 / \sqrt{M^{\prime}}$, it can be distinguished. Therefore, constraints are imposed as sparse linear combinations to distinguish them from noise values.

The constraints of sparse linear combinations limit the number of linear combinations that can be taken such that the peak value is prevented from becoming too small. Now, estimating the number of samples $M^{\prime}$ after the linear combination (assuming that $\Omega_{j}$ is constant) depends only on $\left|B_{q}(\boldsymbol{K})\right|$, and finding the modular bias in a rigorous manner is critical. In a Fourier analysis-based attack, bias computation is performed using FFT, which has a computational complexity of $O\left(L_{\mathrm{FFT}} \log L_{\mathrm{FFT}}\right)$ and can thus be calculated efficiently. However, range reduction is not known to be inefficient and requires considerable computational time. Table 3 in [3] shows that the bias computation (FFT) consumes 1 hour, but range reduction (collision) consumes 42 hours when the key length is 162-bit, and the nonce is 1 bit leak with $\varepsilon=0.027$.

## 2.5 $\mathcal{K}$-list sum problem

Let the birthday problem be the problem of choosing $x_{1} \in \mathcal{L}_{1}$ and $x_{2} \in \mathcal{L}_{2}$ from 2 lists $\mathcal{L}_{1}$ and $\mathcal{L}_{2}$ with random $n$ bits elements that satisfy $x_{1} \oplus x_{2}=0$. In addition, given a list of $\mathcal{K}$ with $n$ bits values, the problem of selecting 1 of elements from each list and finding a pair of values for which the XOR of those $\mathcal{K}$ values is 0 is known as the Generalized Birthday Problem (GBP). In [12], Bleichenbacher observed similarities between GBP and the Fourier analysisbased attack [4]. The $\mathcal{K}$-list sum algorithm solves $\mathcal{K}$-list sum problem [6] which is the GBP subproblem.

Aranha et al. [3] used the $\mathcal{K}$-list sum algorithm to increase the number of samples while increasing the widths of peaks through linear combination. Algorithm 4 shows a 1 -fold 4 -list sum algorithm. Algorithm 4 first finds the pairs from two of the given four lists such that the higher $a$ bits of the sum is a certain value, and it stores the sum in sorted lists $\mathcal{L}_{1}^{\prime}$ and $\mathcal{L}_{2}^{\prime}$. Next, from $\mathcal{L}_{1}^{\prime}$ and $\mathcal{L}_{2}^{\prime}$, select a pair $\left(x_{1}^{\prime}, x_{2}^{\prime}\right)$ whose higher $n$ bits are equal and calculate the absolute difference $\left|x_{1}^{\prime}-x_{2}^{\prime}\right|$, where the higher $n$ bits are 0 . We then obtain sorted lists with $(\lambda-n)$ bits elements. Because the higher $a$ bits are first chosen to be equal, we only need to check whether $(a-n)$ bits are equal. The algorithm increases the $M=2^{m}=2^{a+2}$ sequences of length $\lambda$ received as input to $2^{3 a+v-n}$ sequences of length $(\lambda-n)$ by linear combination.

```
Algorithm 4 Parameterized 4-list sum algorithm based on Howgrave-Graham-
Joux
```

Input: $\left\{\mathcal{L}_{i}\right\}_{i=1}^{4}$ : Sorted list of uniform random samples of $\lambda$ bits uniform random
samples of length $2^{a} . n$ : Number of higher bits to be discarded in each round.
$v \in[0, a]$ : Parameter
Output: $\mathcal{L}^{\prime}:$ List of $(\lambda-n)$-bit samples

1: For each $c \in\left[0,2^{v}\right)$ :
(a) Search for a pair $\left(x_{1}, x_{2}\right) \in \mathcal{L}_{1} \times \mathcal{L}_{2}$ satisfying $\operatorname{MSB}_{a}\left(x_{1}+x_{2}\right)=c$. Output a new sorted list $\mathcal{L}_{1}^{\prime}$ with $x_{1}+x_{2}$ as $2^{a} \cdot 2^{a} \cdot 2^{-a}=2^{a}$ elements. Similarly, for $\mathcal{L}_{3}, \mathcal{L}_{4}$, the sorted list $\mathcal{L}_{2}^{\prime}$ is obtained.
(b) Search for a pair $\left(x_{1}^{\prime}, x_{2}^{\prime}\right) \in \mathcal{L}_{1}^{\prime} \times \mathcal{L}_{2}^{\prime}$ satisfying $\operatorname{MSB}_{n}\left(\left|x_{1}^{\prime}-x_{2}^{\prime}\right|\right)=0$. Output a new sorted list $\mathcal{L}^{\prime}$ with $\left|x_{1}^{\prime}-x_{2}^{\prime}\right|$ as $2^{a} \cdot 2^{a} \cdot 2^{-(n-a)}=2^{3 a-n}$ elements.
return $\mathcal{L}^{\prime}$

Algorithm 5 is an iterative 4-list sum algorithm that calls Algorithm 4 as a subroutine. If $2^{a}$ is the length of each sublist, it can be expressed as $M=2^{m}=$ $4 \cdot 2^{a}=2^{a+2}$. Let $n$ be the number of higher bits to be nullified, and let $N=2^{n}$. $M^{\prime}=2^{m^{\prime}}<2^{2 a}$ is the number of samples output with the higher $n$ bits as 0. In addition, $v$ is the number of iterations in range reduction with $v \in[0, a]$, and $T=2^{t}=2^{a+v}$ and $T$ is the time complexity. From [6], it holds that $T M^{2}=N$. Now, the $N$ is $2^{4} M^{\prime} N$ and therefore the following holds.

$$
\begin{equation*}
2^{4} M^{\prime} N=T M^{2} \tag{5}
\end{equation*}
$$

From Equation (5), we obtain

$$
\begin{equation*}
m^{\prime}=3 a+v-n \tag{6}
\end{equation*}
$$

Let $r$ be the number of times the attacker repeats the 4 -list sum algorithm. By iterating, find a small linear combination of $4^{r}$ integers that satisfies the budget parameter of the FFT table so that it is less than $L_{\mathrm{FFT}}=2^{\ell_{\mathrm{FFT}}}$ and so that the FFT computation is tractable. In this case, the trade-off equation for each round $i=0, \ldots, r-1$ can be rewritten as

$$
\begin{equation*}
m_{i}^{\prime}=3 a_{i}+v_{i}-n_{i} \tag{7}
\end{equation*}
$$

where $m_{i+1}=m_{i}^{\prime}$. The output of the $i$-th round is used for the input of the $i+1$-th round.

Table 1 lists the constraints of a linear programming problem when Algorithm 5 is optimized in terms of time, memory, and the number of signatures. Consider the optimization case in which $m_{\text {in }}$ is minimized. Let $t_{\text {max }}$ be the maximum time spent in each round, $m_{\max }$ be the maximum memory, and $\ell_{\mathrm{FFT}}=\log L_{\mathrm{FFT}}$ be the memory size for the FFT. These are quantities determined by the amount that can be spent (i.e., cost). The $\alpha$ is a slack parameter that enables the peak to be more observable and depends on the maximum possible noise value. This value can be estimated by examining the distribution of $\left\{h_{j}^{\prime}\right\}_{j=1}^{M^{\prime}}$ and is given by approximately $\sqrt{2 \ln \left(2 L_{\mathrm{FFT}} / \varepsilon\right)}[3]$.

Letting $m_{r}:=\log M^{\prime}, m_{r}=2\left(\log \alpha-4^{r} \log \left|B_{q}(\boldsymbol{K})\right|\right)$ is derived from the constraint of sparse linear combinations. Estimating $\left|B_{q}(\boldsymbol{K})\right|$ is sufficient to estimate the number of samples $M^{\prime}$ required after linear combination. In addition, $\left|B_{q}(\boldsymbol{K})\right|$ is the only value related to the number of bits $l$ in the leaked nonce. Depending on the length $\lambda$ of the secret key, each $n_{i}$ is differently chosen and the choice of $n_{i} \mathrm{~s}$ affects other parameters.

```
Algorithm 5 Iterative HGJ 4-list sum algorithm
Input: \(\mathcal{L}\) : List of \(M=4 \times 2^{a}\) uniforml random \(\lambda\)-bit samples. \(\left\{n_{i}\right\}_{i=0}^{r-1}\) : The number of
    higher bits to be discarded in each round. \(\left\{v_{i}\right\}_{i=0}^{r-1}\) : Parameters where \(v_{i} \in\left[0, a_{i}\right]\).
Output: \(\mathcal{L}^{\prime}\) : List of \(\left(\lambda-\sum_{i=0}^{r-1} n_{i}\right)\)-bit samples with length \(2^{m_{r}}\).
    : Let \(a_{0}=a\).
    : For each \(i=0, \ldots, r-1\) :
    (a) Divide \(\mathcal{L}\) into four lists \(\mathcal{L}_{1}, \mathcal{L}_{2}, \mathcal{L}_{3}, \mathcal{L}_{4}\) of length \(2^{a_{i}}\) and sort each list.
    (b) Give parameters \(n_{i}\) and \(v_{i}\) and \(\left\{\mathcal{L}_{i}\right\}_{i=1}^{4}\) to Algorithm 4. Obtain a single list \(\mathcal{L}^{\prime}\)
        of length \(2^{m_{i+1}}=2^{3 a_{i}+v_{i}-n_{i}}\). Let \(\mathcal{L}:=\mathcal{L}^{\prime}\) and \(a_{i+1}=m_{i+1} / 4\).
    return \(\mathcal{L}^{\prime}\)
```

Table 1. Linear programming problem based on iterative HGJ 4-list sum algorithm (Algorithm 5). Each column is a constraint to optimize time and space and data [3].


## 3 Modular bias for multiple bit leakage

Aranha et al. [3] discussed the security of ECDSA only for 1 bit noisy leakage. Considering practical circumstances, more bit leakage can be obtained. This section will analyze the security for the case where more noisy bits are obtained.

### 3.1 Modular bias for 2 bits leakage

We extend the evaluation of the modular bias for a single noisy bit case presented in Equation (3) to one when the nonce leaks multiple bits with errors. We begin with the most simple case: modular bias for $l=2$ and extend the result for general $l$. The modular bias is also given for the case in which each bit has a different error rate. The nonce obtained by a side-channel attack is not necessarily completely error-free. Thus far, evaluation of the case of nonce leakage with errors has been limited to the case of 1 bit leakage as done by Aranha et al. [3]. This work allows us to evaluate and discuss the security of the ECDSA in more detail by estimating the modular bias in the case of multiple bit leakage.

Lemma 2 (Modular bias for $l=2$ ). Suppose that the random variable $\boldsymbol{K}$ follows the following distribution over $\mathbb{Z}_{q}$ for $b \in\{0,1,2,3\}, \varepsilon_{1}, \varepsilon_{2} \in[0,1 / 2]$ and
even $q>0$.

$$
\left\{\begin{aligned}
\operatorname{Pr}\left[\boldsymbol{K}=k_{i}\right] & =\frac{(1-b)(2-b)(3-b)}{6} \cdot \frac{\left(1-\varepsilon_{1}\right)\left(1-\varepsilon_{2}\right)}{q / 4}+\frac{b(2-b)(3-b)}{2} \cdot \frac{\varepsilon_{1} \varepsilon_{2}}{q / 4} \\
& -b(1-b)(3-b) \cdot \frac{\varepsilon_{1}\left(1-\varepsilon_{2}\right)}{q / 4}+\frac{b(1-b)(2-b)}{6} \cdot \frac{\left(1-\varepsilon_{1}\right) \varepsilon_{2}}{q / 4} \text { if } 0 \leq k_{i}<q / 4 \\
\operatorname{Pr}\left[\boldsymbol{K}=k_{i}\right] & =\frac{(1-b)(2-b)(3-b)}{6} \cdot \frac{\left(1-\varepsilon_{1}\right) \varepsilon_{2}}{q / 4}+\frac{b(2-b)(3-b)}{2} \cdot \frac{\left(1-\varepsilon_{1}\right)\left(1-\varepsilon_{2}\right)}{q / 4} \\
& -b(1-b)(3-b) \cdot \frac{\varepsilon_{1} \varepsilon_{2}}{q / 4}+\frac{b(1-b)(2-b)}{6} \cdot \frac{\varepsilon_{1}\left(1-\varepsilon_{2}\right)}{q / 4} \text { if } q / 4 \leq k_{i}<q / 2 \\
\operatorname{Pr}\left[\boldsymbol{K}=k_{i}\right] \quad & =\frac{(1-b)(2-b)(3-b)}{6} \cdot \frac{\varepsilon_{1}\left(1-\varepsilon_{2}\right)}{q / 4}+\frac{b(2-b)(3-b)}{2} \cdot \frac{\left(1-\varepsilon_{1}\right) \varepsilon_{2}}{q / 4} \\
& -b(1-b)(3-b) \cdot \frac{\left(1-\varepsilon_{1}\right)\left(1-\varepsilon_{2}\right)}{q / 4}+\frac{b(1-b)(2-b)}{6} \cdot \frac{\varepsilon_{1} \varepsilon_{2}}{q / 4} \text { if } q / 2 \leq k_{i}<3 q / 4 \\
\operatorname{Pr}\left[\boldsymbol{K}=k_{i}\right] \quad & =\frac{(1-b)(2-b)(3-b)}{6} \cdot \frac{\varepsilon_{1} \varepsilon_{2}}{q / 4}+\frac{b(2-b)(3-b)}{2} \cdot \frac{\varepsilon_{1}\left(1-\varepsilon_{2}\right)}{q / 4} \\
& -b(1-b)(3-b) \cdot \frac{\left(1-\varepsilon_{1}\right) \varepsilon_{2}}{q / 4}+\frac{b(1-b)(2-b)}{6} \cdot \frac{\left(1-\varepsilon_{1}\right)\left(1-\varepsilon_{2}\right)}{q / 4} \text { if } 3 q / 4 \leq k_{i}<q
\end{aligned}\right.
$$

Let $\boldsymbol{K}_{b}$ be a uniform distribution over $[b q / 4,(b+1) q / 4)$. The modular bias of $\boldsymbol{K}$ is then given by

$$
B_{q}(\boldsymbol{K})=\left\{\left(1-2 \varepsilon_{1}\right)\left(1-\varepsilon_{2}\right)+\mathrm{i}\left(1-2 \varepsilon_{1}\right) \varepsilon_{2}\right\} B_{q}\left(K_{b}\right)
$$

Proof. See Appendix A.
Remark 1. We now consider the case in which $\varepsilon_{2}=0.5$, (i.e., the same case in which no bias exists in the second bit, which is completely random). In this case, the absolute value of the bias is given by
$\left|B_{q}(\boldsymbol{K})\right|=\left|\left(1-2 \varepsilon_{1}\right) \times 0.5+\mathrm{i}\left(1-2 \varepsilon_{1}\right) \times 0.5\right| \cdot \frac{2^{2}}{\pi} \sin \frac{\pi}{2^{2}}=\left(1-2 \varepsilon_{1}\right) \cdot \frac{2^{1}}{\pi} \sin \frac{\pi}{2^{1}}$.
We can easily verify that the value is equal to Equation (4), which is the expression for $l=1$. In addition, it is better to point out that the bias is 0 regardless of the value of $\varepsilon_{2}$ in the case of $\varepsilon_{1}=0.5$.

### 3.2 Generalization to modular bias for multiple bit leakage

We next generalize the modular bias to the case in which the higher $l$ bits of the nonce leaks with errors. To simplify the discussion, consider the case where each bit contains an error with probability $\varepsilon$. Given $l$, let $\boldsymbol{K}_{b}$ be a uniform distribution over $\left[b q / 2^{l},(b+1) q / 2^{l}\right) . b \in\left\{0,1, \ldots, 2^{l}-1\right\}$. We can easily verify that all of $\left|B_{q}\left(\boldsymbol{K}_{b}\right)\right|$ are equal regardless of the value of $b$. Therefore, it is enough to obtain $B_{q}\left(\boldsymbol{K}_{0}\right)$. Let $\mathcal{H}(j)$ be the Hamming weight when $j$ is expressed in binary. If the higher $l$ bits of the nonce are all 0 and no errors occur, $\boldsymbol{K}_{0}$ corresponding to $b=0$ is uniformly distributed over $\left[0, q / 2^{l}\right)$. When an error is contained in each bit with probability $\varepsilon$, each bit is 1 with probability $\varepsilon$. Thus, the number of bits containing errors is the same as the number of 1 bits and can be expressed in terms of Hamming weights. In addition, the number of error-free bits is $l-\mathcal{H}(j)$. From this, for the higher $l$ bits, if an error occurs in each bit with an error rate of $\varepsilon$, the modular bias is expressed as

$$
\begin{equation*}
\left\{\sum_{j=0}^{2^{l}-1} \exp \left(\frac{2 j \pi}{2^{l}} \mathrm{i}\right) \varepsilon^{\mathcal{H}(j)}(1-\varepsilon)^{l-\mathcal{H}(j)}\right\} B_{q}\left(\boldsymbol{K}_{0}\right) \tag{8}
\end{equation*}
$$

We next simplify the term $\sum_{j=0}^{2^{l}-1} \exp \left(2 j \pi \mathrm{i} / 2^{l}\right) \varepsilon^{\mathcal{H}(j)}(1-\varepsilon)^{l-\mathcal{H}(j)}$ appeared in Equation (8).

$$
\begin{aligned}
& \sum_{j=0}^{2^{l}-1} \exp \left(\frac{2 j}{2^{l}} \pi \mathrm{i}\right) \varepsilon^{\mathcal{H}(j)}(1-\varepsilon)^{l-\mathcal{H}(j)} \\
& =\sum_{j=0}^{2^{l-1}-1} \exp \left(\frac{4 j}{2^{l}} \pi \mathrm{i}\right) \varepsilon^{\mathcal{H}(2 j)}(1-\varepsilon)^{l-\mathcal{H}(2 j)}+\sum_{j=0}^{2^{l-1}-1} \exp \left(\frac{4 j+2}{2^{l}} \pi \mathrm{i}\right) \varepsilon^{\mathcal{H}(2 j+1)}(1-\varepsilon)^{l-\mathcal{H}(2 j+1)} \\
& =\sum_{j=0}^{2^{l-1}-1} \exp \left(\frac{2 j}{2^{l-1}} \pi \mathrm{i}\right) \varepsilon^{\mathcal{H}(2 j)}(1-\varepsilon)^{l-1+1-\mathcal{H}(2 j)} \\
& +\sum_{j=0}^{2^{l-1}-1} \exp \left(\frac{2 j}{2^{l-1}} \pi \mathrm{i}+\frac{2}{2^{l}} \pi \mathrm{i}\right) \varepsilon^{\mathcal{H}(2 j)+1}(1-\varepsilon)^{l-(\mathcal{H}(2 j)+1)} \\
& =\sum_{j=0}^{2^{l-1}-1} \exp \left(\frac{2 j}{2^{l-1}} \pi \mathrm{i}\right) \varepsilon^{\mathcal{H}(j)}(1-\varepsilon)^{l-1-\mathcal{H}(j)} \times\left\{(1-\varepsilon)+\varepsilon \exp \left(\frac{2}{2^{l}} \pi \mathrm{i}\right)\right\} \\
& =\prod_{j=1}^{l}\left((1-\varepsilon)+\varepsilon \exp \left(\frac{2 \pi \mathrm{i}}{2^{j}}\right)\right)
\end{aligned}
$$

During the equation transformation, we use the equation $\mathcal{H}(2 j+1)=\mathcal{H}(j)+1$ for a non-negative integer $j$. Note that in the case of $b=0$, we just consider the Hamming distance to the binary representation $00 \cdots 0$ of the $l$-bit. In the general $b$ case, we slightly modify to consider the Hamming distance to the binary representation of $b$. From this, the bias with error is expressed by the following theorem.
Theorem 1. The modular bias for the l-bit nonce leakage with error rate $\varepsilon$ is given by

$$
\begin{equation*}
\prod_{j=1}^{l}\left((1-\varepsilon)+\varepsilon \exp \left(\frac{2 \pi \mathrm{i}}{2^{j}}\right)\right) B_{q}\left(\boldsymbol{K}_{b}\right) \tag{9}
\end{equation*}
$$

For the absolute value of the modular bias, the following holds and can be expressed without using complex numbers.

Corollary 1. The absolute value of the modular bias for the l-bit nonce leakage with error rate $\varepsilon$ is given by

$$
\begin{align*}
& \left|\prod_{j=1}^{l}\left((1-\varepsilon)+\varepsilon \exp \left(\frac{2 \pi \mathrm{i}}{2^{j}}\right)\right)\right|\left|B_{q}\left(\boldsymbol{K}_{b}\right)\right| \\
& =\sqrt{\prod_{j=1}^{l}\left(1-4 \varepsilon(1-\varepsilon) \sin ^{2} \frac{\pi}{2^{j}}\right)\left|B_{q}\left(\boldsymbol{K}_{b}\right)\right| .} \tag{10}
\end{align*}
$$

Here, a simple calculation confirms that the absolute value of the modular bias is 0 in Equation (8)-(10) if $\varepsilon=0.5$.

Corollary 1 can be used to find the absolute value of the modular bias for a given number of bits and the leakage error rate. The concrete values are shown in Table 2. Each column is the number of bits leaked by the nonce, and each row is the value of the nonce's error rate.

Only the values for $\varepsilon=0$ are shown in Table 1 of [8]. Only the values for $l=1$ are shown in Lemma 4.2 of [3]. With the help of Corollary 1, we can calculate the precise absolute value of the modilar bias for arbitrary $\varepsilon$ and $l$ (as shown in yellow in the table).

These values are extended to Figure 1 shows the modular bias plotted for each error rate. We can find that the value increases as $l$ increases and depends on the error rate. It converges to some value that depends on the error rate $\varepsilon$ at approximately $l=6$. Moreover, we can see that the graph for $\varepsilon=0.01$ has almost the same shape as that for $\varepsilon=0$. In [3], they attacked in $\varepsilon=0.01$ and $\varepsilon=0.027$ cases and succeeded in recovering the secret keys.

The modular bias for $\varepsilon=0.1$ and $l \geq 2$ is larger than that for $\varepsilon=0$ and $l=1$. This means that the number of signatures for a 2 bits leak with an error rate of 0.1 is less than that for a 1-bit leak with no errors. Thus, fewer signatures are required for a successful attack. We give experimental reults comaring two cases in 4.2.

Table 2. Absolute values of modular bias

| $l$ | 1 | 2 | 3 | 4 | 5 | 6 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\varepsilon=0$ | 0.6366 | 0.9003 | 0.9749 | 0.9935 | 0.9983 | 0.9999 |
| $\varepsilon=0.01$ | 0.6238 | 0.8735 | 0.9427 | 0.9605 | 0.9649 | 0.9660 |
| $\varepsilon=0.1$ | 0.5092 | 0.6522 | 0.6870 | 0.6957 | 0.6978 | 0.6984 |
| $\varepsilon=0.3$ | 0.2546 | 0.2742 | 0.2780 | 0.2788 | 0.2790 | 0.2791 |
| $\varepsilon=0.4$ | 0.1273 | 0.1298 | 0.1302 | 0.1303 | 0.1304 | 0.1304 |

### 3.3 Case for different error rate of each bit

Equation (10), as presented in Section 3.2, shows the absolute value of the modular bias for which the error rates of each bit are equal (say, $\varepsilon$ ). We next show the modular bias for the different error rates of each bit.

Again, we consider $\boldsymbol{K}_{0}$ and we attempt to update the value corresponding to $\prod_{j=1}^{l}\left((1-\varepsilon)+\varepsilon \exp \left(2 \pi \mathrm{i} / 2^{j}\right)\right)$ in Equation (9). The values up to $j=1$ and $j=2$ are $(1-\varepsilon)-\varepsilon$ and $((1-\varepsilon)-\varepsilon)((1-\varepsilon)+\varepsilon i)$, respectively. Thus, we are considering cases when the MSBs do not contain errors and when MSBs contain errors. Multiplying by $1-\varepsilon$ and $\varepsilon$ i enables us to consider those cases in which the second MSB error is not included and when it is included, respectively. In general $j, 1-\varepsilon$ and $\varepsilon \exp \left(2 \pi \mathrm{i} / 2^{j}\right)$ can be considered as the error-free


Fig. 1. Modular bias under multiple bit leakage with errors.
bits and error-containing bits, respectively. In other words, at the $j$-th factor, $((1-\varepsilon)-\varepsilon) \cdots\left((1-\varepsilon)+\varepsilon \exp \left(2 \pi \mathrm{i} / 2^{j}\right)\right)$ is considered as each $2^{j-1}$ combination of the $(j-1)$ bits from the MSB to the $(j-1)$-th bit, with and without errors. Therefore, to establish the case in which the $j$-th bit does not include an error, we multiply by $1-\varepsilon$. To create the case where the $j$-th bit contains an error, we multiply by $\varepsilon \exp \left(2 \pi \mathrm{i} / 2^{j}\right)$. From this, we can say that the $j$-th $\varepsilon$ represents the error rate of the $j$-th bit from the MSB. If the error rate of each leaked bit in the nonce is different, we denote $\varepsilon_{j}$ as the error rate of the $j$-th MSB. The modular bias in the case in which the error rate is different for each bit of Theorem 1 is as in the following theorem.

Theorem 2. The modular bias when the nonce leaks l-bit with an error rate $\varepsilon_{j}$ is given by

$$
\begin{equation*}
\prod_{j=1}^{l}\left(\left(1-\varepsilon_{j}\right)+\varepsilon_{j} \exp \left(\frac{2 \pi \mathrm{i}}{2^{j}}\right)\right) B_{q}\left(\boldsymbol{K}_{b}\right) . \tag{11}
\end{equation*}
$$

In the case of $l=2$, it matches Lemma 2 .

The absolute value of the modular bias for the different error rates of each bit is given by

$$
\begin{equation*}
\sqrt{\prod_{j=1}^{l}\left(1-4 \varepsilon_{j}\left(1-\varepsilon_{j}\right) \sin ^{2} \frac{\pi}{2^{j}}\right)}\left|B_{q}(\boldsymbol{K})\right| \tag{12}
\end{equation*}
$$

If $\varepsilon_{j}=\varepsilon$ for all $j$, Equation (12) is equal to Equation (10). As Equation (12) shows, the error rates of the higher bits have a greater effect on the modular bias. Table 3 shows the values of $\sin ^{2}\left(\pi / 2^{j}\right)$ for each $j$. This shows that the contribution for $j=1$ is much greater than for the other cases. Figure 2 shows the exact values of the modular bias when the error rates between the first bit and second and after bits are different. The figure indicates that the absolute value of the bias is greater when the error rate of the first bit is smaller than that of the second and subsequent bits as compared to when the error rate of the first bit is greater than that of the second and subsequent bits. In other words, if the error rate is different for each bit of the nonce, the modular bias is highly dependent on the first MSB. This can be seen from the approximated equation, since for small $x, \sin ^{2} x$ is approximated as $x^{2}$. That is, $\sin ^{2}\left(\pi / 2^{j}\right) \approx \pi^{2} / 2^{2 j}$ if $j \geq 5$. A visual explain of the bias function for multi-bit leakage associated with this value is shown in Appendix B.

Table 3. Values of $\sin ^{2} \frac{\pi}{2^{j}}$ at each $j$.

$$
\begin{array}{c|c|c|c|c|c|c|c|c}
j & 1 & 2 & 3 & 4 & 5 & 6 & \cdots & 10 \\
\hline \sin ^{2}\left(\pi / 2^{j}\right) & 1 & 0.5 & 0.146 & 0.038 & 0.009 & 0.002 & \cdots & 0.000009
\end{array}
$$

The term $\sqrt{1-4 \varepsilon_{j}\left(1-\varepsilon_{j}\right) \sin ^{2}\left(\pi / 2^{j}\right)}$ in Equation (12) can be expressed as

$$
\begin{equation*}
\sqrt{1-4 \varepsilon_{j}\left(1-\varepsilon_{j}\right)\left(\pi^{2} / 2^{2 j}\right)} \tag{13}
\end{equation*}
$$

from the above approximation. We can see that this term rapidly converges to 1 as $l \rightarrow \infty$, regardless of the value of $\varepsilon_{j}$.

Intuitively, the proof of Lemma 2 shows that $\operatorname{Pr}\left[\boldsymbol{K}=k_{i}\right]$ is designed so that one term of each $\operatorname{Pr}\left[\boldsymbol{K}=k_{i}\right]$ remains depending on the value of $b$. Related figures are shown in Figures 3 and 4. Figure 3 shows the modular bias for the 2 bits case, and Figure 4 shows the modular bias for the 3 bits case. The sum of the absolute values of the four or eight vectors is 1 , respectively. The absolute value of the sum of these vectors is the absolute value of the modular bias. An interesting fact is that a vector with 2 or 3 bits wrong has a smaller effect on the absolute value of the modular bias than a vector with only 1 bits wrong. In addition, Figures 3 and 4 show the sum of the vectors is 0 if $\varepsilon_{1}=0.5$, which is mentioned at the end of Remark 1.


Fig. 2. Modular bias for different error rate for each bit of nonce

### 3.4 The number of signatures required for key-recovery and error rates

The constraint of sparse linear combinations is given by $\left|B_{q}(K)\right|^{\Omega_{j}} \gg \alpha / \sqrt{M^{\prime}}$. Suppose that $\left|B_{q}(K)\right| \alpha$ are given for this inequality. We can satisfy the inequality by choosing smaller $\Omega_{j}$, which is the number of linear combinations, or larger $M^{\prime}$, which is the number of samples after linear combination. The number of samples after linear combination required for $r$ rounds is given by the following equation based on the error rate and bias.

$$
\begin{equation*}
M^{\prime} \gg \alpha /\left(\sqrt{\prod_{j=1}^{l}\left(1-4 \varepsilon_{j}\left(1-\varepsilon_{j}\right) \sin ^{2} \frac{\pi}{2^{j}}\right)}\left|B_{q}\left(\boldsymbol{K}_{b}\right)\right|\right)^{2 \times 4^{r}} \tag{14}
\end{equation*}
$$

From the fact that $v_{i} \leq a_{i}$ in the input of Algorithm 5 and Equation (7), we obtain $m_{i+1} \leq 4 m_{i}-n_{i}-8$. We then have the following.

$$
\begin{equation*}
m^{\prime}=m_{r} \leq 4^{r} m_{0}-\sum_{i=0}^{r-1} 4^{r-i-1} n_{i}-\frac{8}{3}\left(4^{r}-1\right) \tag{15}
\end{equation*}
$$

In the 4 -list sum algorithm, it holds that $t_{i}=a_{i}+v_{i}, m_{i}=a_{i}+2$, and $v_{i} \leq a_{i}$ in the input of Algorithm 5. Accordingly, the inequations $t_{i} \leq 2 m_{i}-4$ are obtained.

16 S. Osaki et al.



Fig. 3. Modular bias illustrated on Fig. 4. Modular bias illustrated on the unit the unit circle with 2 bits leakage. circle with 3 bits leakage.

We then have the sum of time complexity as follows.

$$
\begin{equation*}
\sum_{i=0}^{r-1} t_{i} \leq 2 \sum_{i=0}^{r-1} m_{i}-4 r \tag{16}
\end{equation*}
$$

From Equations (14) and (15), the estimated number of signatures required for the attack is bounded by

$$
\begin{equation*}
M \geq \frac{1}{\prod_{j=1}^{l}\left(1-4 \varepsilon_{j}\left(1-\varepsilon_{j}\right) \sin ^{2}\left(\pi / 2^{j}\right)\right)} \times \frac{1}{\left\{\left(2^{l} / \pi\right) \cdot \sin \left(\pi / 2^{l}\right)\right\}^{2}} \times 2^{\mathcal{A}} \tag{17}
\end{equation*}
$$

where

$$
\begin{equation*}
\mathcal{A}=\sum_{i=0}^{r-1} 4^{-i-1} n_{i}+\frac{8}{3}\left(1-4^{-r}\right) \tag{18}
\end{equation*}
$$

From Equation (17), we can see that a higher error rate increases the number of signatures required and that an increase in the length of the known nonce reduces the number of signatures required. For example, from Table 2, a comparison of $\varepsilon=0.01$ and $\varepsilon=0.1$ when $l=2$ reveals that $0.8725^{2} / 0.6522^{2} \approx 1.79$ times increase. In addition, comparing $l=3$ and $l=1$ for $\varepsilon=0.01$, we see that $0.9427^{2} / 0.6238^{2} \approx 2.284$ times increase in the number of signatures. Furthermore, we find that the error rate and size of the bias do not affect the number of signatures required, whereas the number of rounds is varied. Note that the values for $l=1$ are completely consistent with the evaluation of Aranha et al.

To understand Equation (17), we can break it down into three separate parts and analyze each one individually.

Third term, represented by $2^{\mathcal{A}}$, remains constant regardless of any changes to $l$ or $\varepsilon$. By utilizing Equations (17) and (18), we can determine that the number of required signatures for an attack is solely dependent on $r$ and $n_{i} \mathrm{~s}$, provided that $l$ and $\varepsilon$ remain unchanged. These values are utilized in the calculation of $\mathcal{A}$. Moreover, if $r$ is fixed, it depends only on $n_{i}$ s. Therefore, the number of signatures
required for the attack depends on the value of $n_{i}$. Here, $n_{i}$ is represented by constraints such as $m_{i+1}=3 a_{i}+v_{i}-n_{i}$ and $\lambda-\ell_{\mathrm{FFT}}-f \leq \sum_{r=0}^{r-1} n_{i}$ as given in Table 1. $M$ is minimized if the equality holds. It is also multiplied by the square of the inverse of the modular bias. Considering each value in Table 2, we find that the number of signatures required increases significantly for high error rates.

The initial component of Equation (17) is referred to as the penalty term, which is always greater than 1 , except when all $\varepsilon_{j}$ values are zero. Moreover, as the value of $\varepsilon_{j}$ increases, this term also increases, ultimately leading to an increase in $M$. This aligns with our natural intuitions.

The value of the second term is determined solely by the parameter $l$. As $l$ increases, this term gradually decreases and approaches 1, but it always remains greater than 1 . As $l$ increases, the required signatures decrease, which intuitively makes sense. The penalty term prevents the second term from reducing $M$, and its significance increases with an increase in $\varepsilon_{j}$. However, it does not completely eliminate the possibility of the second term reducing $M$.

Combing Equation (17) with Equation (13), we can estiamte the contribution $j$-th MSB leakage. We can see that as $l$ becomes larger, $M$ will decrease, but its rate of decrease will be negligibly small.

## 4 Experimental results

### 4.1 Extension to multiple bit leakage with errors

Aranha et al. [3] have posted a script on GitHub [10] for solving linear programming problems based on Table 1. In this script, $\varepsilon$ is freely changeable. On the other hand, the number $l$ of nonce bits to leak is fixed to $l=1$. In a Fourier analysis-based attack, the leakage bit length and error rate affect only $\left|B_{q}(\boldsymbol{K})\right|$ in the constraints of Table 1. Therefore, we can easily obtain the script for multiple bits leakage by replacing the $\left|B_{q}(\boldsymbol{K})\right|$ evaluation equation for the [10] script with Equation (10).

We first naively optimize the number of signatures for multiple bit leakage with errors using a script with only $\left|B_{q}(\boldsymbol{K})\right|$ modifications. Figure 5 shows the optimal number of signatures for each $\varepsilon$ and $l$. Here, $\lambda=162, m_{\max }=40$, $\ell_{\mathrm{FFT}}=40, t_{\mathrm{max}}=80, r=2$. In addition, $\alpha$ depends only on the value of $\varepsilon$ because $L_{\mathrm{FFT}}$ is fixed.

### 4.2 Attack experiment

For 131-bit ECDSA, we recover the secret key when nonces have 1 bit leakage without error and when 2 bits leakage, each with an error rate of 0.1 . The computer used in the experiments has Intel Xeon Silver 4214R CPU $\times 2$ and 256 GB of DDR4 RAM. The parameters for the $l=1, \varepsilon=0$ and $l=2, \varepsilon=0.1$ cases are shown in Table 4. Table 5 shows the obtained $M^{\prime}=2^{m^{\prime}}=2^{m_{2}}$, mean value of bias and peak bias as a result of range reduction. In both cases,


Fig. 5. The number of signatures required for the extended [3] script.
the top 29 bits were successfully recovered. The experimental results show that $l=2, \varepsilon=0.1$ successfully recovers the secret key with a smaller bias value. Note that the error rate of 2 bits is 0.19 , since the error rate of each bit is 0.1 .

Table 4. Paramenters of attack experiment.

$$
\begin{array}{c|c|c|c|c|c|c} 
& a_{0} & a_{1} & v_{0} & v_{1} & n_{0} & n_{1} \\
\hline(l, \varepsilon)=(1,0) & 22 & 24 & 18 & 18 & 48 & 55 \\
\hline(l, \varepsilon)=(2,0.1) & 22 & 24 & 18 & 16 & 48 & 55
\end{array}
$$

From the experimental results, when $l=2, \varepsilon=0.1$, the secret key was successfully recovered with about $1 / 16$ of the number of samples after linear combination than when $l=1, \varepsilon=0$. Furthermore, the time required for range reduction is about 0.26 times smaller. Although the value of $M^{\prime}$ is changed by the parameter $v$ in this case, the number of signatures required can be changed by changing other parameters, and it can be inferred that the 2 bits leakage requires a smaller number of signatures.

Next, the parameters in Table 4 were changed to $a_{0}=20, a_{1}=23$, and $v_{1}=18$ to confirm the experiment in the case with errors. As a result, $M^{\prime}=2^{23.0}$,

Table 5. Experiment result.

|  | $M^{\prime}$ | Average noise | Peak bias | Range reduction time (sec) |
| :---: | :---: | :---: | :---: | :---: |
| $(l, \varepsilon)=(1,0)$ | $2^{28.1}$ | $1.5 \times 10^{-5}$ | $1.5 \times 10^{-4}$ | 5957 |
| $(l, \varepsilon)=(2,0.1)$ | $2^{22.1}$ | $2.0 \times 10^{-6}$ | $1.7 \times 10^{-5}$ | 1555 |

the time was 1984 seconds, and the secret key is successfully recovered. This shows that $l=2, \varepsilon=0.1$ can be recovered with fewer signatures and in less time than without errors.

## 5 Conclusion

We first evaluated the number of signatures by finding the formula of the modular bias for multiple bit leakage in the nonce. The modular bias as indicated by De Mulder et al. [8] and Aranha et al. [3] was extended to the case in which the MSBs of the nonce were leaked with multiple errors. We then proved Theorem 1. As the modular bias can now be calculated for any $l, \varepsilon$, we can now estimate the required number of signatures using a linear combination algorithm. In addition, the absolute value of the modular bias was given by Corollary 1. This corollary indicates that the error rate of the first MSB of the nonce has a greater effect on the modular bias than the error rates of the other bits. We then provided an estimate of the number of signatures required for various error rates.

We evaluated the number of signatures and computation time by obtaining the parameters of the 4 -list sum algorithm. Then, we performed an attack on 131-bit ECDSA with $l=2, \varepsilon=0.1$, and succeeded in recovering the secret key with fewer signatures with $l=1, \varepsilon=0$.

Acknowledgements This work was supported by JST CREST Grant Number JPMJCR2113 and JSPS KAKENHI Grant Number 21H03440.

## References

1. Albrecht, M.R., Heninger, N.: On bounded distance decoding with predicate: Breaking the "lattice barrier" for the hidden number problem. In: EUROCRYPT 2021. LNCS, vol. 12696, pp. 528-558. Springer (2021). https://doi.org/10.1007/978-3-030-77870-5\_19
2. Aranha, D.F., Fouque, P., Gérard, B., Kammerer, J., Tibouchi, M., Zapalowicz, J.: GLV/GLS decomposition, power analysis, and attacks on ECDSA signatures with single-bit nonce bias. In: ASIACRYPT 2014. LNCS, vol. 8873, pp. 262-281. Springer (2014). https://doi.org/10.1007/978-3-662-45611-8\_14
3. Aranha, D.F., Novaes, F.R., Takahashi, A., Tibouchi, M., Yarom, Y.: Ladderleak: Breaking ECDSA with less than one bit of nonce leakage. In: ACM CCS '20:. pp. 225-242 (2020). https://doi.org/10.1145/3372297.3417268
4. Bleichenbacher, D.: On the generation of one-time keys in dl signature schemes. In: Presentation at IEEE P1363 working group meeting. p. 81 (2000)
5. Boneh, D., Venkatesan, R.: Hardness of computing the most significant bits of secret keys in diffie-hellman and related schemes. In: CRYPTO '96. LNCS, vol. 1109, pp. 129-142. Springer (1996). https://doi.org/10.1007/3-540-68697-5\_11
6. Dinur, I.: An algorithmic framework for the generalized birthday problem. Des. Codes Cryptogr. 87(8), 1897-1926 (2019). https://doi.org/10.1007/s10623-018-00594-6
7. Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: An update. In: CT-RSA 2013. LNCS, vol. 7779, pp. 293-309. Springer (2013). https://doi.org/10.1007/978-3-642-36095-4 19
8. Mulder, E.D., Hutter, M., Marson, M.E., Pearson, P.: Using bleichenbacher's solution to the hidden number problem to attack nonce leaks in 384-bit ECDSA. In: CHES 2013. LNCS, vol. 8086, pp. 435-452 (2013). https://doi.org/10.1007/978-3-642-40349-1 \_25
9. Sun, C., Espitau, T., Tibouchi, M., Abe, M.: Guessing bits: Improved lattice attacks on (EC)DSA with nonce leakage. IACR TCHES 2022(1), 391-413 (2022). https://doi.org/10.46586/tches.v2022.i1.391-413
10. Takahashi, A.: ladderleak attack ecdsa (2020), https://github.com/ akiratk0355/ladderleak-attack-ecdsa/blob/master/tradeoffs-submit. ipynb
11. Takahashi, A., Tibouchi, M., Abe, M.: New bleichenbacher records: Fault attacks on qdsa signatures. IACR TCHES 2018(3), 331-371 (2018). https://doi.org/10.13154/tches.v2018.i3.331-371
12. Wagner, D.A.: A generalized birthday problem. In: CRYPTO 2002. LNCS, vol. 2442, pp. 288-303 (2002). https://doi.org/10.1007/3-540-45708-9 ${ }_{\text {_ }} 19$

## A Proof of Lemma 2

The proof for $b=0$ is as follows. Note that for simplicity, we denote $\exp \left(2 \pi \mathrm{i} k_{i} / q\right)$ by $\mathcal{E}_{q}\left(k_{i}\right)$.

$$
\begin{aligned}
B_{q}(\boldsymbol{K}) & =\boldsymbol{E}[\exp (2 \pi \mathrm{i} \boldsymbol{K} / q)]=\sum_{k_{i} \in \mathbb{Z}_{q}} \mathcal{E}_{q}\left(k_{i}\right) \cdot \operatorname{Pr}\left[\boldsymbol{K}=k_{i}\right] \\
& =\frac{\left(1-\varepsilon_{1}\right)\left(1-\varepsilon_{2}\right)}{q / 4} \sum_{k_{i} \in[0, q / 4)} \mathcal{E}_{q}\left(k_{i}\right)+\frac{\left(1-\varepsilon_{1}\right) \varepsilon_{2}}{q / 4} \sum_{k_{i} \in[q / 4, q / 2)} \mathcal{E}_{q}\left(k_{i}\right) \\
& +\frac{\varepsilon_{1}\left(1-\varepsilon_{2}\right)}{q / 4} \sum_{k_{i} \in[q / 2,3 q / 4)} \mathcal{E}_{q}\left(k_{i}\right)+\frac{\varepsilon_{1} \varepsilon_{2}}{q / 4} \sum_{k_{i} \in[3 q / 4, q)} \mathcal{E}_{q}\left(k_{i}\right) \\
& =\frac{\left(1-\varepsilon_{1}\right)\left(1-\varepsilon_{2}\right)}{q / 4} \sum_{k_{i} \in[0, q / 4)} \mathcal{E}_{q}\left(k_{i}\right)+\frac{\left(1-\varepsilon_{1}\right) \varepsilon_{2}}{q / 4} \sum_{k_{i}^{(1)} \in[0, q / 4)} \mathcal{E}_{q}\left(k_{i}^{(1)}+q / 4\right) \\
& +\frac{\varepsilon_{1}\left(1-\varepsilon_{2}\right)}{q / 4} \sum_{k_{i}^{(2)} \in[0, q / 4)} \mathcal{E}_{q}\left(k_{i}^{(2)}+q / 2\right)+\frac{\varepsilon_{1} \varepsilon_{2}}{q / 4} \sum_{k_{i}^{(3)} \in[0, q / 4)} \mathcal{E}_{q}\left(k_{i}^{(3)}+3 q / 4\right) \\
& =\frac{\left(1-\varepsilon_{1}\right)\left(1-\varepsilon_{2}\right)}{q / 4} \sum_{k_{i} \in[0, q / 4)} \mathcal{E}_{q}\left(k_{i}\right)+\mathrm{i} \frac{\left(1-\varepsilon_{1}\right) \varepsilon_{2}}{q / 4} \sum_{k_{i}^{(1)} \in[0, q / 4)} \mathcal{E}_{q}\left(k_{i}^{(1)}\right)
\end{aligned}
$$

$$
\begin{aligned}
& -\frac{\varepsilon_{1}\left(1-\varepsilon_{2}\right)}{q / 4} \sum_{k_{i}^{(2)} \in[0, q / 4)} \mathcal{E}_{q}\left(k_{i}^{(2)}\right)-\mathrm{i} \frac{\varepsilon_{1} \varepsilon_{2}}{q / 4} \sum_{k_{i}^{(3)} \in[0, q / 4)} \mathcal{E}_{q}\left(k_{i}^{(3)}\right) \\
& =\frac{\left(1-2 \varepsilon_{1}\right)\left(1-\varepsilon_{2}\right)}{q / 4} \sum_{k_{i} \in[0, q / 4)} \mathcal{E}_{q}\left(k_{i}\right)+\mathrm{i} \frac{\left(1-2 \varepsilon_{1}\right) \varepsilon_{2}}{q / 4} \sum_{k_{i} \in[0, q / 4)} \mathcal{E}_{q}\left(k_{i}\right) \\
& =\left\{\left(1-2 \varepsilon_{1}\right)\left(1-\varepsilon_{2}\right)+\mathrm{i}\left(1-2 \varepsilon_{1}\right) \varepsilon_{2}\right\} B_{q}\left(K_{b}\right)
\end{aligned}
$$

## B Visual explanation of the bias function for multi-bit leakage

In this appendix, Equation (9) is represented graphically. In Equation (9), at each $j,(1-\varepsilon)+\exp \left(2 \pi \mathrm{i} / 2^{j}\right)$ can be understood as a point in the complex plane with 1 and $\exp \left(2 \pi \mathrm{i} / 2^{j}\right)$ endowed by $1-\varepsilon: \varepsilon$. The endpoints in the complex plane at $j=1,2$ and $j=3,4$ in Figures 6 and 7 , respectively, are indicated by red dots. $\varepsilon$ and $1-\varepsilon$ in the figures represent ratios. When $j=1$, the two points 1 and -1 are endowed by $1-\varepsilon: \varepsilon$, and the red point is in the complex plane at coordinates $1-2 \varepsilon$. When $j=2$, we endow 1 and i , and when $j=3$, we endow 1 and $\exp (\pi \mathrm{i} / 4)$ with $1-\varepsilon: \varepsilon$.

As $j$ increases, $\exp \left(2 \pi \mathrm{i} / 2^{j}\right)$ approximates 1 . Therefore, $\exp \left(2 \pi \mathrm{i} / 2^{j}\right)$ and the interior point of 1 also approximates 1 . Figures 6 and 7 also show that the interior point approximates 1 in the complex plane. The absolute value also approximates 1.

Table 2 shows that as $l$ increases, the value does not readily increase. As explained in Section 3.3, this is because $\sin ^{2}\left(\pi / 2^{j}\right)$ is closer to 0 . This can also be observed in Figures 6 and 7.



Fig. 6. When the first and second bits Fig. 7. When the third and fourth bits leak. leak.

# Single Trace Analysis of Comparison Operation based Constant-Time CDT Sampling and Its Countermeasure 

Keon Hee Choi ${ }^{1[0009-0007-2557-566 X]}$, Ju-Hwan Kim ${ }^{1[0000-0001-8762-5553]}$, Jaeseung Han ${ }^{1[0000-0001-7111-2315]}$, Jae Won Huh ${ }^{1[0000-0002-6452-8002]}$, and Dong-Guk Han ${ }^{1,2}$ [0000-0003-1695-5103]<br>${ }^{1}$ Department of Financial Information Security, Kookmin University, Seoul, Republic of Korea<br>${ }^{2}$ Department of Information Security, Cryptology, and Mathematics, Kookmin University, Seoul, Republic of Korea<br>\{dy1637,zzzz2605,gjwodnjs987,jae1115, christa\}@kookmin.ac.kr


#### Abstract

Cumulative Distribution Table(CDT) sampling is a Gaussian sampling technique commonly used for extracting secret coefficients or core matrix values in lattice-based Post-Quantum Cryptography (PQC) algorithms like FrodoKEM and FALCON. This paper introduces a novel approach: a single trace analysis(STA) method for comparison operation based constant-time CDT sampling, as employed in SOLMAE-a candidate for Korean Post-Quantum Cryptography(KPQC) first-round digital signature Algorithm. The experiment is measuring power consumption during the execution of SOLMAE's sampling operation on an 8-bit AVR compiler microcontrollers unit(MCU) using ChipWhisperer-Lite. By utilizing STA, this paper recovered output of comparison operation based constant-time CDT sampling. The source of CDT sampling leakage is investigated through an in-depth analysis of the assembly code. The 8 -bit AVR MCU conducts comparison operations on values exceeding 8 bits by dividing them into 8 -bit blocks. Consequently, the execution time of a CDT sampling operation is influenced by the outcome of each block's comparison operation due to conditional branching. To address these concerns, this paper begins by summarizing trends in CDT sampling related research to design robust countermeasures against single trace analysis. Furthermore, a novel implementation method for comparison operation based constant-time CDT sampling against STA is proposed. This assembly-level implementation removes branching statements and performs comparative operations on all data words. Through experimental validation, this paper demonstrates the safety of the proposed countermeasure algorithm against STA.


Keywords: Side Channel Analysis • Single Trace Analysis • PQC . Gaussian sampling • CDT sampling • KPQC • SOLMAE • AVR

## 1 Introduction

The usage of public key cryptographic algorithms, such as Public-key Encryption(PKE)/Key Encapsulation Mechanism(KEM) and Digital Signature Algorithm(DSA), is widespread across various fields. However, it has been demonstrated that these algorithms will become vulnerable in the future due to the emergence of quantum computers and Shor's algorithm.[1,2] To address these security concerns, the National Institute of Standards and Technology (NIST) initiated the PQC standardization competition in 2016. The objective of this competition is to develop public-key cryptographic algorithms that can resist attacks from quantum computers. Currently, a subround is in progress following the final round of the competition. Additionally, as part of the competition, new algorithms are being proposed that build upon the shortlisted and selected algorithms. The competition was divided into two main areas for public-key cryptography, namely PKE/KEM and Digital Signature. Importantly, numerous lattice-based algorithms have been proposed in both areas. In these lattice-based cryptographic algorithms, important values are extracted from the Gaussian distribution, and the method employed to extract them using a table is known as CDT sampling. In other words, CDT sampling is a crucial role in lattice-based algorithms.

There are many ways to implement CDT sampling. The first proposed CDT sampling has been analyzed using the technique proposed by [3], resulting in the proposal of constant-time CDT sampling. This constant-time CDT sampling was implemented using subtraction in FrodoKEM and Lizard. Additionally, [4,5] proposed STA for CDT sampling. then secret value of FrodoKEM was leaked. Repeatedly, CDT sampling is very important. In this paper, we study in detail the security of side channel analysis for comparison operation based CDT sampling in MITAKA [6] and SOLMAE [7], which are a similar structure of the Falcon. Importantly, the security of side channel analysis for these comparison operation based CDT sampling techniques has not been studied before this work. This paper recovery the sampling value of CDT sampling through STA for vulnerability that is variable the operating time of CDT sampling depending on the results of comparative operations in 8 -bit AVR MCU. To validate this vulnerability, the paper employs ChipWhisperer-Lite to measure power consumption during CDT sampling on the Atmel XMEGA-128, using the AVR compiler for the 8 -bit processor. Additionally, using assembly code root cause analysis, the paper proposes a secure constant-time CDT sampling method using comparison operations to counter STA.

### 1.1 Contribution

This paper addresses the safety of comparison operation based constant-time CDT sampling from a side-channel analysis perspective, which has not been previously studied. In addition, by analyzing the power consumption traces used in SOLMAE, we identified the basesampler in the overall cryptographic operation
algorithm. This increases the feasibility of the STA in this paper. So, this paper describes the reason for vulnerability in comparison operation-based CDT sampling in great detail. Experiments have confirmed that CDT sampling in 8 -bitAVR MCU varies in operating time depending on comparison operation results. The cause analysis was performed using an assembly code. In the 8-bit AVR MCU, during CDT sampling operations, when comparing values larger than 8 bits, the process is divided into 8 -bit units. The analysis reveals that the operation concludes the moment the result is determined, resulting in a change in execution time. In essence, not all blocks undergo comparison operations, and this behavior is closely associated with the presence of branch statements.

A novel STA is propose for comparison operation based CDT sampling. Additionally, a new CDT sampling implementation method is propose to resist side-channel analysis, contributing to the development of secure algorithms for CDT sampling. The practical implementation removes the branch statements from the assembly code and presents a structure where all blocks can be compared. Experimental verification demonstrates the resistance to STA through power consumption trace analysis.

### 1.2 Organization

The remainder of this paper introduces STA for CDT sampling through a total of five sections. In Section 2, it provides detailed explanation of lattice, LWE, and NTRU, emphasizing the significance of Gaussian sampling. This highlights the importance of CDT sampling, the two implementation methods, and the imperative need to investigate the security of comparison based CDT sampling, which has not been previously explored. Moving on to Section 3, it presents the experimental setup and target implementation. Section 4 delves into a side channel analysis of CDT sampling based on comparison operations. Here, it detailed describe the application method of STA and the cause of the vulnerability. In Section 5, we present the implementation of CDT sampling in which vulnerabilities are mitigated through an analysis of the underlying causes. To demonstrate their resistance against the attack technique proposed in this paper, it collect actual a power consumption trace. Finally, Section 6 addresses conclusions and future research directions.

## 2 Backgrounds

In this section, we provide an introductory overview of lattice-based cryptography[8], LWE, and NTRU encryption schemes. Following that, we delve into the Gaussian distribution and proceed to describe CDT sampling, which is of paramount importance as a module. We then elaborate on timing side-channel analysis conducted on the original CDT sampling, followed by an in-depth description of a STA of the subtraction operation based constant-time CDT sampling.

### 2.1 Lattice

Definition 1. Lattice: Given $n$ linearly independent vectors $\mathbf{b}_{\mathbf{1}}, \mathbf{b}_{\mathbf{2}}, \ldots, \mathbf{b}_{\mathbf{n}} \in$ $\mathbb{R}^{m}$, the lattice $\mathcal{L}\left(\mathbf{b}_{\mathbf{1}}, \mathbf{b}_{\mathbf{2}}, \ldots, \mathbf{b}_{\mathbf{n}}\right)$ is defined as the set of all linear combinations of $\mathbf{b}_{\mathbf{1}}, \mathbf{b}_{\mathbf{2}}, \ldots, \mathbf{b}_{\mathbf{n}}$ with integer coefficients, i.e.,

$$
\mathcal{L}\left(\mathbf{b}_{\mathbf{1}}, \mathbf{b}_{\mathbf{2}}, \ldots, \mathbf{b}_{\mathbf{n}}\right)=\left\{\sum_{i=1}^{n} x_{i} \mathbf{b}_{\mathbf{i}} \mid x_{i} \in \mathbb{Z}\right\}
$$

We refer to $\mathbf{b}_{\mathbf{1}}, \mathbf{b}_{\mathbf{2}}, \ldots, \mathbf{b}_{\mathbf{n}}$ as a basis of the lattice.
Equivalently, if we define $\mathbf{B}$ as the $m \times n$ matrix whose columns are $\mathbf{b}_{\mathbf{1}}, \mathbf{b}_{\mathbf{2}}, \ldots, \mathbf{b}_{\mathbf{n}}$, then the lattice generated by $\mathbf{B}$ is

$$
\mathcal{L}(\mathbf{B})=\left\{\mathbf{B} \mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^{n}\right\}
$$

### 2.2 NTRU and LWE

The first public key cipher based on Lattice was proposed by A.M. in 1997, and since then, various studies have been conducted to create efficient encryption algorithms [9]. In Lattice-based encryption, efficiency primarily refers to speed and key size. NTRU, proposed by Hoffstein et al. in 1996 [10], is known for its fast encryption process. Falcon, MITAKA, and SOLMAE are examples of NTRU-based encryption algorithms [6,11,12].

Definition 2. NTRU: Let $q$ be a positive integer, and $z(x) \in \mathbb{Z}$ be a monic polynomial. Then, a set of NTRU secrets consists of four polynomials $f, g, F, G \in R_{q}$ which satisfy the NTRU equation:

$$
f G-g F \equiv q \quad \bmod z(x)
$$

And define $h$ as $h \leftarrow g \cdot f^{-1} \bmod q$. Then, given $h$, find $f$ and $g$.
LWE was proposed by Regev in 2005 [13]. LWE is known to be NP-hard, even when adding small values of noise. CRYSTAL-KYBER and CRYSTALDilithium are examples of LWE-based cryptographic algorithms [12,14].

Definition 3. LWE: Let $n$ and $q$ be positive integers, and let $\chi$ be a distribution over $\mathbb{Z}$. For a vector $s \in \mathbb{Z}_{q}^{n}$, the LWE distribution $\mathcal{A}_{s, \chi}$ over $\mathbb{Z}_{q}^{n} \times \mathbb{Z}_{q}$ obtained by choosing $a \in \mathbb{Z}_{q}^{n}$ and an integer error $e$ from $\chi$. The distribution returns the pair $(a,\langle a, s\rangle+e \bmod q)$.

There are two important concepts of LWE.

- Search-LWE problem: Given $m$ independent samples $\left(a_{i}, b_{i}\right) \in \mathbb{Z}_{q}^{n} \times \mathbb{Z}_{q}$ drawn from $\mathcal{A}_{s, \chi}$, find $s$.
- Decision-LWE problem: Given $m$ independent samples $\left(a_{i}, b_{i}\right) \in \mathbb{Z}_{q}^{n} \times \mathbb{Z}_{q}$, distinguish whether each sample is drawn from the uniform distribution or from $\mathcal{A}_{s, \chi}$.


### 2.3 Discrete Gaussian Distribution

In this paper, CDT sampling is the method to extract random values from Gaussian distribution. Prior to CDT sampling, the definition of the discrete Gaussian distribution on the lattice is given as follows.

Definition 4. Discrete Gaussian Distribution over Lattice: Let $\forall c \in$ $\mathbb{R}^{n}, \sigma \in \mathbb{R}^{+}$,

$$
\forall x \in \mathbb{R}^{n}, \rho_{\sigma, c}(x)=\exp \left(\frac{-\pi\|x-c\|^{2}}{\sigma^{2}}\right) .
$$

Then, for $\forall c \in \mathbb{R}^{n}, \sigma \in \mathbb{R}^{+}, n$-dimensional lattice $\mathcal{L}$, define the Discrete Gaussian Distribution over $\mathcal{L}$ as:

$$
\forall x \in \mathcal{L}, \mathcal{D}_{L, \sigma, c}(x)=\frac{\rho_{\sigma, c}(x)}{\rho_{\sigma, c}(\mathcal{L})}
$$

### 2.4 CDT Sampling

Some lattice-based schemes based on LWE extract the error from a Gaussian distribution. Similarly, certain lattice-based schemes based on NTRU create essential values from a Gaussian distribution. CDT sampling is an efficient method for extracting values from these Gaussian distributions, and ensuring the security of such CDT sampling is of utmost importance. The CDT table stores specific probability values of the Gaussian distribution. CDT sampling is an algorithm that randomly generates probability values and determines the range within which the generated values fall among those stored in the table. The value to be sampled at this point corresponds to the determined index. There are several ways to implement CDT sampling, and this paper deals with the safety study of implementing CDT sampling based on comparison operations.

```
Algorithm 1 The CDT sampling vulnerable to timing attack
    Input : CDT table \(\Psi, \sigma, \tau\)
    Output: Sampled value \(S\)
    \(r n d \leftarrow[0, \tau \sigma) \cap \mathbb{Z}\) uniformly at random
    sign \(\leftarrow[0,1] \cap \mathbb{Z}\) uniformly at random
    \(i \leftarrow 0\)
    while \((r n d>\Psi[i])\) do
        \(i++\)
    end while
    \(S \leftarrow((-\operatorname{sign}) \wedge i)+\operatorname{sign}\)
    return \(S\)
```

The initially proposed CDT sampling Algorithm 1 was found to be vulnerable to the timing attack proposed by [3]. This vulnerability arises due to the
different timing of the while loop termination. As a remedy, constant-time CDT sampling utilizes for statements. There are two ways to implement this: CDT sampling based on comparison operations and CDT sampling based on subtraction operations.

```
Algorithm 2 The subtraction operation based CDT sampling
    Input : CDT table \(\Psi\) of length \(\ell, \sigma, \tau\)
    Output : Sampled value \(S\)
    \(r n d \leftarrow[0, \tau \sigma)\) uniformly at random
    sign \(\leftarrow[0,1] \cap \mathbb{Z}\) uniformly at random
    \(S \leftarrow 0\)
    for \(i=0\) to \(\ell-1\) do
        \(S+=(\Psi[i]-r n d) \gg 63\)
    end for
    \(S \leftarrow((-\operatorname{sign}) \wedge S)+\) sign
    return \(S\)
```

Both methods are available for schemes that use CDT sampling. However, only subtraction based CDT sampling has been suggested to be vulnerable. Algorithm 2 is an example of subtraction operation based CDT sampling. LWE-based lattice-based schemes commonly employ this algorithm [15,16]. Additionally, it has been proposed to perform STA by the power differences between negative and positive numbers [4,5]. Moreover, an attack to find the secret key of a cryptographic algorithm has been proposed using this method.

```
Algorithm 3 The comparison operation based CDT sampling: half-Gaussian table
access CDT
    Input : CDT table \(\Psi\) of length \(\ell, \sigma, \tau\)
    Output : Sampled value \(S\)
    \(r n d \leftarrow[0, \tau \sigma)\) uniformly at random
    sign \(\leftarrow[0,1] \cap \mathbb{Z}\) uniformly at random
    \(S \leftarrow 0\)
    for \(i=0\) to \(\ell-1\) do
        \(S+=(r n d \geq \Psi[i])\)
    end for
    \(S \leftarrow((-\operatorname{sign}) \wedge S)+\) sign
    return \(S\)
```

On the other hand, NTRU-based lattice-based schemes often utilize CDT sampling based on comparison operations, especially in [6,7,17] which employs hybrid sampling. Algorithm 3 is the comparison based CDT sampling. Unlike
conventional methods, it performs sampling from Gaussian distribution using comparison operations.

## 3 Experiment Setup

In this section, the experimental environment and the CDT sampling code employed in the STA experiments are described. The C code of SOLMAE, a candidate from the KPQC Round 1 digital signature category, was implemented in the AtmelXMEGA128 environment. Power consumption traces were collected during the operation for analysis.

### 3.1 Implimentation of comparison operation based CDT sampling

The BaseSampler function implemented in SOLMAE and MITAKA employs a comparison operation based CDT sampling approach. Thus, this paper utilizes the reference code of SOLMAE, which was proposed as a candidate for KPQC Round 1 digital signature. Specifically, our focus is on the BaseSampler function within the code. The sampling technique in SOLMAE follows the sampling outlined in [17] and employs a table to generate values from a half-Gaussian distribution. The BaseSampler function is illustrated in Listing 1.1. The CDT table contains 13 values arranged in ascending order, which are sequentially compared against the randomly selected value " r " from the reference code.

```
int base_sampler()
{
    uint64_t r = get64(); //get randomly 64 bits from RNG.
    int res = 0;
    for (int i = 0; i < TABLE_SIZE; i++)
        res += (r >= CDT[i]);
    return res;
}
```

Listing 1.1: BaseSampler function C code

### 3.2 Target Device of Experiment

The board utilized in this paper consists of an AtmelXMEGA128 (8-bit processor) and Chipwhisperer-Lite. The AtmelXMEGA128 is an 8-bit AVR MCU. The BaseSampler function implemented in SOLMAE operates on the AtmelXMEGA128 board, while Chipwhisperer-Lite is employed to collect the power consumption data during the BaseSampler function operation Figure 1.

The experimental steps conducted in this paper are as follows:

- Collection of power consumption data during the comparison operationbased CDT sampling.
- Analysis of the assembly language, considering different compiler optimization levels, to identify vulnerabilities in the comparison operations.
- Investigation of comparison operation vulnerabilities using real-world traces.
- Acquisition of output values for the newly proposed CDT sampling algorithm through STA.


Fig. 1: AtmelXMEGA128(8-bit AVR MCU) and Chipwhisperer-Lite

This paper demonstrates that vulnerable implementations of comparison operations, which could be realistic in a commercialized environment, can expose the actual values of CDT sampling. Furthermore, a CDT sampling algorithm resistant to side-channel attacks is proposed.

## 4 Side Channel Analysis of Comparison Operation based Constant-Time CDT Sampling

### 4.1 Description of the Cause of Vulnerability

The security of comparison operations heavily depends on the specific implementation technique and enviroment like compiler. Let us consider the comparison of two multi-word numbers, denoted as $A$ and $B$ in Figure 2.

Various methods can be employed to compare these numbers. One common approach is to initiate the comparison with the most significant words. Compare $A$ and $B$ as follows:

1. Check if $A_{0}$ is greater than $B_{0}$. If so, $A>B$.
2. Check if $A_{0}$ is less than $B_{0}$. If true, $A<B$.
3. Check if $A_{0}$ and $B_{0}$ are equal. If true, continue to compare the next word until the comparison ends.


Fig. 2: The comparison of two multi-word numbers, denoted as A and B. A and $B$ are each 8 blocks.

This implementation is vulnerable to side channel analysis. For instance, let's consider two scenarios: (1) $A_{0}>B_{0}$ and (2) $A_{0}=B_{0}, A_{1}<B_{1}$. In these situations, the execution time of the comparison operations may differ. As a result, timing vulnerabilities arise, which can be exploited through STA to distinguish between the two scenarios. Therefore, a comparison algorithm resistant to STA is required.

```
<base_sampler>:
    ...
    24c: ldi r22, 0x00
    24e: ldi r23, 0x00
    250: ldd r24, Z+7
    252: cp r25, r24
    254: brcs .+74 ; 0x2a0 <base_sampler+0x92>
    256: cp r24, r25
    258: brne .+66 ; 0x29c <base_sampler+0x8e>
    25a: ldd r24, Z+6
    25c: cp r20, r24
    25e: brcs .+64 ; 0x2a0 <base_sampler+0x92>
    260: cp r24, r20
    262: brne .+56 ; 0x29c <base_sampler+0x8e>
    ...
    29c: ldi r22, 0x01 ; 1
    29e: ldi r23, 0x00 ; 0
    2a0: add r18, r22
    2a2: adc r19, r23
```

    ...
    Listing 1.2: Base Sampler() assembly code

In this section, the vulnerabilities associated with various implementations of weak comparison operations are explored. The assembly code of the BaseSampler
function used in SOLMAE is examined for various optimization levels (Level: $0,1,2,3, s)$ provided by the AtmelXMEGA128. The assembly code depicted in Listing 1.2 illustrates the part of BaseSampler function for the optimized s-level. It is evident that the comparisons are performed sequentially, word by word. Notably, vulnerabilities in the word based comparison method are evident. The process of performing comparison operations for each optimization level follows a similar pattern as shown in Listing 1.2. Subsequent instructions are dependent on the results of the word comparisons, leading to variations in executed operations and resulting in distinct power consumption patterns manifested as differences in power traces.

In more detail, the first word is compared in lines 252, 254, and the next operation varies depending on the result. First, calculate $r 25-r 24$. If a carry occurred, then branch to line 2a0. This indicates that $r 24$ was a greater number than $r 25$. If no carry has occurred, go to lines 256,258 . Then, calculate $r 24-$ $r 25$. If the values are not the same between $r 24$ and $r 25$, branch to line 29c. This means that $r 25$ was a greater number than $r 24$. If the values were the same, compare the next two words by executing the following lines. Repeat this process until the comparison operation is finally completed. In other words, the vulnerability appears in the fact that the processing method in the branch statement varies depending on the result of the comparison operation. This is an important point to understand for design of countermeasure.

### 4.2 Single Trace Analysis on the Comparison Operation based Constant-time CDT Sampling



Fig. 3: The power consumption trace of maximum r on uint $64 \_t$

The BaseSampler function utilized in SOLMAE implements CDT sampling through comparison operations, as depicted in Listing 1.1. The comparison operations are performed between two operands of the uint64_t data type: a random variable $r$ and each the 13 values stored in the CDT table. On an 8-bit processor, these comparison operations are performed by dividing them into 8 words. The
aforementioned comparison operations have two vulnerabilities. First, the number of comparisons depends on the values being compared. Second, the value being added depends on the result of each comparison operation, i.e., an additional operation is required to add 1 . Therefore, it is risky to work with data types larger than word.

Figure 3 shows the power consumption trace of the CDTsampling when $r$ is set to the maximum value of the uint $64 \_t$ data type (i.e., $2^{64}-1$ ). From the power consumption trace, it is evident that the number of comparisons with each CDT table differs, indicating variations in computation time based on the compared values.


Fig. 4: Two power consumption traces differ by only one in sampling values. They differ by only one in $r$ values

Figure 4 shows two power consumption traces with only a difference of 1 in the values of 'r.' More precisely, the return values, sampled by the difference in ' $r$ ' values, also differ by one. The noted discrepancy is a result of the optional addition operation, leading to evident distinctions between the two traces. This is also related to the data type of the resulting value returned. Since the returned data type is a unit larger than the word, a difference also occurs in the addition operation.These discrepancies in power consumption traces enable the visual detection of any divergence in assembly instructions.

An increment of 1 of the sampling result occurs when $r$ is greater than or equal to value of table in the comparison between $r$ and the value of table.

Furthermore, the values in the CDT table are arranged in ascending order. Consequently, once $r$ becomes smaller than a particular value in the CDT table, the resulting value remains unchanged. This implies that if a comparison operation with a CDT table value greater than $r$ is identified, the output of CDT sampling can be obtained. The power consumption traces of the first word in the comparison operation, as depicted in Figure 5, exhibit distinct shapes for the scenarios where $r$ is greater than, equal to, and less than the value in the CDT table, respectively. The visual distinctiveness of these power traces facilitates the acquisition of the CDT sampling value. This vulnerability arises from the inherent characteristics of the weak comparison operation, as discussed earlier.


Fig. 5: The power consumption traces of CDT sampling have different shapes for each r value: (a) $A_{0}<B_{0}$, (b) $A_{0}=B_{0}$, and (c) $A_{0}>B_{0}$ where $A_{i}$ and $B_{i}$ represent individual words.

## 5 Countermeasure

In the previous section, we highlighted the vulnerability of comparison operations when processing data larger than the word size of the processor. To address this issue and ensure the safety of comparison operation based constant-time CDT sampling, we propose a novel implementation method with countermeasure.

Before introducing the proposed countermeasure, we first provide an overview of trends in countermeasures related to CDT sampling. First, in [4] the CDT sampling method using Table was proposed. But it requires a large storage space.

In addition, there is also the protection of sampling through the masking method proposed by [18] and the random shuffling method proposed by [19,20]. However, But these have memory overheads and time overheads. And analysis techniques related to these are being proposed.[21]. However, since there have not been many studies related to sampling using comparison operation, a new concept of implementing CDT sampling using comparison operation has been attempted.

In previous sections, the cause of vulnerability mentioned in this paper were attributed to the varying number of clock cycles depending on the branch statement in the 8-bit AVR MCU environment. Hence, the countermeasure proposed an implementation method that eliminates the discrepancy in the number of clock cycles. The proposed secure CDT sampling algorithm in this paper is denoted as Algorithm 4. The algorithm processes the $r$ and the CDT table in word-sized blocks, corresponding to the processing units of the processor. The values in $r$, CDT table that exceed the word size are divided into $n$ word blocks. Comparison operations are performed identically each block. However, if the outcome of a comparison operation is determined in the previous block, subsequent operations are only performed, i.e., it does not affect the result. Due to the inherent nature of comparison operations, methods employing them may result in branching. Branching commands such as 'brne' and 'brcc' are commonly used. In AVR instruction sets, 'brne' and 'brcc' differ by only 1 with respect to true and false conditions, allowing for an equal adjustment in the number of clock cycles for the operation. However, this implementation approach can be considered risky. Therefore, this paper introduces an assembly code that effectively eliminates the need for branch commands while implementing Algorithm 4.

```
Algorithm 4 STA-Resistant CDT sampling
    Input : -
    Output : Sampled value \(z\)
    \(z \leftarrow 0\)
    \(r_{i} \stackrel{\&}{\leftarrow}\left[0,2^{\text {word size }}\right)\) uniformly random with \(i=0\) to \(n\)
    for \(i=0\) to Table_size -1 do
        \(g t \leftarrow 0, l t \leftarrow 0\)
        for \(j=0\) to \(n-1\) do
            \(g t \mid=(\neg(g t \mid l t)) \&\left(r_{j}>C D T_{i, j}\right)\)
            \(l t \mid=(\neg(g t \mid l t)) \&\left(r_{j}<C D T_{i, j}\right)\)
        end for
        \(z+=1 \oplus l t\)
    end for
    return \(z\)
```

```
<STA-Resistant CDT sampling>:
    278: ldi r18, 0x00 ; 0
    27a: cp r22, r23
    27c: adc r18, r18
    27e: and r24, r18
    280: or r19, r24
    282: mov r24, r19
    284: or r24, r25
    286: com r24
    288: ldi r18, 0x00 ; 0
    28a: cp r23, r22
    28c: adc r18, r18
```

Listing 1.3: The comparison operation of assembly implementation code of countermeasure


Fig. 6: The traces that overlap all three types of STA-Resident CDT sampling. And (a) $A_{0}<B_{0}$, (b) $A_{0}=B_{0}$, and (c) $A_{0}>B_{0}$ where $A_{i}$ and $B_{i}$ represent individual words.

Listing 1.3 is a parts of the assembly code, representing the comparison operation in the proposed countermeasure. The blue and red lines in Listing 1.3 correspond to the comparison operations in Algorithm 4. Lines 278 and 288 initialize the value of register r18, where the result of the comparison operation
will be stored, to zero. Lines 27 a and 28 a perform comparisons between registers r22 and r23 using 'cp' commands, respectively, and store the results in the carry flag. Lines 27 c and 28 c execute an addition operation on the initialized r18 using the 'adc' (add with carry) instruction. During this operation, the stored carry values are combined, resulting in the storage of the comparison operation's result within r18. This approach allowed me to eliminate the need for branching instructions, thus removing the vulnerabilities previously mentioned.

Figure 6 illustrates the power consumption traces of 3 different types of the Listing 1.3 operating in the 8 -bit AVR MCU. The power consumption traces (a), (b), and (c), which are fully examined by overlapping with a , b , and c , represent the corresponding power consumption traces. Similar to Figure 5, (a), (b), and (c) signify whether the most significant block of ' $r$ ' is greater than, equal to, or less than the value in the CDT table. The trace reveals that there are no discernible variations in the comparison time across different values. This serves as compelling evidence that CDT sampling demonstrates resistance against STA.

## 6 Conclusion and Futurework

This paper introduces a secure implementation of CDT sampling for Gaussian sampling techniques. CDT sampling is used by many algorithms to generate important values. And this paper presents an analysis of a previously unexplored vulnerability that STA in comparison operation-based CDT sampling. This paper identifies a vulnerability in which the operation time varies depending on the results of the comparison operation in 8-bit AVR MCU. The cause of the vulnerability was demonstrated through different of the number of instruction at the assembly stage. It was investigated that it was a vulnerability due to the difference in the number of clocks.

The feasibility of extracting CDT sampling outputs in real-world environments, such as AtmelXMEGA128, is demonstrated. AtmelXMEGA128 is an 8 -bit AVR MCU and is used in various environments. We also employed different compiler options ( $0,1,2,3, s$ ) provided by Chipwhisperer in the AtmelXMEGA128 environment and verified the presence of the vulnerability across all of them. In this paper, we utilized the example of compiler option level 's,' which is set as the default among several available options. In this paper, we did not show power consumption traces for other options, as we observed that all options exhibited the same or even greater leakage. In addition, this paper deals with vulnerabilities that depend on the processor's word size and compiler. During our investigation, we observed that the number of clock cycles varied depending on the branch instruction employed. It also showed the impact of the attack by recovering the sampling value. This finding sheds light on the potential risks associated with future cryptographic algorithms that employ CDT sampling with vulnerable comparison operations, using SOLMAE as a case study. We conducted an analysis of power consumption traces to pinpoint the sections of the SOLMAE algorithm utilizing CDT sampling. This demonstrated the practical applicability of STA.

To address these concerns, a robust CDT sampling design is proposed, ensuring security against STA in real-world. To address these issues, our proposed countermeasure for CDT sampling in this paper aims to stabilize the number of clock cycles, irrespective of the branch statement used. So, First we delved into the countermeasure algorithms for CDT sampling that were previously explored. Our investigation revealed the existence of algorithms employing tablebased comparison operations, masking methods, and shuffling techniques. And we present a method for implementing comparison operation based constanttime CDT sampling, designed to mitigate the security risks associated with the previously proposed STA. The algorithm is crafted to segment and store data in units processed by the processor, facilitating comparisons across all blocks. This design allows for sampling without reliance on the results of comparison operations.

In real-world implementations, caution is warranted branch statements. Branch statements, such as 'brne' and 'brcc' commands in 8-bit AVR MCU, introduce variability in clock cycles depending on the outcome of comparison operations. If the result of the branch leads to a distant address, the number of clock cycles will vary based on the outcome. In essence, it is the need for caution in employing branch statements. To address this variability, we propose a comparison operation based constant-time CDT sampling implementation method at the actual assembly code level. Instead of using branch statements, the results of comparison operations are stored in the result register using instructions that 'cp' and 'adc'. This approach ensures uniform operation time without relying on the specific outcome of the comparison operation. Additionally, this paper showed the power consumption traces using Chipwhisperer-Lite when operating proposed countermeasure algorithm in AtmelXMEGA128(8-bit AVR MCU) to demonstrate safety against STA.

The experimental environment of this paper is 8 -bit AVR MCU. In the future, we plan to investigate the possibility of STA for comparison operation based constant-time CDT sampling in various environments.

## References

1. P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM review, vol. 41, no. 2, pp. 303-332, 1999.
2. M. Mosca, "Cybersecurity in an era with quantum computers: Will we be ready?," IEEE Security \& Privacy, vol. 16, no. 5, pp. 38-41, 2018.
3. P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in Advances in Cryptology-CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18-22, 1996 Proceedings 16, pp. 104-113, Springer, 1996.
4. S. Kim and S. Hong, "Single trace analysis on constant time cdt sampler and its countermeasure," Applied Sciences, vol. 8, no. 10, p. 1809, 2018.
5. S. Marzougui, I. Kabin, J. Krämer, T. Aulbach, and J.-P. Seifert, "On the feasibility of single-trace attacks on the gaussian sampler using a cdt," in International Workshop on Constructive Side-Channel Analysis and Secure Design, pp. 149-169, Springer, 2023.
6. T. Espitau, P.-A. Fouque, F. Gérard, M. Rossi, A. Takahashi, M. Tibouchi, A. Wallet, and Y. Yu, "Mitaka: a simpler, parallelizable, maskable variant of falcon," in Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 222-253, Springer, 2022.
7. K. Kim, M. Tibouchi, A. Wallet, T. Espitau, A. Takahashi, Y. Yu, and S. Guilley, "Solmae algorithm specifications." https://kpqc.or.kr/1, 2020.
8. O. Regev, "Lecture notes of lattices in computer science, taught at the computer science tel aviv university," 2009.
9. M. Ajtai and C. Dwork, "A public-key cryptosystem with worst-case/average-case equivalence," in Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, pp. 284-293, 1997.
10. J. Hoffstein, J. Pipher, and J. H. Silverman, "Ntru: A ring-based public key cryptosystem," in International algorithmic number theory symposium, pp. 267-288, Springer, 1998.
11. P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, Z. Zhang, et al., "Falcon: Fast-fourier latticebased compact signatures over ntru," Submission to the NIST's post-quantum cryptography standardization process, vol. 36, no. 5, pp. 1-75, 2018.
12. S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Silder, and D. Stehlé, "Crystals-dilithium: Algorithm specifications and supporting documentation," 2020.
13. O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," Journal of the ACM (JACM), vol. 56, no. 6, pp. 1-40, 2009.
14. R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-kyber algorithm specifications and supporting documentation," NIST PQC Round, vol. 2, no. 4, pp. 1-43, 2019.
15. J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila, "Frodo: Take off the ring! practical, quantum-secure key exchange from lwe," in Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pp. 1006-1018, 2016.
16. J. H. Cheon, D. Kim, J. Lee, and Y. Song, "Lizard: Cut off the tail! a practical postquantum public-key encryption from lwe and lwr," in International Conference on Security and Cryptography for Networks, pp. 160-177, Springer, 2018.
17. J. Howe, T. Prest, T. Ricosset, and M. Rossi, "Isochronous gaussian sampling: From inception to implementation: With applications to the falcon signature scheme," in Post-Quantum Cryptography: 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings 11, pp. 53-71, Springer, 2020.
18. T. Schneider, C. Paglialonga, T. Oder, and T. Güneysu, "Efficiently masking binomial sampling at arbitrary orders for lattice-based crypto," in Public-Key Cryptography-PKC 2019: 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part II 22, pp. 534-564, Springer, 2019.
19. D. E. Knuth, Art of computer programming, volume 2: Seminumerical algorithms. Addison-Wesley Professional, 2014.
20. R. A. Fisher and F. Yates, Statistical tables for biological, agricultural and medical research. Hafner Publishing Company, 1953.
21. K. Ngo, E. Dubrova, Q. Guo, and T. Johansson, "A side-channel attack on a masked ind-cca secure saber kem implementation," IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 676-707, 2021.

# A Lattice Attack on CRYSTALS-Kyber with Correlation Power Analysis 

Yen-Ting Kuo (®) (D) and Atsushi Takayasu (D)<br>The University of Tokyo, Tokyo, Japan<br>\{kuruwakuo,takayasu-a\}@g.ecc.u-tokyo.ac.jp


#### Abstract

CRYSTALS-Kyber is a key-encapsulation mechanism, whose security is based on the hardness of solving the learning-with-errors (LWE) problem over module lattices. As in its specification, Kyber prescribes the usage of the Number Theoretic Transform (NTT) for efficient polynomial multiplication. Side-channel assisted attacks against PostQuantum Cryptography (PQC) algorithms like Kyber remain a concern in the ongoing standardization process of quantum-computer-resistant cryptosystems. Among the attacks, correlation power analysis (CPA) is emerging as a popular option because it does not require detailed knowledge about the attacked device and can reveal the secret key even if the recorded power traces are extremely noisy. In this paper, we present a two-step attack to achieve a full-key recovery on lattice-based cryptosystems that utilize NTT for efficient polynomial multiplication. First, we use CPA to recover a portion of the secret key from the power consumption of these polynomial multiplications in the decryption process. Then, using the information, we are able to fully recover the secret key by constructing an LWE problem with a smaller lattice rank and solving it with lattice reduction algorithms. Our attack can be expanded to other cryptosystems using NTT-based polynomial multiplication, including Saber. It can be further parallelized and experiments on simulated traces show that the whole process can be done within 20 minutes on a 16 -core machine with 200 traces. Compared to other CPA attacks targeting NTT in the cryptosystems, our attack achieves lower runtime in practice. Furthermore, we can theoretically decrease the number of traces needed by using lattice reduction if the same measurement is used. Our lattice attack also outperforms the state-of-the-art result on integrating sidechannel hints into lattices, however, the improvement heavily depends on the implementation of the NTT chosen by the users.


Keywords: CRYSTALS-Kyber, lattice, side-channel attack, number theoretic transform

## 1 Introduction

### 1.1 Background

With the development of quantum computation, what is usually hard to solve on the traditional computer (factorization, DLP, etc) will become efficiently solvable

Y.T. Kuo, A. Takayasu

by applying Shor's algorithm [26], which will make the public-key cryptosystems most people use now unreliable. Thus, there is a significant interest in postquantum cryptography (PQC) algorithms, which are based on mathematical problems presumed to resist quantum attacks. To standardize such algorithms, the National Institute of Standards and Technology (NIST) initiated a process to solicit and evaluate PQC candidates being submitted [22]. After three rounds of the process, they had identified four candidate algorithms for standardization and four more to be evaluated in round 4.

CRYSTALS-Kyber (Kyber) [2] is one out of the four candidates that are confirmed to be standardized in July, 2022, and it is the only public-key encryption and key-establishment algorithm. It belongs to the category of lattice-based cryptography, and in particular a module Learning With Errors (module-LWE) scheme. Kyber prescribes the usage of the Number Theoretic Transform (NTT) for efficient polynomial multiplication. Via point-wise multiplication of transformed polynomials, i.e., $a b=\mathrm{NTT}^{-1}(\mathrm{NTT}(a) \circ \mathrm{NTT}(b))$, multiplication can be performed in time $O(n \log n)$, where $n$ is the degree of polynomial $a$ and $b$. Kyber has three parameter sets: Kyber512, Kyber768 and Kyber1024 with security level similar to that of AES128, AES192 and AES256.

Power analysis attacks, introduced by Kocher [15,16], exploit the fact that the instantaneous power consumption of a cryptographic device depends on the data it processes and on the operation it performs. There exist simple power analysis attacks on Kyber that can compromise a message or private key using only one or several traces. In particular, Primas et al. [24] and Pessl et al. [23] recover data passed through an NTT by templating the multiplications or other intermediate values within the NTT. Hamburg et al. [13] present a sparse-vector chosen ciphertext attack strategy, which leads to full long-term key recovery. These attacks are still limited in that they either require extensive profiling efforts or they are only applicable in specific scenarios like the encryption of ephemeral keys.

As opposed to above methods, Mujdei et al. [21] showed that leakage from the schoolbook polynomial multiplications after the incomplete NTT can be exploited through correlation power analysis (CPA) style attacks. CPA attacks exploit the dependency of power consumption on intermediate values, we provide an introduction of CPA attacks below and refer to work of Mangard et al. [18] for further details. The presented attack required 200 power traces to recover all the coefficients, which enables full key recovery. More precisely, they guess two coefficients at once within the range $\left(-\frac{q}{2}, \frac{q}{2}\right]$, implying a search over $q^{2}$ combinations.

In order to model the effect of these side-channel leakage, Dachman-Soled et al. [8] proposed a general lattice framework that quantifies the LWE security loss when revealing a so-called hint $(\mathbf{v}, \mathbf{w}, l) \in \mathbb{Z}_{q}^{n} \times \mathbb{Z}_{q}^{m} \times \mathbb{Z}$ satisfying

$$
\langle(\mathbf{v}, \mathbf{w}),(\mathbf{s}, \mathbf{e})\rangle=l .
$$

The inner product of this equation is usually performed in $\mathbb{Z}_{q}$, which is referred to as modular-hint. They also dealt with leakage $l$ before $\bmod q$ reduction, a
so-called perfect hint. Their results was later improved by May and Nowakowski [19], where they only addressed hints for the secret $\mathbf{s}$ only, i.e., hints ( $\mathbf{v}, l)$ with $\langle\mathbf{v}, \mathbf{s}\rangle=l$.

### 1.2 Our Contribution

In this paper, we propose a way that utilizes correlation power analysis to fully recover the secret key of Kyber. Our attack consists of two steps. First, by exploiting the correlation of Hamming weight of some intermediates and the power consumption of the decryption process in Kyber, precisely the part where we multiply the secret polynomial with ciphertext, we can recover some of the coefficients of the secret key in the NTT domain. Secondly, since there will be some ambiguity about whether the recovered coefficients are indeed correct, we sample part of the recovered coefficients and construct a lattice problem by Kannan's embedding proposed by [14]. Then one can recover the entire secret key by solving the lattice problem by using lattice reduction algorithms such as BKZ [5].

We also examined the attack on simulated traces of ARM cortex-M0 generated by a toolkit named ELMO [10]. Experiments show that we can indeed recover the secret key with 200 traces. With some fine-tuning on the acceptance threshold of power analysis, we can even have guaranteed success in sampling all correct coefficients with 600 traces and still have enough ones to construct a solvable lattice problem.

There are three parameter sets for Kyber, and our attack can be easily adapted to all parameter sets. The time it takes to recover the secret key is linear to the number of coefficients in the secret key. The power analysis part of our attack can be parallelized to further accelerate the process. Although the idea of our attack is similar to that of Mujdei et al.[21], we only require $O(q)$ search, which directly reflects on the runtime of the CPA. For reference, our attack is about 16 times faster than Mujdei et al. [21] without parallelization. Since our SCA and that of [21] use different methods of measurement, it is hard to compare the result. However, if we use the same measurement, by using the lattice reduction, we can theoretically decrease the number of required power traces. It may get some wrong coefficients by doing so, but we can fix that by sampling portion of recovered coefficients and using lattice reduction to find the rest of them.

For the lattice attack part of our attack, as opposed to the above methods, our approach uses divide-and-conquer methods in a way that we only consider a portion of the secret key at a time. That is, the hints $\langle\mathbf{v}, \mathbf{s}\rangle=l$ gathered from our method are inner products of vectors with smaller dimension. This can be done because in the computation of decryption of Kyber, the secret key is divided into blocks by the intrinsic property of module-LWE. Furthermore, the NTTs of each sub-key are usually incomplete since it can achieve fastest speed in that way [6]. Due to these properties of Kyber, The techniques of Dachman-Soled et al. [8] and May et al. [19] to solve the LWE instance are not suitable for our cases. Since we only consider a portion of the secret key at a time. The number
of hints we need is extremely lower than their methods. However, we do need to perform multiple times of the lattice reduction to achieve a full key recovery.

Our lattice attack can be applied to other cryptosystems that utilizes NTTbased polynomial multiplications. For example, Saber [9] is a lattice-based KEM based on Module Learning With Rounding problem. Although it is not specifically designed to use NTT by choosing an NTT-friendly ring, it is still possible to achieve fast computation by NTT by enlarging the ring as shown in the work by Chung et al. [6]. However, the improvement from our attack depends on the implementation of the NTT, namely how many layers of NTT the implementation chooses to apply to it.

We use the official reference implementation of the Kyber key encapsulation mechanism provided by the authors [3] as the target. We also provide an efficient open-source Python implementation of our framework. The source code is available at https://github.com/kuruwa2/kyber-sca.
Organization. The rest of this paper is organized as follows. In Section 2, we introduce how Kyber is implemented with Number Theoretic Transform. In Section 3, we illustrate how to apply differential power analysis to the NTT part of Kyber. In Section 4, we construct a simpler lattice problem from the recovered coefficients and conduct an experiment by lattice reduction algorithm to determine the least number of coefficients we need to recover from differential power analysis. In Section 5, we analyze the success rate of our attack and conclude the paper.

## 2 Preliminaries

In this section, we explain the lattices and module-learning with errors problem, go into some details about Kyber, and review the Number Theoretic Transform.

### 2.1 Lattices

Let $\mathbf{B}=\left[\mathbf{b}_{1}, \ldots, \mathbf{b}_{n}\right] \in \mathbb{Z}^{m \times n}$ be an integer matrix. We denote by

$$
\Lambda(\mathbf{B}):=\left\{\alpha_{1} \mathbf{b}_{1}+\ldots+\alpha_{n} \mathbf{b}_{n} \mid \alpha_{i} \in \mathbb{Z}\right\}
$$

the lattice generated by $\mathbf{B}$. If the rows of $\mathbf{B}$ are linearly independent, $\mathbf{B}$ is a basis matrix of $\Lambda(\mathbf{B})$. The number of rows $n$ in any basis matrix of some lattice $\Lambda$ is called the rank of $\Lambda$. The determinant of a lattice $\Lambda$ with basis matrix $\mathbf{B}$ is defined as

$$
\operatorname{det}(\Lambda):=\sqrt{\operatorname{det}\left(\mathbf{B B}^{T}\right)}
$$

The determinant does not depend on the choice of basis. We also denote by $\lambda_{i}(\Lambda)$ the $i$-th successive minimum of $\Lambda$. A lattice vector $\mathbf{v} \in \Lambda$ such that $\|\mathbf{v}\|=\lambda_{1}(\Lambda)$ is called the shortest vector of $\Lambda . \lambda_{1}(\Lambda)$ can be estimated by the following heuristic.
Heuristic 1 (Gaussian Heuristic) Let $\Lambda$ be an n-dimensional lattice. Gaussian heuristic predicts that the norm of the shortest vector $\lambda_{1}(\Lambda)$ equals

$$
g h(\Lambda):=\sqrt{\frac{n}{2 \pi e}} \operatorname{det}(\Lambda)^{1 / n} .
$$

### 2.2 Module-LWE

Learning with errors (LWE) problem [25] and its extension over rings [17] or modules are the basis of multiple NIST PQC candidates.

Let $\mathbb{Z}_{q}$ be the ring of integers modulo $q$ and for given power-of- 2 degree $n$, define $\mathcal{R}_{q}=\mathbb{Z}_{q}[x] /\left(x^{n}+1\right)$ as the polynomial ring of polynomials modulo $x^{n}+1$. For any ring $\mathcal{R}, \mathcal{R}^{\ell \times k}$ denotes the ring of $\ell \times k$-matrices over $\mathcal{R}$. We also simplify $\mathcal{R}^{\ell \times 1}$ to $\mathcal{R}^{\ell}$ if there is no ambiguity. Single polynomials are written without markup, vectors are bold lower case a and matrices are denoted with bold upper case A. $\beta_{\eta}$ denotes the centered binomial distribution with parameter $\eta$ and $\mho$ denote the uniform distribution. If $\chi$ is a probability distribution over a set $S$, then $x \leftarrow \chi$ denotes sampling $x \in S$ according to $\chi$. If $\chi$ is only defined on $\mathbb{Z}_{q}$, $x \leftarrow \chi\left(\mathcal{R}_{q}\right)$ denotes sampling the polynomial $x \in \mathcal{R}_{q}$, where all coefficients of the coefficients in $x$ are sampled from $\chi$.

The learning with errors (LWE) problem was introduced by Regev [25] and its decision version states that it is hard to distinguish $m$ uniform random samples $\left(\mathbf{a}_{i}, b_{i}\right) \leftarrow \mho\left(\mathbb{Z}_{q}^{n} \times \mathbb{Z}_{q}\right)$ from $m$ LWE-samples of the form

$$
\left(\mathbf{a}_{i}, b_{i}=\mathbf{a}_{i}^{\top} \mathbf{s}+e_{i}\right) \in \mathbb{Z}_{q}^{n} \times \mathbb{Z}_{q},
$$

where the secret vector $\mathbf{s} \leftarrow \beta_{\eta}\left(\mathbb{Z}_{q}^{n}\right)$ is fixed for all samples, $\mathbf{a}_{i} \leftarrow \mho\left(\mathbb{Z}_{q}^{n}\right)$ and $e_{i} \leftarrow \beta_{\eta}\left(\mathbb{Z}_{q}\right)$ is a small error. A module version of LWE, called Mod-LWE [4] essentially replaces the ring $\mathbb{Z}_{q}$ in the above samples by a quotient ring of the form $\mathcal{R}_{q}$ with corresponding error distribution $\beta_{\eta}\left(\mathcal{R}_{q}\right)$.

$$
\left(\mathbf{a}_{i}, b_{i}=\mathbf{a}_{i}^{\top} \mathbf{s}+e_{i}\right) \in \mathcal{R}_{q}^{k \times 1} \times \mathcal{R}_{q} .
$$

The rank of the module is $k$ and the dimension of the $\operatorname{ring} \mathcal{R}_{q}$ is $n$. The case $k=1$ corresponds to the ring-LWE problem introduced in [17]. We also commonly integrate $m$ number of samples by the matrix multiplication,

$$
(\mathbf{A}, \mathbf{b}=\mathbf{A} \mathbf{s}+\mathbf{e}) \in \mathcal{R}_{q}^{m \times k} \times \mathcal{R}_{q}^{m} .
$$

Let $\lambda_{i}(\Lambda)$ denote the $i$-th minimum of lattice $\Lambda$. The LWE problem can be considered as an average version of the Bounded Distance Decoding (BDD) problem: Given a vector such that its distance from the lattice is at most $\lambda_{1}(\Lambda) / 2$, the goal is to find the closest lattice vector to it. A dual problem of BDD is the so-called unique Shortest Vector Problem (uSVP): Given $\gamma \geq 1$, and lattice $\Lambda$ such that $\lambda_{2}(\Lambda) \geq \gamma \cdot \lambda_{1}(\Lambda)$, the goal is to find a non-zero vector $\mathbf{v} \in \Lambda$ of norm $\lambda_{1}(\Lambda)$. The reduction between LWE, BDD, and USVP will be further discussed in Section 4.2.

### 2.3 CRYSTALS-Kyber

Kyber [2] is a Key Encapsulation Mechanism (KEM) submitted to the NIST standardization process, and it is among the four confirmed candidates to be standardized [22]. The security of Kyber is based on the module-LWE problem. For the three parameter sets in the proposal, Kyber512, Kyber768, and

Table 1: Parameter sets for Kyber [1].

| name | $n$ | $k$ | $q$ | $\eta_{1}$ | $\eta_{2}$ |
| :--- | :---: | :---: | :---: | :---: | :---: |
| Kyber512 | 256 | 2 | 3329 | 3 | 2 |
| Kyber768 | 256 | 3 | 3329 | 2 | 2 |
| Kyber1024 | 256 | 4 | 3329 | 2 | 2 |

Kyber1024, the parameters are all set to $n=256$ and $q=3329$. For most parameters $\eta=2$ is used, except for Kyber512, where $\eta=3$. The parameter sets differ in their module dimension $k=2,3$, and 4 respectively. The three parameter sets listed in Table 1.

Kyber consists of the CCA2-KEM Key Generation, PKE- and CCA2-KEMEncryption, and CCA2-KEM-Decryption algorithms, which are summarized in Algorithms 1, 2, 3 and 4, respectively.

```
Algorithm 1 Kyber-CCA2-KEM Key Generation (simplified)
    Output: Public key \(p k\), secret key \(s k\)
    Choose uniform seeds \(\rho, \sigma, z\)
    \(\mathcal{R}^{k \times k} \ni \hat{\mathbf{A}} \leftarrow\) Sample \(_{\vartheta}(\rho)\)
    \(\mathcal{R}_{q}^{k} \ni \mathbf{s}, \mathbf{e} \leftarrow\) Sample \(_{\beta_{\eta}}(\sigma)\)
    \(\hat{\mathbf{s}} \leftarrow \mathrm{NTT}(\mathbf{s})\)
    \(\hat{\mathbf{t}} \leftarrow \hat{\mathbf{A}} \circ \hat{\mathbf{s}}+\operatorname{NTT}(\mathbf{e})\)
    \(\operatorname{return}(p k:=(\hat{\mathbf{t}}, \rho), s k:=(\hat{\mathbf{s}}, p k, \operatorname{Hash}(p k), z))\)
```

```
Algorithm 2 Kyber-PKE Encryption (simplified)
    Input: Public key \(p k=(\hat{\mathbf{t}}, \rho)\), message \(m\), seed \(\tau\)
    Output: Ciphertext \(c\)
    \(\mathcal{R}^{k \times k} \ni \hat{\mathbf{A}} \leftarrow\) Sample \(_{\mho}(\rho)\)
    \(\mathcal{R}_{q}^{k} \ni \mathbf{r}, \mathbf{e}_{1}, \mathcal{R}_{q} \ni e_{2} \leftarrow\) Sample \(_{\beta_{\eta}}(\tau)\)
    \(\mathbf{u} \leftarrow \operatorname{NTT}^{-1}\left(\hat{\mathbf{A}}^{\top} \circ \operatorname{NTT}(\mathbf{r})\right)+\mathbf{e}_{1}\)
    \(v \leftarrow \operatorname{NTT}^{-1}\left(\hat{\mathbf{t}}^{\top} \circ \operatorname{NTT}(\mathbf{r})\right)+e_{2}+\operatorname{Encode}(m)\)
    return \(c:=(\mathbf{u}, v)\)
```

In these algorithms, and in the rest of this paper, the notation $a \circ b$ means pairwise multiplication of polynomials, or vectors of polynomials, in the NTT domain. For example, if $a=\left(a_{0}, a_{1}\right)$ and $b=\left(b_{0}, b_{1}\right), a \circ b=\left(a_{0} b_{0}, a_{1} b_{1}\right)$.

Kyber uses a variant of the Fujisaki-Okamoto transform [11] to build an INDCCA2 secure KEM scheme. This transform applies an additional re-encryption of the decrypted message, using the same randomness as used for the encryp-
tion of the received ciphertext. The decryption is only valid if the re-computed ciphertext matches the received ciphertext.

```
Algorithm 3 Kyber-CCA2-KEM Encapsulation (simplified)
    Input: Public key \(p k=(\hat{\mathbf{t}}, \rho)\)
    Output: Ciphertext \(c\), shared key \(K\)
    Choose uniform \(m\)
    \((\bar{K}, \tau) \leftarrow \operatorname{Hash}(m \| \operatorname{Hash}(p k))\)
    \(c \leftarrow \operatorname{PKE} . \operatorname{Enc}(p k, m, \tau)\)
    \(K \leftarrow \operatorname{KDF}(\bar{K} \| \operatorname{Hash}(c))\)
    return \((c, K)\)
```

```
Algorithm 4 Kyber-CCA2-KEM Decapsulation (simplified)
    Input: Secret key \(s k=(\hat{\mathbf{s}}, p k, h, z)\), ciphertext \(c=(\mathbf{u}, v)\)
    Output: Shared key \(K\)
    \(m \leftarrow \operatorname{Decode}\left(v-\operatorname{NTT}^{-1}\left(\hat{\mathbf{s}}^{\top} \circ \operatorname{NTT}(\mathbf{u})\right)\right)\)
    \((K, \tau) \leftarrow \operatorname{Hash}(m \| h)\)
    \(c^{\prime} \leftarrow \operatorname{PKE} \cdot \operatorname{Enc}(p k, m, \tau)\)
    if \(c=c^{\prime}\) then
        return \(K:=\operatorname{KDF}(K \| \operatorname{Hash}(c))\)
    else
        return \(K:=\operatorname{KDF}(z \| \operatorname{Hash}(c))\)
    end if
```


### 2.4 Number Theoretic Transform

For lattice-based schemes using polynomial rings, polynomial multiplications in en-/decryption are the most computationally expensive step. The Number Theoretic Transform (NTT) is a technique that can achieve efficient computation for those multiplications.

The NTT is similar to the Discrete Fourier Transform (DFT), but instead of over the field of complex numbers, it operates over a prime field $\mathbb{Z}_{q}$. It can be seen as a mapping between the coefficient representation of a polynomial from $\mathcal{R}_{q}$ (called the normal domain) to the evaluation of the polynomial at the $n$-th roots of unity (called the NTT domain). This bijective mapping is typically referred to as forward transformation. The mapping from the NTT domain to the normal domain is referred to as backward transformation or inverse NTT. In the NTT domain, the multiplication of polynomials can be achieved by pointwise multiplication, which is much cheaper than multiplication in the normal domain. Typically, one would perform the forward transformation, multiply the


Fig. 1: 8-coefficient Cooley-Tukey decimation in time NTT
polynomials pointwisely in the NTT domain, and go back using the backward transformation. For $\mathcal{R}_{q}$ with a $2 n$-th primitive root of unity $\zeta$, the NTT transformation of an $n$-degree polynomial $f=\sum_{i=0}^{n-1} f_{i} x^{i}$ is defined as:

$$
\hat{f}=\operatorname{NTT}(f)=\sum_{i=0}^{n-1} \hat{f}_{i} x^{i}, \quad \text { where } \quad \hat{f}_{i}=\sum_{j=0}^{n-1} f_{j} \zeta^{(2 i+1) \cdot j} .
$$

Similarly,

$$
\begin{aligned}
& f=\operatorname{NTT}^{-1}(\hat{f})=\sum_{i=0}^{n-1} f_{i} x^{i}, \quad \text { where } \\
& f_{i}=n^{-1} \sum_{j=0}^{n-1} \hat{f}_{j} \zeta^{-i \cdot(2 j+1)} .
\end{aligned}
$$

The NTT transform and its inverse can be applied efficiently by using a chaining of $\log _{2} n$ butterflies. It is a divide and conquer technique that splits the input in half in each step and solves two problems of size $n / 2$. The construction for an 8-coefficient NTT using the Cooley-Tukey butterfly [7] with decimation in time is depicted in Figure 1, with the output being in bit-reversed order. Notice that both NTT and inverse NTT are a linear transform, thus they can be expressed by matrix multiplications, e.g. $\left[f_{i}\right]^{\top}=\mathbf{M}\left[\hat{f}_{i}\right]^{\top}$ for some $n \times n$ matrix M.

Kyber uses an NTT-friendly ring. But in Kyber, only $n$-th primitive roots of unity exist, therefore the modulus polynomial $x^{n}+1$ only factors into polynomials of degree 2. Hence, the last layer between nearest neighbors of the NTT is skipped and in NTT domain multiplication is not purely point-wise, but multiplications of polynomials of degree 1 . That is, the Kyber ring is effectively $\mathbb{F}_{q^{2}}[y] /\left(y^{128}+1\right)$, where $\mathbb{F}_{q^{2}}$ is the field $\mathbb{Z}_{q}[x] /\left(x^{2}-\zeta\right)$. Also note that in Kyber, polynomials in the NTT domain are always considered in bit-reversed order (cf. Figure 1). Therefore, in the following bit-reversal is implicitly expected in the NTT domain and indices for NTT-coefficients are noted in regular order.

## 3 Correlation Power Analysis

In this section, we provide a comprehensive introduction to correlation power analysis (CPA) provided by Mangard et al. [18] in Section 3.1, and then we apply the idea to reveal the secret key of Kyber in Section 3.2.

The goal of CPA is to reveal secret keys of cryptographic devices based on a large number of power traces that have been recorded while the devices encrypt or decrypt different plaintexts. The probability of success for CPA depends on the quality and number of traces. Due to the fact that CPA does not require detailed knowledge about the attacked devices, it is the most popular type of power analysis attack. Furthermore, they can reveal the secret key even if the recorded power traces are extremely noisy.

### 3.1 General Description

We now discuss in detail how such an analysis reveals the secret keys of cryptographic devices in five steps. To reveal one coefficient we need to apply the five steps, however, step 2 can be applied only once and the power consumption can be used multiple time for each coefficient that needs to be recovered.
Step 1: Choosing an Intermediate Result of the Executed Algorithm. The first step of a CPA is to choose an intermediate result of the cryptographic algorithm that is executed by the device. This intermediate value needs to be a function $f(d, k)$, where $d$ is a known non-constant data value and $k$ is a small part of the key. In most attack scenarios, $d$ is either the plaintext or the ciphertext.
Step2: Measuring the Power Consumption. The second step of a CPA is to measure the power consumption of the device while it encrypts or decrypts $D$ different data blocks. For each of these encryption or decryption runs, the attacker needs to know the corresponding data value $d$ that is involved in the calculation of the intermediate result chosen in Step 1. We denote these known data values by vector $\mathbf{d}=\left(d_{1}, \ldots, d_{D}\right)^{\top}$, where $d_{i}$ denotes the data value in the $i$-th encryption or decryption process.

During each of these runs, the attacker records a power trace. We denote the power trace that corresponds to data block $d_{i}$ by $\mathbf{t}_{i}^{\top}=\left(t_{i, 1}, \ldots, t_{i, T}\right)$, where $T$ denotes the length of the trace. The attacker measures a trace for each of the $D$ data blocks, and hence, the traces can be written as matrix $\mathbf{T}$ of size $D \times T$.

It is important that the measured traces are correctly aligned. This means that the power consumption values of each column $t_{j}$ of the matrix $\mathbf{T}$ need to be caused by the same operation. In practice, attackers typically try to measure only the power consumption that is related to the targeted intermediate result. If the plaintext is known, the attacker sets the trigger of the oscilloscope to the sending of the plaintext from the PC to the cryptographic device and records the power consumption for a short period of time.
Step 3: Calculating Hypothetical Intermediate Values. The next step of the attack is to calculate a hypothetical intermediate value for every possible choice of $k$. We write these possible choices as vector $\mathbf{k}=\left(k_{1}, \ldots, k_{K}\right)$, where $K$ denotes the total number of possible choices of $k$. In the context of CPA, we
usually refer to the elements of this vector as key hypotheses. Given the data vector $\mathbf{d}$ and the key hypotheses $\mathbf{k}$, an attacker can easily calculate hypothetical intermediate values $f(d, k)$ for all $D$ en-/decryption runs and for all $K$ key hypotheses. This calculation results in a matrix $\mathbf{V}$ of size $D \times K$.

$$
\mathbf{V}=\left[f\left(d_{i}, k_{j}\right)\right]_{D \times K}
$$

A $j$-th column of $\mathbf{V}$ contains the intermediate results that have been calculated based on the key hypothesis $k_{j}$. It is clear that one column of $\mathbf{V}$ contains those intermediate values that have been calculated in the device during the $D$ en$/$ decryption runs because $\mathbf{k}$ contains all possible choices for $k$. We refer to the index of this element as $c k$. Hence, $k_{c k}$ refers to the key of the device. The goal of CPA is to find out which column of $\mathbf{V}$ has been processed during the $D$ en/decryption runs. We immediately know $k_{c k}$ as soon as we know which column of $\mathbf{V}$ has been processed in the attacked device.
Step 4: Mapping Intermediate Values to Power Consumption Values. The next step of a CPA is to map the hypothetical intermediate values $\mathbf{V}$ to a matrix $\mathbf{H}$ of hypothetical power consumption values. For this purpose, the attacker typically uses models like Hamming-weight model or Hamming-distance model depending on the scenarios of attack. Using the techniques, the power consumption of the device for each hypothetical intermediate value $v_{i, j}$ is simulated in order to obtain a hypothetical intermediate value $h_{i, j}$.

The quality of the simulation strongly depends on the knowledge of the attacker about the analyzed device. The better the simulation of the attacker matches the actual power consumption characteristics of the device, the more effective the CPA is. The most commonly used power models to map $\mathbf{V}$ to $\mathbf{H}$ are the Hamming-distance and Hamming-weight models.
Step 5: Comparing the Hypothetical Power Consumption Values with the Power Traces. After having mapped $\mathbf{V}$ to $\mathbf{H}$, the final step of a CPA can be performed. In this step, each column $\mathbf{h}_{i}$ of the matrix $\mathbf{H}$ is compared with each column $\mathbf{t}_{j}$ of the matrix $\mathbf{T}$. This means that the attacker compares the hypothetical power consumption values of each key hypothesis with the recorded traces at every position. The result of this comparison is a matrix $\mathbf{R}$ of size $K \times T$, where each element $r_{i, j}$ contains the result of the comparison between the columns $\mathbf{h}_{i}$ and $\mathbf{t}_{j}$. The comparison is done based on the Pearson correlation coefficient,

$$
r_{i, j}=\frac{\sum_{d=1}^{D}\left(h_{d, i}-\bar{h}_{i}\right) \cdot\left(t_{d, j}-\bar{t}_{j}\right)}{\sqrt{\sum_{d=1}^{D}\left(h_{d, i}-\bar{h}_{i}\right)^{2} \cdot \sum_{d=1}^{D}\left(t_{d, j}-\bar{t}_{j}\right)^{2}}}
$$

where $\bar{h}_{i}$ and $\bar{t}_{j}$ denote the mean values of the columns $\mathbf{h}_{i}$ and $\mathbf{t}_{j}$. It has the property that the value $r_{i, j}$ is the higher, the better columns $\mathbf{h}_{i}$ and $\mathbf{t}_{j}$ match. The key of the attacked device can hence be revealed based on the following observation.

The power traces correspond to the power consumption of the device while it executes a cryptographic algorithm using different data inputs. The intermediate result that has been chosen in step 1 is a part of this algorithm. Hence, the device
needs to calculate the intermediate value $\mathbf{v}_{c k}$ during the different executions of the algorithm. Consequently, also the recorded traces depend on these intermediate values at some position. We refer to this position of the power traces as $c t$, i.e., the column $\mathbf{t}_{c t}$ contains the power consumption values that depend on the intermediate value $\mathbf{v}_{c k}$.

The hypothetical power consumption values $\mathbf{h}_{c k}$ have been simulated by the attacker based on the values $\mathbf{v}_{c k}$. Therefore, the columns $\mathbf{h}_{c k}$ and $\mathbf{t}_{c t}$ are strongly related. In fact, these two columns lead to the highest value in $\mathbf{R}$, i.e., the highest value of the matrix $\mathbf{R}$ is the value $r_{c k, c t}$. An attacker can hence reveal the index for the correct key ck and the moment of time ct by simply looking for the highest value in the matrix $\mathbf{R}$. The indices of this value are then the result of the CPA.

Sometimes, CPA produce high correlation coefficients for many key hypotheses at the time when targeted intermediate result is processed. The high correlation peaks for wrong keys are sometimes referred to as ghost peaks. These peaks happen because the hypothetical intermediate values are correlated. The height of these correlations depends on the intermediate result that is attacked.

### 3.2 Application on CRYSTALS-Kyber

Our attack targets the decryption process of Kyber, i.e. line 1 of Algorithm 4 , with the aim of recovering the victim's secret key $\hat{\mathbf{s}}$. To decrypt a message the recipient calculates $\operatorname{NTT}^{-1}\left(\hat{\mathbf{s}}^{\top} \circ \hat{\mathbf{u}}\right)$, where $\hat{\mathbf{u}}$ is the decompressed ciphertext in the NTT domain and o denotes the pairwise multiplication. The pairwise multiplication is done in the quotient ring $\mathbb{Z}_{q}[x] /\left(x^{2}-\zeta_{i}\right)$ as we discussed in Section 2.4, where $\zeta_{i}$ are the primitive roots of unity of $\mathbb{Z}_{q}$. In such a ring, the product of two polynomials $a=a_{0}+a_{1} x$ and $b=b_{0}+b_{1} x$ can be easily computed as

$$
a b=\left(a_{0} b_{0}+a_{1} b_{1} \zeta_{i}\right)+\left(a_{0} b_{1}+a_{1} b_{0}\right) x \bmod q .
$$

However, in most of the processors, modular multiplication is still expensive since it needs divisions by $q$. Fortunately, we can avoid the divisions by the Montgomery reduction algorithm summarized in Algorithm 5. By setting $R=$ $2^{16}$, division by $R$ can be replaced by a simple bit shifting and $x \bmod R$ can be done by returning the lower 16 bits of $x$, which results in an integer between $-R / 2$ and $R / 2-1$. The algorithm works because first, $t$ is chosen so that $a-t q$ is divisible by $R$. Second, $t$ is in the range $[-R / 2, R / 2-1]$, thus $a-t q$ is in the range $[-q R+q, q R-1]$, which guarantees that $b$ is in the correct range.

Let $x_{0}$ and $y_{0}$ be two integers in the range $[-q+1, q-1]$, we refer to the result of Montgomery reduction of $x_{0} \times y_{0}$ by Algorithm 5 as $\operatorname{fqmul}\left(x_{0}, y_{0}\right)$. Then the

```
Algorithm 5 Montgomery reduction
    Input: Integers \(q, R\) with \(\operatorname{gcd}(q, R)=1\)
                Integer \(q^{-1} \in[-R / 2, R / 2-1]\) such that \(q q^{-1} \equiv 1 \bmod R\)
            Integer \(a \in[-q R / 2, q R / 2-1]\)
    Output: Integer \(b \in[-q+1, q-1]\) such that \(b \equiv a R^{-1} \bmod q\)
    \(t \leftarrow\left((a \bmod R) q^{-1}\right) \bmod R\)
    \(b \leftarrow(a-t q) / R\)
    return b
```

product $r_{0}+r_{1} x=a b 2^{-16}$ can be computed as follow:

$$
\begin{align*}
& r_{0} \leftarrow \text { fqmul }\left(a_{1}, b_{1}\right) \\
& r_{0} \leftarrow \operatorname{fqmul}\left(r_{0}, \zeta_{i} 2^{16}\right) \\
& r_{0} \leftarrow \operatorname{fqmul}\left(a_{0}, b_{0}\right)+r_{0}  \tag{1}\\
& r_{1} \leftarrow \operatorname{fqmul}\left(a_{1}, b_{0}\right) \\
& r_{1} \leftarrow \operatorname{fqmul}\left(a_{0}, b_{1}\right)+r_{1}
\end{align*}
$$

The unwanted constant can be dealt within the inverse NTT together when we divide the coefficient by $n$, thus no extra multiplications is needed.

Now suppose we want to reveal the coefficients $\left(\hat{s}_{2 i}, \hat{s}_{2 i+1}\right)$, notice that they are point-wisely multiplied by the ciphertext ( $\hat{u}_{2 i}, \hat{u}_{2 i+1}$ ), then our first chosen intermediate value is $\mathrm{fqmul}\left(\hat{s}_{2 i+1}, \hat{u}_{2 i+1}\right)$, i.e. $r_{0}$ in the first line of equation (1). The intermediate value meets the requisite described in Section 3.1, and the total number of possible choices of $\hat{s}_{2 i+1} \in[0, q-1]$ is $q$. Following the steps in Section 3.1, we can get a list of the most possible candidates of $\hat{s}_{2 i+1}$. There can be some incorrect candidates with high score in this step, for example, $q-\hat{s}_{2 i+1}$ can be such a candidate since the Hamming weight of $\operatorname{fqmul}\left(q-\hat{s}_{2 i+1}, \hat{u}_{2 i+1}\right)$ is strongly correlated with $\mathrm{fqmul}\left(\hat{s}_{2 i+1}, \hat{u}_{2 i+1}\right)$.

Now that we have some highly confident candidates for $\hat{s}_{2 i+1}$, we can then use it and newly guessed $\hat{s}_{2 i}$ to calculate the hypothetical value of $r_{1}$. And we can repeat the same process except that the intermediate values are now $\mathrm{fqmul}\left(\hat{s}_{2 i}, \hat{u}_{2 i+1}\right)+\mathrm{fqmul}\left(\hat{s}_{2 i+1}, \hat{u}_{2 i}\right)$, i.e. $r_{1}$ in the last line of equation 1 . Following the same steps, we can find the candidate with the highest correlation coefficient, and if it is higher than some threshold, we accept the guess. If not, we try the next candidate of $\hat{s}_{2 i+1}$. If there is no candidate with high enough correlation coefficient, we just return failure. Then we guess the next one with same process targeting the next intermediate values.

The complexity can be easily calculated, if $K$ is the number of possible keys, $T$ is the scanned window size, $D$ is the number of power traces, then we need $T K$ computations of correlation coefficient of length $D$ vectors to recover one coefficient of the secret key, which is linear to all the parameters. For Kyber512, we need to repeat the process above 256 times to recover the 512 coefficients in the NTT domain. The CPA process is identical across different parameter sets of Kyber, thus it is easy to adapt to Kyber768/1024 without any problem. It can also be parallelized as long as we know the starting point of each fqmul in
the power trace, since the length of all power traces is the same, we only need to evaluate the starting point once and store the result. For Kyber512 on a 16 core computer, our CPA can scan through all coefficients within 5 minutes.

However, we will run into some problems. If the correct coefficient $\left(\hat{s}_{2 i}, \hat{s}_{2 i+1}\right)$ has high score, then it is likely that $\left(q-\hat{s}_{2 i}, q-\hat{s}_{2 i+1}\right)$ has high score too, since the Hamming weight of them are highly correlated. So to prevent it from getting accepted, we can increase the threshold for acceptance, however, it may cause the correct ones to get rejected too. Furthermore, in some rare cases, $\left(q-\hat{s}_{2 i}, q-\hat{s}_{2 i+1}\right)$ may have a higher score than the correct one and be accepted, we call such cases false positive. The way we deal with it is to sample the accepted guesses and hope the coefficients we sampled are all correct ones. The number of sampled coefficients will be further discussed in Section 4.

## 4 Lattice Attack

In this section, we describe how to construct a simpler LWE problem from the coefficients that have been recovered in the CPA attack, then we do a hardness analysis that determines the least number of coefficients needed to be recovered in the CPA.

### 4.1 Lattice Construction

Now we have some of the coefficients being recovered, the next step is to recover the unknown coefficients by the lattice attack. Because of the structure of incomplete NTT in Kyber, we know that coefficients are split into $2 k$ groups of 128 ones. We will focus on one group and notice that the rest of the steps need to repeat $2 k$ times to derive the full secret key.

Let $\mathbf{M}=\left[\mathbf{m}_{0}, \mathbf{m}_{2}, \ldots, \mathbf{m}_{254}\right]$ be the inverse NTT matrix as we mentioned in Section 2.4. Suppose we have recovered $128-\ell$ coefficients in $\hat{\mathbf{s}}_{i}$, one of the groups in $\hat{\mathbf{s}}$, from the polynomial multiplication $\hat{\mathbf{s}} \circ \hat{\mathbf{u}}$, i.e., we need to recover the remaining $\ell$ coefficients. Let $A=\left\{a_{0}, a_{1}, \ldots, a_{127-\ell}\right\}$ be the indices that are successfully recovered in the CPA step, and $B=\left\{b_{0}, b_{1}, \ldots, b_{\ell-1}\right\}$ be the indices that are still unknown, then the inverse NTT $\operatorname{NTT}^{-1}\left(\hat{\mathbf{s}}_{i}\right)=\mathbf{M} \hat{\mathbf{s}}_{i}=\mathbf{s}_{i} \bmod q$ can be split into two halves as followed:

$$
\mathbf{M}_{A} \hat{\mathbf{s}}_{i, A}+\mathbf{M}_{B} \hat{\mathbf{s}}_{i, B}=\mathbf{s}_{i} \quad \bmod q,
$$

where $\mathbf{M}_{A}:=\left[\mathbf{m}_{a_{0}}, \ldots, \mathbf{m}_{a_{127}-\ell}\right]$ is a matrix whose columns are those of $\mathbf{M}$ whose indices are in $A, \hat{\mathbf{s}}_{i, A}=\left[\hat{s}_{a_{0}}, \ldots, \hat{s}_{a_{127-\ell}}\right]^{\top}$, and the similar definition for $\mathbf{M}_{B}$ and $\hat{\mathbf{s}}_{i, B}$.

Notice that $\mathbf{s}_{i}$ is an extremely short vector since it is the secret key sampled from $\beta_{\eta}$. By calling the known vector $\mathbf{t}=\mathbf{M}_{A} \hat{\mathbf{s}}_{i, A}$, the known basis $\mathbf{A}=-\mathbf{M}_{B}$, and an unknown vector $\mathbf{s}_{i}^{\prime}=\hat{\mathbf{s}}_{i, B}$, we now have $\mathbf{t}=\mathbf{A} \mathbf{s}_{i}^{\prime}+\mathbf{s}_{i} \bmod q$, which is exactly the definition of an LWE problem. Compared to the original moduleLWE problem in Kyber, this problem becomes simpler since the rank of $\mathbf{A}$ is less than the original one.

### 4.2 Hardness Analysis

We use the standard technique of Kannan's embedding to solve the LWE problem. First we treat the LWE problem as a BDD/uSVP problem and then apply a lattice reduction algorithm. For example, given the instance above $\left(\mathbf{A}, \mathbf{t}=\mathbf{A s}_{i}^{\prime}+\mathbf{s} \bmod q\right)$, consider the lattice $\Lambda\left(\mathbf{B}_{B D D}\right)$ generated by

$$
\mathbf{B}_{B D D}=\left[\begin{array}{cc}
\mathbf{I}_{\ell} & \mathbf{A}^{\prime} \\
\mathbf{0} & q \mathbf{I}_{n-\ell}
\end{array}\right],
$$

where $\left[\mathbf{I}_{\ell} \mid \mathbf{A}^{\prime}\right]$ denotes the reduced row echelon matrix of $\mathbf{A}^{\top}$, which can be easily calculated by Gaussian elimination. We can then solve the BDD of $\Lambda\left(\mathbf{B}_{B D D}\right)$ with respect to the target point $\mathbf{t}$ which reveals $\mathbf{s}^{\prime}$ and $\mathbf{s}$.

Alternatively, we can reduce this BDD to USVP by a technique called Kannan's embedding [14]. Given the BDD instance above, we consider the following basis matrix

$$
\mathbf{B}_{K a n}=\left[\begin{array}{cc|c}
\mathbf{I}_{l} & \mathbf{A}^{\prime} & \mathbf{0} \\
\mathbf{0} q \mathbf{I}_{n-\ell} & \\
\hline \mathbf{t}^{\top} & 1
\end{array}\right] .
$$

Recall that the lattice $\Lambda\left(\mathbf{B}_{\text {Kan }}\right)$ contains all linear combinations of the vectors in $\mathbf{B}_{\text {Kan }}$. The equation $\mathbf{t}=\mathbf{A s} \mathbf{s}_{i}^{\prime}+\mathbf{s}_{i} \bmod q$ can be written as $\mathbf{t}=\mathbf{A} \mathbf{s}_{i}^{\prime}+\mathbf{s}_{i}+q \mathbf{k}$, where $\mathbf{k} \in \mathbb{Z}_{q}^{n}$, so there exists a row vector $\left[-\mathbf{s}^{\prime \prime \top}\left|-\mathbf{k}^{\prime \top}\right| 1\right] \in \mathbb{Z}_{q}^{n+1}$ such that the shortest vector in $\Lambda\left(\mathbf{B}_{\text {Kan }}\right)$ is $\left[-\mathbf{s}^{\prime \prime \top}\left|-\mathbf{k}^{\prime \top}\right| 1\right] \cdot \mathbf{B}_{\text {Kan }}=\left[\mathbf{s}_{i}^{\top} \mid 1\right] \in \mathbb{Z}_{q}^{n+1}$.

The norm of vector $\left[\mathbf{s}_{i}^{\top} \mid 1\right]$ is $\sqrt{\left\|\mathbf{s}_{i}\right\|^{2}+1} \approx \sqrt{n} \sigma_{s}$. If this norm is smaller than the norm of the shortest vector estimated by the Gaussian Heuristic, this uSVP instance can be solved, and the more gap between the first and second successive minima, i.e., the bigger $\lambda_{2}\left(\Lambda\left(\mathbf{B}_{\text {Kan }}\right)\right) / \lambda_{1}\left(\Lambda\left(\mathbf{B}_{\text {Kan }}\right)\right)$ is, the easier the USVP will be. Since the volume of the lattice $\Lambda\left(\mathbf{B}_{\text {Kan }}\right)$ is $q^{n-\ell,} \lambda_{2}\left(\Lambda\left(\mathbf{B}_{\text {Kan }}\right)\right)$ can be estimated by

$$
\lambda_{2}\left(\Lambda\left(\mathbf{B}_{K a n}\right)\right) \approx \sqrt{\frac{n+1}{2 \pi e}} q^{(n-\ell) /(n+1)} .
$$

To determine the least number of coefficients we must recover in the CPA step, we do an experiment on solving the SVP randomly generated by script. The result is shown in Fig. 2, where the blue line is the success rate of finding $\left[\mathbf{s}_{i}^{\top} \mid 1\right]$ by the BKZ algorithm ${ }^{1}$ of block size 50 for 20 randomly generated $\mathbf{s}$, and the red line is the running time of the algorithm. From the result, the critical point of guaranteed success is on $\ell=89, \ell=90$ for Kyber512, Kyber768/1024, respectively. This means that in the CPA step, we need at least $128-89=39$ (or 38 for Kyber768/1024) recovered coefficients so that we can have a fully recovered secret key when using the BKZ algorithm of block size 50 to solve the reduced SVP problem. Notice that in order to do a full key recovery, the number of recovered coefficients need to be multiplied by $2 k$, where $k$ is the module dimension for each version of Kyber. The reason that Kyber768/1024 is easier to solve is because $\eta$ of Kyber768/1024 is smaller than that of Kyber512.

[^5]

Fig. 2: Success rate and running time on randomly generated uSVP in the lattice $\mathbf{B}_{\text {Kan }}$ for (a) Kyber512 and (b) Kyber768/1024

## 5 Experiments

We experimented our attacks on simulated power traces of the ARM cortex-M0 processor, then estimate how many traces we need to conduct our attack.

### 5.1 ELMO

Our simulated traces were generated using the ELMO [10], which emulates the power consumption of an ARM Cortex M0 processor and produces noise-free traces. The tool reproduces the 3 -stage pipeline of an M0 processor, which means that the algorithmic noise is taken into account. ELMOs quality has been established by comparing leakage detection results between simulated and real traces from a STM32F0 Discovery Board [20]. For reference, to conduct a successful key recovery power analysis on the lattice-based signature scheme FALCON, the required numbers of simulated power traces and real acquisitions are 2000 and 5000 [12].

### 5.2 Results

Table 2 gives the results of our experiment done on the simulated traces. The threshold is the minimum correlation coefficient of acceptance that we set as a parameter in Section 3.2. Recovered rate is the average number of successfully recovered coefficients, and false positive is the average number of coefficients that are accepted but turn out to be wrong. The success rate is the possibility of all $39 / 38$ coefficients we randomly sample being the correct ones when we choose from all coefficients that are accepted by the CPA step, which can be directly calculated by $\binom{a-39}{b} /\binom{a}{b}$ if $a$ is the recovered rate and $b$ is the false positive. Therefore, it does not mean the overall success rate of our attack, the overall success rate will be arbitrarily closed to 1 if we keep sampling the coefficients as long as we have at least $39 / 38$ correct ones.

Y.T. Kuo, A. Takayasu

Table 2: Experimental results on different acceptance threshold and trance number. Left hand side of success rate is for Kyber512 and right is for Kyber768/1024. Threshold Trace number Recovered rate False positive Success rate

| 0.63 | 200 | $110.5 / 128$ | $6 / 128$ | $0.07(0.07)$ |
| ---: | ---: | ---: | ---: | ---: |
|  | 400 | $118.75 / 128$ | $4.25 / 128$ | $0.18(0.19)$ |
|  | 600 | $124.75 / 128$ | $3 / 128$ | $0.32(0.33)$ |
|  | 800 | $124.75 / 128$ | $1.75 / 128$ | $0.52(0.54)$ |
| 0.65 | 200 | $98.75 / 128$ | $4.5 / 128$ | $0.10(0.11)$ |
|  | 400 | $109 / 128$ | $4.25 / 128$ | $0.15(0.16)$ |
|  | 600 | $112 / 128$ | $2.25 / 128$ | $0.39(0.40)$ |
|  | 800 | $116.25 / 128$ | $1.5 / 128$ | $0.55(0.56)$ |
| 0.67 | 200 | $79.25 / 128$ | $2.5 / 128$ | $0.19(0.20)$ |
|  | 400 | $86 / 128$ | $0.5 / 128$ | $0.77(0.78)$ |
|  | 600 | $83.75 / 128$ | $0.25 / 128$ | $0.88(0.89)$ |
|  | 800 | $86.5 / 128$ | $0 / 128$ | $1(1)$ |
|  | 200 | $58 / 128$ | $1 / 128$ | $0.33(0.34)$ |
| 0.69 | 400 | $53.25 / 128$ | $0.25 / 128$ | $0.82(0.82)$ |
|  | 600 | $49.75 / 128$ | $0 / 128$ | $1(1)$ |
|  | 800 | $49.5 / 128$ | $0 / 128$ | $1(1)$ |

It can be seen that although adding trace numbers does not help much to increase the recovered coefficients, it does help to lower the false positive, which directly affects the success rate. Increasing the threshold of acceptance will also lower the false positive and recovered rate, but notice that if the recovered rate drops below 39, our attack may fail. Since the running time of the overall attack is dominated by CPA, we would argue that the fewer the number of power traces the better it is, as long as the success rate is higher than 0.05 .

### 5.3 Application to Saber

Saber [9] is a lattice-based key encapsulation mechanism based on the Module Learning With Rounding problem. Saber is one of the round 3 candidates of the NIST post-quantum cryptography standardization competition. The polynomial ring used within Saber is $R_{q}=\mathbb{Z}_{q}[x] /\left(x^{n}+1\right)$ with $q=2^{13}$ and $n=256$ across all parameter sets. Saber also offers three security levels: Lightsaber with security level similar to AES-128, Saber with one similar to AES-192 and Firesaber with one similar to AES-256.

Because Saber was not specifically designed to benefit from NTT-based multiplication by using an NTT-friendly ring, it uses a combination of Toom-4 and Karatsuba to implement efficient polynomial arithmetic. However, as shown in the work by [6], NTTs can be used to obtain efficient polynomial arithmetic in finite fields modulo a power-of-two. They did this by choosing a a prime $p>n q^{2} / 2$ such that $n \mid(p-1)$, computing the multiplication by the NTT over

(c)

Fig. 3: Success rate and running time on randomly generated uSVP for (a) Lightsaber, (b) Saber and (c) Firesaber
$\mathbb{Z}_{p}[x]$, and then reducing the result back to $\mathbb{Z}_{q}[x]$. Since the modulus is much bigger in the NTT for Saber, the SCA for pointwise multiplication on Saber needs to target a smaller portion of the intermdeiate value, which results in smaller signal-to-noise ratio. In [21], a minimum of 10000 traces was required to mount a successful attack.

Figure 3 shows our lattice attack when applying to the SCA proposed by [21]. Since the implementation uses 6 layers of NTTs, we divide the coefficients into $512 / 2^{6}=8$ groups and find the minimum number of coefficients we needed to recover other one. We can see that it needs $9 / 8 / 7$ coefficients out of 64 to guarantee a successful attack for each parameter sets of Saber, which means a total of $72 / 64 / 56$ coefficients are needed. This saves about $86 \% \sim 89 \%$ of the running time for the SCA. Another way to see the improvement is the possibility to reduce the traces of SCA. Although by doing so, there may be incorrectly recovered coefficients, by our sampling approach as shown before, we only need portion of the coefficients correct to recover the whole secret key. We do want to point out that the improvement heavily depends on the implementation of the incomplete NTT of choice. That is, the less layers of incomplete NTTs an implementation chooses, the less coefficients we need to perform the lattice attack.

## 6 Conclusion

In this paper, we propose a combined CPA and lattice attack on Kyber. With 200 traces, our attack terminated within 20 minutes on a 16 -core computer. Compared to other SCA targeting NTT in the cryptosystems, our attack achieves lower runtime in practice. Furthermore, there is potential for decreasing the number of traces by using lattice reduction if the same measurement is used.

Our future works are to migrate the attacks to real devices and other cryptosystems using the NTT transform multiplication like Saber or NTRU. We can also investigate the effect of popular countermeasures of CPA like masking and hiding on our attack.
Acknowledgement. This research was conducted under a contract of Research and Development for Expansion of Radio Wave Resources (JPJ000254) the Ministry of Internal Affairs and Communications, Japan. This work was partially supported by JSPS KAKENHI Grant Number 19K20267, Japan.

## References

1. Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé., D.: CRYSTALS-Kyber (version 3.02) submission to round 3 of the NIST post-quantum project. Specification document (2021)
2. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehle, D.: CRYSTALS - Kyber: A CCA-secure module-lattice-based KEM. In: 2018 IEEE European Symposium on Security and Privacy (EuroS\&P). pp. 353-367 (2018). https://doi.org/10.1109/EuroSP.2018.00032
3. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J., Schwabe, P., Seiler, G., Stehlé, D.: Kyber. https://github.com/pq-crystals/kyber (2023)
4. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. pp. 309-325. ITCS '12, Association for Computing Machinery (2012). https://doi.org/10.1145/2090236.2090262, https: //doi.org/10.1145/2090236.2090262
5. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology - ASIACRYPT 2011. pp. 1-20 (2011)
6. Chung, C.M.M., Hwang, V., Kannwischer, M.J., Seiler, G., Shih, C.J., Yang, B.Y.: NTT multiplication for NTT-unfriendly rings. Cryptology ePrint Archive, Paper 2020/1397 (2020), https://eprint.iacr.org/2020/1397, https://eprint.iacr.org/ 2020/1397
7. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex Fourier series. Mathematics of Computation 19, 297-301 (1965)
8. Dachman-Soled, D., Ducas, L., Gong, H., Rossi, M.: LWE with side information: Attacks and concrete security estimation. In: Advances in Cryptology CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Proceedings, Part II. pp. 329-358 (2020). https://doi.org/10.1007/978-3-030-56880-1_12, https://doi.org/10.1007/978-3-030-56880-1_12
9. D'Anvers, J.P., Karmakar, A., Sinha Roy, S., Vercauteren, F.: Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) Progress in Cryptology - AFRICACRYPT 2018. pp. 282-305 (2018)
10. ELMO: Evaluating leaks for the arm cortex-m0. https://github.com/sca-research/ ELMO, accessed: 2022-10-17
11. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Proceedings. pp. 537-554 (1999). https://doi. org/10.1007/3-540-48405-1_34
12. Guerreau, M., Martinelli, A., Ricosset, T., Rossi, M.: The hidden parallelepiped is back again: Power analysis attacks on Falcon. Cryptology ePrint Archive, Paper 2022/057 (2022), https://eprint.iacr.org/2022/057, https://eprint.iacr.org/2022/ 057
13. Hamburg, M., Hermelink, J., Primas, R., Samardjiska, S., Schamberger, T., Streit, S., Strieder, E., van Vredendaal, C.: Chosen ciphertext k-trace attacks on masked CCA2 secure Kyber. IACR Transactions on Cryptographic Hardware and Embedded Systems 2021, Issue 4, 88-113 (2021). https://doi.org/10.46586/tches.v2021. i4.88-113, https://tches.iacr.org/index.php/TCHES/article/view/9061
14. Kannan, R.: Minkowski's convex body theorem and integer programming. Mathematics of Operations Research 12(3), 415-440 (1987), http://www.jstor.org/ stable/3689974
15. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) Advances in Cryptology - CRYPTO' 99. pp. 388-397 (1999)
16. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Proceedings. pp. 104-113 (1996). https: //doi.org/10.1007/3-540-68697-5__9
17. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. J. ACM 60(6) (2013). https://doi.org/10.1145/2535925, https://doi. org/10.1145/2535925
18. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer Publishing Company, Incorporated, 1st edn. (2010)
19. May, A., Nowakowski, J.: Too many hints - when LLL breaks LWE. Cryptology ePrint Archive, Paper 2023/777 (2023), https://eprint.iacr.org/2023/777, https: //eprint.iacr.org/2023/777
20. McCann, D., Oswald, E., Whitnall, C.: Towards practical tools for side channel aware software engineering: Grey box' modelling for instruction leakages. In: Proceedings of the 26th USENIX Conference on Security Symposium. pp. 199-216 (2017)
21. Mujdei, C., Wouters, L., Karmakar, A., Beckers, A., Mera, J.M.B., Verbauwhede, I.: Side-channel analysis of lattice-based post-quantum cryptography: Exploiting polynomial multiplication. ACM Trans. Embed. Comput. Syst. (2022). https:// doi.org/10.1145/3569420, https://doi.org/10.1145/3569420
22. National Institute of Standards and Technology. Post-quantum cryptography standardization. https://csrc.nist.gov/projects/post-quantum-cryptography, accessed: 2022-10-12
23. Pessl, P., Primas, R.: More practical single-trace attacks on the number theoretic transform. Cryptology ePrint Archive, Paper 2019/795 (2019), https://eprint.iacr. org/2019/795, https://eprint.iacr.org/2019/795
24. Primas, R., Pessl, P., Mangard, S.: Single-trace side-channel attacks on masked lattice-based encryption. Cryptology ePrint Archive, Paper 2017/594 (2017), https: //eprint.iacr.org/2017/594, https://eprint.iacr.org/2017/594
25. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing. vol. 56, pp. 84-93 (2005). https://doi.org/10.1145/1568318.1568324
26. Shor, P.: Algorithms for quantum computation: Discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science. pp. 124-134 (1994). https://doi.org/10.1109/SFCS.1994.365700

# A Comparative Analysis of Rust-Based SGX Frameworks: Implications for building SGX applications ${ }^{\star}$ 

Heekyung Shin ${ }^{1}$, Jiwon Ock ${ }^{1}$, Hyeon No ${ }^{1}$, and Seongmin Kim ${ }^{1 \star \star}$<br>Sungshin Women's University, Seoul, Korea ${ }^{1}$<br>\{220224009, 220224011, 220224007, sm.kim\}@sungshin.ac.kr


#### Abstract

The widespread adoption of Intel Software Guard Extensions (SGX) technology has garnered significant attention, primarily owing to its robust hardware-based data-in-use protection. To alleviate the complexities of SGX application development, an approach involving the incorporation of a Library Operating System (LibOS) within an enclave has gained prominence. This strategy enables SGX utilization without necessitating extensive modifications to legacy code. However, this approach increases the potential attack surface and may be susceptible to memory corruption vulnerabilities. To address this challenge, the trend of leveraging Rust programming language offering memory safety guarantees for implementing system components has prompted the development of Rust-based SGX frameworks. But still, a gap exists in providing guidelines or systematic analyses to aid developers in selecting a suitable Rust-based SGX framework, considering factors like implementation cost and runtime overhead. This study undertakes a comprehensive comparative analysis of three representative SGX frameworks implemented with Rust: Rust SGX SDK, Occlum, and Fortanix EDP. Our analysis encompasses an exploration of their internal implementations, focusing on their impact on both performance and security. Additionally, we quantify the engineering effort required for migrating legacy Rust applications and evaluate the supplementary overhead incurred when subjecting these frameworks to CPU and memory-intensive workloads. By conducting this analysis, we aim to provide valuable guidance to developers seeking to choose a Rust-based SGX framework that aligns with their application's specific purpose and workload characteristics.


Keywords: Trusted Execution Environment • Intel SGX • Rust

## 1 Introduction

The commercialization of Intel Software Guard Extensions (SGX) technology [12] has garnered substantial industrial and academic attention. In particular, In-

[^6]tel SGX technology plays a pivotal role in evolving the confidential computing paradigm [28]. This interest is primarily driven by its robust hardware-based data-in-use protection and its inherent practicality, notably its compatibility with the x86 architecture ensuring native speed [6]. By leveraging SGX to legacy applications, it is possible to guarantee the confidentiality and integrity of cloudbased TEE service. In fact, leading cloud service providers (CSPs) have begun offering public cloud instances supporting SGX functionalities. These groundbreaking solutions, known as confidential VMs, include commercial products like Amazon Nitro Enclaves [3] and Azure Confidential Computing [26]. Such innovation has expedited the widespread adoption of confidential computing across diverse domains, such as safeguarding AI/ML models [13, 21], protecting digital assets [22], and securing key management services $[10,35]$.

Basically, there are two primary approaches for implementing the SGX program: 1) porting an application based on SGX SDK [1] and 2) running unmodified applications on top of frameworks that support SGX compatibility [6]. In particular, the adoption of a Library Operating System (LibOS) within the enclave has emerged as a viable strategy to facilitate the utilization of SGX without necessitating modifications to legacy code [5,6,27,33]. The LibOS-based strategy offers distinct advantages when porting legacy applications into the SGX environment. Developers are relieved from the complexities of segregating securitysensitive components from the original code-base and re-implementing system call wrappers for enclave transitions. However, it is important to note that this design choice expands the potential attack surface, given that the entire LibOS codebase is loaded and executed within an SGX enclave. SGX does not guarantee the memory safety of the enclave, which means that memory corruption vulnerabilities inherent in traditional code written in languages like C or $\mathrm{C}++$ (e.g., Heartbleed [7]) can still be effective even when executed within the security boundary provided by SGX CPU [20, 29]. Therefore, an additional instrumentation or protection mechanism is required to achieve robustness over memory vulnerabilities.

Simultaneously, the rise of the Rust programming language has equipped developers with a potent instrument for constructing robust and secure applications. Rust delegates memory safety checking (e.g., rust pointer always references valid memory) to the Rust compiler. In contrast to low-level codes implemented in C or $\mathrm{C}++$ that are prone to subtle memory bugs, Rust guarantees memory safety by rejecting the compilation of them by introducing features, such as ownership and lifetime elision rules [24]. Furthermore, Rust is fast and memoryefficient as its runtime does not require a garbage collector to reclaim memory space, making it well-suited for the development of performance-critical services. This appeal leads to the adoption of Rust in state-of-the-art system software, including container runtimes [2], microkernels [19], and storage systems [17].

Such a trend has also spurred the development of the SGX framework tailored for Rust utilization. The state-of-the-art LibOS-based SGX frameworks have extended support for the execution of Rust applications [27,33]. Besides, several studies $[8,31,34]$ utilize Rust programming language [24] as the foun-
dation for building SGX frameworks. Such design choice enables developers to reduce runtime overhead (e.g., garbage collection), thereby drawing attention to the potential of leveraging Rust in SGX framework development. Nevertheless, a notable gap persists in the absence of comprehensive guidelines or systemic analyses that can aid developers in selecting the most suitable Rust-based SGX framework for their applications. Such guidelines would encompass considerations related to implementation cost and runtime overhead, crucial factors when deciding to execute existing applications or develop new Rust applications in the SGX environment.

This study conducts a comparative study on existing Rust-based SGX frameworks to provide implications for newly implementing or porting legacy securitysensitive Rust applications. For this, we conduct an in-depth analysis between three cutting-edge Rust-based SGX frameworks: Rust SGX SDK, Occlum, and Fortanix EDP. First, we explore the internal implementation details of each framework relevant to the application performance and security. Then, we quantify the engineering effort required to deploy legacy Rust applications atop these frameworks, providing insights into the ease of transition. Finally, we evaluate the additional overhead incurred by each framework, subjecting them to CPU-intensive and memory-intensive workloads to gauge their performance implications. We believe our analysis provides guidance for developers to select an appropriate Rust-based SGX framework when implementing an SGX application according to its purpose and workload characteristics.

## 2 Background

### 2.1 Intel SGX and LibOS-based SGX Framework

Intel SGX is a secure processor architecture to ensure trustworthiness of application to protect sensitive and valuable information. It offers an isolated protection domain in memory called an enclave, which is only decrypted within the CPU package when executing it as an enclave mode. This ensures that even system administrators or other software running on the host cannot access the sensitive data in the enclave. To help developers implement SGX applications, Intel provides the SGX Software Development Kit (SDK). The SDK offers essential libraries and toolchains for tasks such as enclave signing and debugging [25]. It simplifies the process of creating secure enclaves and managing their execution. For building an SGX application using SDK, a developer needs to separate an application codebase into two parts, an enclave region and an untrusted region. In addition, the transition interface between them must be defined by a developer in the Enclave Definition Language (EDL). This interface specifies the secure functions ECALLs for entering an enclave mode and functions OCALLs that can be invoked to switch execution to the untrusted region. Additionally, EDLs detail how data should be transferred in and out of the enclave, specifying data structures and communication mechanisms. Note that OCALLs are typically used for handling system calls, as SGX does not allow executing syscall instructions in an enclave mode.

LibOS-based SGX focuses on using a Library OS that provides operating system functionality in the form of a library to act as an interface between applications and hardware. It runs entirely within an enclave, and to port an application into an enclave, the application binary needs to be loaded and executed along with the libraries it relies on. One of the key advantages of LibOS-based SGX is the simplification of the enclave interface. This minimizes the number of system calls that occur within the enclave, ensuring that the code running within the enclave does not require system calls that involve crossing between user and kernel domains. LibOS also plays a crucial role in implementing and managing necessary operating system functionalities within the enclave when executed in user space. This allows enclaves to handle privileged operations that would typically require execution in processor supervisor mode, maintaining security isolation while performing necessary tasks. Operations represented as system calls, particularly those related to file system operations, can be straightforwardly implemented within LibOS by modifying data structures related to the file system implementation. These system calls do not impact the security of other application programs and do not require execution by privileged system software [30]. Frameworks such as Grammine [33], SGX-LKL [27], and Haven [6], which implement LibOS-based SGX, offer the advantage of enhancing portability by freeing applications from dependence on a specific operating system.

### 2.2 Rust Programming Language

Rust is a newly introduced programming language developed by Mozilla Research that guarantees safety on the memory side with cost-free abstraction [24]. Rust delegates memory safety checking (e.g., rust pointer always references valid memory) to the Rust compiler. In contrast to low-level codes implemented in C or C++ prone to subtle memory bugs, Rust guarantees memory safety by rejecting their compilation by introducing features, such as ownership and lifetime elision rules [24]. Such design choice enables developers to minimize a runtime overhead (e.g., garbage collection), which in turn introduces the attention to utilizing Rust for implementing system software [24]. Rust introduces a unique ownership system central to its memory safety guarantees [16]. The ownership system enforces strict rules about how memory is allocated and deallocated, ensuring that memory is managed safely without the risk of common bugs like null pointer dereferences, data races, and memory leaks. Rust also incorporates lifetime, which are annotations that specify the scope or duration for which references are valid [16]. It prevents references from outliving the data they point to or being used after the data has been deallocated.

## 3 Characteristics Analysis of Frameworks

To take advantage of Rust mentioned above (e.g., guaranteeing in-enclave memory safety), recent studies utilize Rust when implementing an SGX framework itself and enable developers to execute Rust applications on SGX environment [8,


Fig. 1: Rust-based SGX Framework Overview. (The red boxes indicate regions that are isolated and protected by the enclave application, while the black dashed boxes are regions that are written in Rust.)
$31,34]$. In particular, we provide an overview of three existing frameworks that facilitate the development of SGX applications in Rust: Rust SGX SDK, Occlum, and Fortanix EDP. As depicted in Figure 1, these frameworks each exhibit a distinct system architecture. It is worth noting that Occlum exclusively employs a LibOS-based approach, while both Rust SGX SDK and Fortanix EDP offer a custom interface to interact with the host OS for system operations.

### 3.1 Fortanix EDP

Enclave development platform (EDP) [8], developed by Fortanix, offers a distinct advantage in generating and running enclave from scratch with Rust code, eliminating the dependency on the Intel SGX SDK [8]. Notably, Fortanix EDP introduces its own unique API and ABI while ensuring binary-level compatibility for Rust applications. Specifically, EDP's usercall interface is designed not to expose existing enclave interface attack surfaces. It achieves this by incorporating elements that handle memory allocation in user space and data copying from user memory within the context of a Rust-type system. This approach effectively safeguards against direct memory access, preemptively mitigating time-of-check time-of-use (TOCTOU) attacks. It's worth noting that the usercall interface establishes a connection to the syscall interface through an enclave. Within the untrusted region, an enclave runner takes on the responsibility of managing enclave loading and serves as an intermediary layer bridging the gap between usercall requests originating from the enclave and the syscall interface required for external interactions. While EDP enables the utilization of much of Rust's standard library for application implementation, it intentionally imposes restrictions on specific functionalities, such as multi-processing support and file system operations, for security reasons.

### 3.2 Occlum

Occlum is a memory-safe multi-process LibOS for Intel SGX to enable execution of legacy applications without modifying the source code [31]. Occlum proposes multi-domain software fault isolation (MMDSFI) by leveraging Intel Memory Protection Extensions (MPX) technology [14] to preserve isolation between processes that share a single address space. To support this, the Occlum framework has newly implemented SGX LibOS, the Occlum toolchain, and the Occlum verifier. Untrusted $\mathrm{C} / \mathrm{C}++$ code can generate executable binaries through the Occlum toolchain and be verified by the Occlum verifier, ensuring the integrity of MMDSFI. Consequently, the verified MMDSFI enables the secure construction of the LibOS within the enclave.

LibOS based on Intel SGX SDK and Rust SGX SDK is predominantly implemented in Rust, accounting for approximately $90 \%$ of the codebase, with the remainder implemented in C. This supports the execution of enclaves in both C and Rust, providing protection for enclave programs against potential memory vulnerabilities. Furthermore, to protect LibOS from unsafe entities, a shim layer called occlum-PAL is provided to the application, offering APIs. This isolation mechanism is crucial for security as it prevents one process from interfering with or accessing the memory of another with strict boundary checking. By securely sharing the enclave's single address space with Occlum's SFI-isolated processes (SIPs) which is a unit of application domain, it supports multi-tasking efficiently. For example, compared to other SGX frameworks that utilize LibOS with supporting multi-tasking [5, 6, 33], startup time is 1000 times faster and IPC (inter-process communication) is up to 3 times faster [31].

### 3.3 Rust SGX SDK (Teaclave SGX SDK)

The Rust SGX SDK, developed by Baidu, offers a secure platform for executing Rust-based applications within SGX environments [34]. This SDK introduces a wrapper Rust API that layers Rust functionalities on top of the SGX SDKs, originally implemented in C and C++. Through this layered approach, it establishes a secure connection between the Intel SGX SDK code and the trusted application. Notably, as a dependency on the Intel SGX SDK, it places trust exclusively in the software operating within an enclave while maintaining untrusted towards the rest of the system. The SDK doesn't provide its own Application Binary Interface (ABI) but instead adheres to the same ABI as the vanilla Intel SGX SDK. This strategic choice ensures seamless compatibility between the Rust SGX SDK and the Intel SGX environment. Consequently, any updates or alterations within the SGX ecosystem can be swiftly accommodated without the risk of breaking compatibility.

## 4 Qualitative aspects affecting application performance

In this section, we conduct in-depth analysis by systemically exploring the internal design of each framework and categorize three key indicators related to
application performance: Memory boundary check, Enclave transition, and additional runtime overhead. Table 1 summarizes our analysis result.

|  | Memory boundary <br> check | Enclave <br> Transition | Runtime <br> Overhead | Memory <br> Safety |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Occlum | MMDSFI | PAL API | Enclave SIP | Enclave SIP |
| Incubator Teaclave <br> SGX SDK | Runtime (Enclave-runner) | Legacy ECALL/OCALL | Rust Wrapper API | Rust Wrapper API |
| Fortanix EDP | Sanitizable function | Usercall (Custom) | Own ABI | Own API and ABI |

Table 1: Estimating framework performance impact overhead based on framework analysis

### 4.1 Memory boundary check

To avoid overhead caused by unnecessary bound checking, Rust SGX SDK provides a Sanitizable function to check the raw byte array and verify that memory represents a valid object when binding an application. For the case of Fortanix EDP, the enclave-runner runtime checks before entering an enclave to ensure processor state sanitation, similar to Rust SGX SDK. Finally, Occlum utilizes SFI (Software Fault Isolation), a software instrumentation technique that sandboxes untrusted domains within a single address space to reduce the enclave size in a multi-tasking environment. However, Occlum performs boundary checking for every memory access to ensure that it does not deviate from the domain boundary, which becomes a runtime overhead.

### 4.2 Enclave transition (ECALL/OCALL)

Rust SGX SDK follows the design choice made by Intel SGX SDK for implementing enclave transition wrapper, ECALL (enclave call) and ocall (out-call) ${ }^{1}$. To make legacy Ecalls and ocalls implemented in C compatible with Rust application code, Rust SGX SDK provides wrapper routines by leveraging Rust's unsafe keyword, which explicitly translates the boundary between C code and Rust code for foreign function interface (FFI). During the conversion, sanity checking is performed, resulting in runtime overhead. Fortanix EDP, on the other hand, defines the usercall interface written in Rust, instead of writing ECALL and OCALL for enclave transition. Because they use their own call process, which is not optimized for SGX, each interaction related to the enclave would generate transition overhead using the usercall interface [34]. Similarly, Occlum inserts a trampoline code with a byte that identifies the domain ID in MMDSFI

[^7]to securely implement untrusted binaries generated by the toolchain in LibOS. In other words, entry into the LibOS within the Enclave can only occur using this trampoline code. Furthermore, to exit outside the LibOS, one must verify the predefined domain ID once again before being allowed to escape. Therefore, from the user's perspective in Occlum, there is no need to write an EDL file. Instead, users can utilize the pre-defined occlum build command to build the enclave image and the occlum run command to use the enclave entry point. Within the Occlum framework, the run command is passed to the PAL API Layer to enter the enclave. The process of passing through the PAL Layer to enter the enclave can involve transition overhead [18].

### 4.3 Runtime overhead (Miscellaneous)

The Rust SGX SDK raises an additional overhead due to the dependency on Intel SGX SDK by calling a different directory SGX instruction with the Rust layer, rather than directly executing the assembly code. On the other hand, Fortanix EDP uses its own ABI, called fortanix-sgx-abi [9], implemented with a pure rust abstraction layer, so it is relatively overhead-free [15]. When assuming multi-tasking scenario, Occlum has an advantage compared to other frameworks, as it handles multiple process domains(SIPs) within a single enclave region. Such a design also saves the cost of inter-process communication (IPC) overhead between processes.

### 4.4 Memory safety guranteed by each framework

Both the Rust SGX SDK and Occlum have dependencies on the C language Intel SGX SDK layers, with the Rust SGX SDK utilizing a wrapper API implemented in Rust, and Occlum having $90 \%$ of its LibOS code written in Rust. When these frameworks have dependencies on the Intel SGX SDK, they remain susceptible to various vulnerabilities, including DoS attacks and side-channel attacks. In other words, Occlum and Rust SGX SDK may share similar security threats at the library level. However, Occlum can leverage enclave SIP to defend the enclave against attacks such as code injection and ROP attacks by providing isolation between processes that protect SIP from other SIPs and between processes that protect LibOS itself from any SIP and LibOS.

In contrast, Fortanix EDP distinguishes itself by defining its own API and ABI based on the Rust language, thereby enhancing security against vulnerabilities like side-channel attacks that are inherent in the Intel SGX SDK. Additionally, Fortanix EDP is designed in a way that similar to how a LibOS operates, does not expose the enclave interface surface to the user. Additionally, by limiting the number of usercall interfaces to fewer than 20, it reduces the attack surface. Furthermore, it allocates memory in user space and utilizes elements like fortanix_sgx::usercalls::alloc to prevent direct memory access, thereby proactively mitigating Time-of-Check-to-Time-of-use (TOCTOU) attack.

Rust SGX SDK introduces an extra layer of wrappers, which can lead to performance degradation. This may manifest as slower enclave execution and a
higher demand for system resources. While Occlum provides isolation between SIPs, there can be overhead in terms of communication and data sharing between processes due to this isolation. Fortanix EDP makes changes to memory allocation and access methods to defend against TOCTOU attacks. However, these changes can result in additional overhead for memory management and internal enclave operations. Additionally, limiting the number of user call interfaces for security purposes can restrict the functionality and flexibility of enclaves. All three frameworks may require extra security and compliance checks during enclave execution and communication, which can slow down the overall execution speed.

## 5 Performance evaluation

In this section, we describe our experimental setup and present the results of our experimental evaluations of application workloads on each framework. Based on the analysis Section 4, specified the following evaluation metrics: 1)Execution time measurement to evaluate the performance of the application according to the characteristics, 2) Enclave size measurement result to evaluate the enclave hardening and security. The results of the two performance evaluations are summarized in Table 2 and Table 3.
Experimental Setup. Our evaluation was assessed on Ubuntu 20.04. The SGX SDK for developing SGX applications utilized 2.18v. For the Rust language, we used rustc 1.66.0-nightly, which is compatible with all frameworks. Additionally, Occlum used glibc 2.31, as there are glibc versions compatible with running musl-based applications.
Application Benchmark. Ring is a library that exposes a Rust API, primarily utilized for performing CPU-intensive workloads related to encryption. It emphasizes the implementation, testing, and optimization of a core set of cryptographic operations exposed through an API that is both easy to use and resistant to misuse. Considering the computationally intensive nature of encryption and decryption processes, we intend to leverage this code to evaluate the CPU computational load of each framework.

HashMap in Rust is utilized for mapping and storing keys and values, offering swift search and insertion operations. However, this process entails the need for basic object implementations, an array of hash tables, and individual objects for each hash item, resulting in a memory-intensive workload with substantial RAM consumption. Moreover, this hash map not only provides a default hash function but also allows users to specify hash functions for custom data types. It permits custom hash behavior for specific data, enabling the implementation of optimal hashing strategies. Chaining is primarily employed for collision handling, and the size dynamically adjusts to automatically optimize memory usage when adding or removing data. We intend to employ this HashMaps to assess the memory computational load of each framework.

### 5.1 Performance Overhead

We evaluated the execution times of Ring, and Hashmap core logic within an Enclave, using a local environment as a baseline, without employing SGX Enclave.

Occlum performs processes by excluding the Occlum toolchain and Occlum verifier from the LibOS, instead delivering only verified MMDSFI to the LibOS. Accordingly, the necessary code (LibOS) is loaded inside the Enclave, minimizing time delays associated with context switching and exhibiting execution times similar to baseline environment. On the other hand, Fortanix EDP, which employs an intermediate Shim layer called enclave-runner to load the Enclave and handle logic processing, resulted in significantly higher program execution times. When a user invokes the enclave, the Enclave-runner inspects and sanitizes the code using the Enclave entry ABI, then loads and enters the enclave. Once inside the Enclave, after performing the logic between the enclave-runner and the Enclave, the enclave exit ABI is called to terminate the thread. Therefore, including these processes, Fortanix EDP had the longest execution times for application workloads.

Incubator Teaclave SGX SDK demonstrated the fastest execution times in the Hashmap and Ring workloads. This can be attributed to the use of a Rust wrapper optimized for the Intel SGX API, enabling faster execution even within the SGX environment, including Without SGX execution. Notably, the sgx_tcrypto used in the Ring workload called the crypto module implemented in C through unsafe calls, resulting in faster execution times. However, it did not guarantee Rust's memory safety. Therefore, Incubator Teaclave SGX SDK implements functions such as Rust's Lifetimes to ensure memory safety by automatically invoking drop functions when the lifespan of objects within sgx_tcrypto expires, securely releasing internal references to data in the $\mathrm{C} / \mathrm{C}++$ heap, without relying on unsafe calls.

In summary, the performance overhead shows that Incubator Teaclave SGX SDK, which uses SGX-optimized APIs, is the fastest, while Fortanix EDP, which utilizes the intermediate layer of enclave-runner, incurs the most significant performance overhead.

### 5.2 Enclave Size

Our goal is to evaluate the confidentiality of each framework by measuring the size of the TCB(Trusted Computing Base) that must be safeguarded within the enclave.

In the case of Occlum, we determine the enclave's size by assessing the size of the generated binary. For the Rust SGX SDK, the enclave size can be determined by examining the Enclave.so file generated during the compilation process. In the case of Fortanix EDP, the process involves converting binary files generated using Cargo into SGXS (SGX Stream) files, which adhere to the SGX enclave format. The measurement of enclave size in Fortanix EDP is based on the resulting SGXS file.


Fig. 2: Breakdown of benchmark execution time. (Figure 2a and Figure 2b represent charts illustrating the overall runtime of the frameworks and the runtime within the SGX Enclave, respectively. In particular, in the Hashmap workload, the runtime attributed to memory access increases, rendering the framework runtime itself negligible in the representation.)

The usercall API of Fortanix EDP is included within the enclave, yet it allows for the creation of the smallest possible enclave size. This is attributed to the intentional design choice of keeping the usercall API minimal, which is considered to be the reason for this outcome. The Rust SGX SDK follows the enclave design of the Intel SGX SDK but necessitates the inclusion of various Rust wrapper libraries depending on the nature of the workload. As a result, it can be observed that Fortanix EDP generates a relatively larger enclave size compared to the Rust SGX SDK.

As a result, Occlum's Enclave size is assessed as the largest among the frameworks. Occlum incorporates the entire LibOS within a single Enclave. Within the LibOS, there are components such as a binary loader for verifying whether the binary files are signed by the Occlum verifier or Occlum's encrypted file system to securely protect files, contributing to the larger Enclave size evaluation.

|  | Without SGX (baseline) | Occlum | Incubator Teaclave SGX SDK | Fortanix EDP |
| :--- | :--- | :--- | :--- | :--- |
| Framework runtime | 0.011 s | 0.011 s | 0.012 s | 0.146 s |
| Usercode Execution time | 0.0084 s | 0.0090 s | 0.0004 s | 0.0965 s |
| Enclave size | N/A | 4.4 MB | 1.4 MB | 1.18 MB |

Table 2: Hashmap workload results for each framework

## 6 Qunatifying engineering effort

To assess the qualitative effort in development, we describe the engineering effort according to the characteristics of the framework and analyze the results for Lines of Code as a factor to evaluate.

|  | Without SGX (baseline) | Occlum | Incubator Teaclave SGX SDK | Fortanix EDP |
| :--- | :--- | :--- | :--- | :--- |
| Framework runtime | 7.661 s | 7.863 s | 0.225 s | 149.037 s |
| Usercode Execution time | 7.6584 s | 7.8610 s | 0.2130 s | 148.9848 s |
| Enclave size | N/A | 4.5 MB | 1.6 MB | 1.19 MB |

Table 3: Ring(sha2) workload results for each framework

Basically, Rust SGX SDK and Fortanix EDP support utilizing the Rust standard library, and Occlum utilizes the C standard library(musl_libc and glibc). However, Rust SGX SDK and Fortanix EDP have limitations of several functionalities (e.g., environment variable, timing, networking) due to security concerns. Therefore, development costs are incurred in that developers have to implement these functions themselves to use. In contrast, Occlum not only utilizes using easy-of-use command-line tools unique to Occlum but also provides several builtin toolchains and libraries to facilitate developer porting or development tasks. Then, developers have the disadvantage of having to spend a lot of time learning about SGX SDK APIs, programming models, and systems. In addition, Fortanix EDP can implement the ability to handle memory isolation, usercalls, and SGX instruction sets by adding only std: :os::fortanix_sgx proprietary modules compared to general Rust standard libraries, and relatively reduce programmer development costs. Fortanix EDP also has the advantage of not requiring much experience from developers because it does not require SGX background knowledge and does not require EDL files to separate trust areas.

|  |  | Rust Code | EDL File <br> (ECALL/OCALL def) | Cargo.toml | Configuration <br> File |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Without SGX (baseline) |  | 12 | $\mathrm{~N} / \mathrm{A}$ | 10 | $\mathrm{~N} / \mathrm{A}$ |
| Incubator Teaclave SGX SDK | modified | 2 | $\mathrm{~N} / \mathrm{A}$ | 8 | $\mathrm{~N} / \mathrm{A}$ |
|  | add | 81 | 10 | 34 | $\mathrm{~N} / \mathrm{A}$ |
| Occlum | add | 0 | $\mathrm{~N} / \mathrm{A}$ | 0 | 17 |
| Fortanix EDP | add | 0 | $\mathrm{~N} / \mathrm{A}$ | 3 | $\mathrm{~N} / \mathrm{A}$ |

Table 4: Hashmap Workload Lines of Code

This evaluation is based on a Hashmap workload in a local environment without utilizing the SGX enclave as a reference. The results of the additional Lines of Code are summarized in Table 4 as follows. Rust's Cargo serves as a package manager for building and managing Rust applications. To build packages using Cargo, the creation of a Cargo.toml configuration file is required. Additionally, SGX also requires the Enclave.edl file with the context switch. This file defines ECALLs for entering the reserved Enclave and OCALLs for returning from the Enclave to the user space.

Rust SGX SDK provides a Rust wrapper for the Intel SGX SDK, originally written in $\mathrm{C} / \mathrm{C}++$. It uniquely distinguishes between the app and Enclave areas, necessitating the definition of the Enclave.edl file. As a result, in the main logic of the app layer, instead of using the pure Rust standard libraries, the developer employed the provided sgx_types and sgx_urts. It also, involved writing code for creating the Enclave, making function calls to enter the Enclave, executing code within the Enclave, and retrieving the results. Within the Enclave, the developer performed the Hashmap workload. Ultimately, this resulted in 2 lines being modified and an additional 81 lines of source code being written.

Occlum offers a user-friendly Occlum-cargo command to execute Rust applications, and it provides shell scripts and yaml files for this purpose. As a result, there was no need to modify or add significant code to the core logic of the Hashmap workload or the Cargo.toml file. However, there was a requirement to write 17 lines of source code for the shell scripts and yaml file.

In Fortanix EDP, a pure Rust language approach was utilized, along with a custom $\mathrm{ABI} / \mathrm{API}$, to ensure security by not exposing the Enclave interface to developers. This design choice allowed for the avoidance of writing an Enclave.edl file. The core logic of the Hashmap workload was leveraged without any modifications, thanks to the support of the Rust standard library. Instead of using a custom ABI/API, the Cargo.toml file was configured with a build target of x86_64-fortanix-unknown-sgx for building. As a result, only three lines of source code were added to the Cargo.toml file.

To minimize the developer's effort, it is evaluated as most suitable to utilize Fortanix EDP, which allows the development of applications using only the Rust language without requiring background knowledge of the SGX architecture.

## 7 Related Work

Gramine [18], previously known as Graphene, is a lightweight library operating system designed for Linux multi-process applications. This unique library OS facilitates the execution of existing applications within SGX enclaves without necessitating any modifications, except for the inclusion of an enclave manifest specifying security settings and configurations. Gramine uses this manifest to perform authenticity and integrity verification and subsequently leverages it to load the application along with its requisite dependencies.

SCONE [4] is a software platform designed for securely running containerbased applications using SGX within Docker containers. It offers a secure C standard library interface that automatically encrypts and decrypts input/output (I/O) data, thereby minimizing the performance impact of thread synchronization and system calls during the enclave transition. In addition, SCONE supports user-level threading and asynchronous system calls to improve performance.

PANOPLY [32] represents a system designed to bridge the gap between the standard OS abstraction and the specific requirements of SGX for commercial Linux applications. Inspired by the principles of micro-kernels, PANOPLY has completely rethought the logic of the OS without trying to emulate it. It achieves
this by intercepting calls to the glibc API, which allows the glibc library to reside outside the enclave's TCB. Consequently, even if the underlying OS encounters issues or malfunctions, PANOPLY ensures the application's integrity attributes remain intact, ensuring its continued proper functioning.

Among them, SCONE and PANOPLY employ thin "shim" layers that encapsulate API layers like system call tables. This architectural strategy serves the purpose of minimizing the code required within the enclave, thereby reducing both the interface's size and the potential attack surface between the enclave and the untrusted OS. Gramine, SCONE, and Panoply all represent solutions for enhancing the security of applications in container environments. They share the common characteristic of being developed in the C programming language, which means that they may not exhibit the same level of robust memory safety as the Rust-based SGX frameworks examined in this paper.

Several studies have aimed to streamline the engineering effort required for deploying applications in SGX environments, simplifying the process for developers. Glamdring [23] proposes automating the code partitioning process to utilize SGX. Once developers annotate security-sensitive data of the target application, Glamdring automatically splits the application into two sections: one for the trusted enclave and the other for the untrusted, non-enclave part. Through efficient code relocation, including the creation of SDK interface specifications and the relocation of resource-intensive features outside the enclave via runtime profiling, Glamdring minimizes the engineering effort involved.

Hasan et al. [11] conduct the comparison of the comparison between 'Port' and 'Shim' approaches for implementing SGX applications. The porting approach entails rewriting or modifying the application's code to align with the SGX environment. While it may be more complex, it typically offers superior performance. Conversely, the shimming approach involves the creation of an intermediary layer that acts as an adapter between the application and the new SGX environment. This approach requires fewer code changes due to the presence of SGX libraries but may introduce some performance overhead. The choice between 'Port' and 'Shim' hinges on various factors, including time constraints, available resources, and performance requirements, providing developers with flexibility in their approach.

Existing research on SGX-related studies for enhancing application security in container environments commonly share the characteristic of being developed in the C programming language. However, it is essential to note that, compared to the Rust-based frameworks analyzed in this paper, these solutions may not be as robust in terms of memory safety, owing to their development in $\mathrm{C} / \mathrm{C}++$. In contrast to the aforementioned studies, our studies focus on analyzing SGX frameworks that utilize the Rust programming language to enhance the security of user code and data from a memory safety perspective. Furthermore, we assess the performance of these three frameworks, each with distinct methods of supporting SGX, from the standpoint of developers. This assessment aims to provide guidelines that can promote the adoption of SGX.

## 8 Conclusion

This paper analyzes the implementation cost when developing Rust applications with existing Rust-based SGX frameworks. Through the comparative analysis over three frameworks, we confirm that Occlum has strength in performance, while developing Rust applications using Fortanix EDP is effective from the implementation cost perspective.

## References

1. Intel Software Guard Extensions (Intel SGX) SDK. https://software.intel. com/content/www/us/en/develop/topics/software-guard-extensions/sdk. html (accessed on Jun. 2021)
2. Agache, A., Brooker, M., Iordache, A., Liguori, A., Neugebauer, R., Piwonka, P., Popa, D.M.: Firecracker: Lightweight virtualization for serverless applications. In: 17th USENIX symposium on networked systems design and implementation (NSDI 20). pp. 419-434 (2020)
3. Amazon: AWS Nitro Enclaves. https://aws.amazon.com/ec2/nitro/ nitro-enclaves/
4. Arnautov, S., Trach, B., Gregor, F., Knauth, T., Martin, A., Priebe, C., Lind, J., Muthukumaran, D., O’Keeffe, D., Stillwell, M.L., Goltzsche, D., Eyers, D., Kapitza, R., Pietzuch, P., Fetzer, C.: SCONE: Secure linux containers with intel SGX. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). pp. 689-703. USENIX Association, Savannah, GA (Nov 2016), https://www.usenix.org/conference/osdi16/ technical-sessions/presentation/arnautov
5. Arnautov, S., Trach, B., Gregor, F., Knauth, T., Martin, A., Priebe, C., Lind, J., Muthukumaran, D., O’Keeffe, D., Stillwell, M.L., Goltzsche, D., Eyers, D., Kapitza, R., Pietzuch, P., Fetzer, C.: Scone: Secure linux containers with intel sgx. In: SCONE: Secure Linux Containers with Intel SGX. p. 689-703. OSDI'16, USENIX Association, USA (2016)
6. Baumann, A., Peinado, M., Hunt, G.: Shielding applications from an untrusted cloud with haven. In: 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14). pp. 267-283. USENIX Association, Broomfield, CO (Oct 2014), https://www.usenix.org/conference/osdi14/ technical-sessions/presentation/baumann
7. Durumeric, Z., Kasten, J., Adrian, D., Halderman, A., Bailey, M., Li, F., Weaver, N., Amann, J., Beekman, J., Payer, M., Paxson, V.: The Matter of Heartbleed. In: Proc. IMC. ACM (2014)
8. Fortanix: Fortanix EDP. https://edp.fortanix.com/
9. Fortanix: Fortanix sgx abi. https://edp.fortanix.com/docs/api/fortanix_sgx_ abi/
10. Han, J., Yun, I., Kim, S., Kim, T., Son, S., Han, D.: Scalable and secure virtualization of hsm with scaletrust. IEEE/ACM Transactions on Networking (2022)
11. Hasan, A., Riley, R., Ponomarev, D.: Port or shim? stress testing application performance on intel sgx. In: 2020 IEEE International Symposium on Workload Characterization (IISWC). pp. 123-133 (2020). https://doi.org/10.1109/IISWC50251.2020.00021
12. Hoekstra, M., Lal, R., Pappachan, P., Phegade, V., Del Cuvillo, J.: Using innovative instructions to create trustworthy software solutions. HASP@ ISCA 11(10.1145), 2487726-2488370 (2013)
13. Hunt, T., Song, C., Shokri, R., Shmatikov, V., Witchel, E.: Chiron: Privacypreserving machine learning as a service. arXiv preprint arXiv:1803.05961 (2018)
14. Intel: Intel MPX. https://intel-mpx.github.io/
15. Jo Van Bulck, Fritz Alder, F.P.: A case for unified abi shielding in intel sgx runtimes. In: A Case for Unified ABI Shielding in Intel SGX Runtimes. systex ' 22 (2022)
16. Jung, R., Jourdan, J.H., Krebbers, R., Dreyer, D.: Rustbelt: Securing the foundations of the rust programming language. Proc. ACM Program. Lang. 2(POPL) (dec 2017). https://doi.org/10.1145/3158154, https://doi.org/10.1145/3158154
17. Kulkarni, C., Moore, S., Naqvi, M., Zhang, T., Ricci, R., Stutsman, R.: Splinter:\{Bare-Metal\} extensions for \{Multi-Tenant\}\{Low-Latency\} storage. In: 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18). pp. 627-643 (2018)
18. Kuvaiskii, D., Kumar, G., Vij, M.: Computation offloading to hardware accelerators in intel sgx and gramine library os (2022)
19. Lankes, S., Breitbart, J., Pickartz, S.: Exploring rust for unikernel development. In: Proceedings of the 10th Workshop on Programming Languages and Operating Systems. pp. 8-15 (2019)
20. Lee, J., Jang, J., Jang, Y., Kwak, N., Choi, Y., Choi, C., Kim, T., Peinado, M., Kang, B.B.: Hacking in darkness: Return-oriented programming against secure enclaves. In: 26th USENIX Security Symposium (USENIX Security 17). pp. 523539 (2017)
21. Lee, T., Lin, Z., Pushp, S., Li, C., Liu, Y., Lee, Y., Xu, F., Xu, C., Zhang, L., Song, J.: Occlumency: Privacy-preserving remote deep-learning inference using sgx. In: The 25th Annual International Conference on Mobile Computing and Networking. pp. 1-17 (2019)
22. Liang, X., Shetty, S., Zhang, L., Kamhoua, C., Kwiat, K.: Man in the cloud (mitc) defender: Sgx-based user credential protection for synchronization applications in cloud computing platform. In: 2017 IEEE 10th International Conference on Cloud Computing (CLOUD). pp. 302-309. IEEE (2017)
23. Lind, J., Priebe, C., Muthukumaran, D., O'Keeffe, D., Aublin, P.L., Kelbert, F., Reiher, T., Goltzsche, D., Eyers, D., Kapitza, R., Fetzer, C., Pietzuch, P.: Glamdring: Automatic application partitioning for intel SGX. In: 2017 USENIX Annual Technical Conference (USENIX ATC 17). pp. 285-298. USENIX Association, Santa Clara, CA (Jul 2017), https://www.usenix.org/conference/atc17/ technical-sessions/presentation/lind
24. Matsakis, N.D., Klock, F.S.: The rust language. In: The Rust Language. p. 103-104. HILT '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2663171.2663188, https://doi.org/ 10.1145/2663171. 2663188
25. McKeen, F.X., Alexandrovich, I., Berenzon, A., Rozas, C.V., Shafi, H., Shanbhogue, V., Savagaonkar, U.R.: Innovative instructions and software model for isolated execution. In: HASP '13 (2013)
26. Microsoft: Azure confidential computing. https://azure.microsoft.com/en-us/ solutions/confidential-compute/
27. Priebe, C., Muthukumaran, D., Lind, J., Zhu, H., Cui, S., Sartakov, V.A., Pietzuch, P.R.: SGX-LKL: securing the host OS interface for trusted execution. CoRR abs/1908.11143 (2019), http://arxiv.org/abs/1908.11143
28. Rashid, F.Y.: The rise of confidential computing: Big tech companies are adopting a new security model to protect data while it's in use-[news]. IEEE Spectrum 57(6), 8-9 (2020)
29. Schwarz, M., Weiser, S., Gruss, D.: Practical enclave malware with intel sgx. In: Detection of Intrusions and Malware, and Vulnerability Assessment: 16th International Conference, DIMVA 2019, Gothenburg, Sweden, June 19-20, 2019, Proceedings 16. pp. 177-196. Springer (2019)
30. Shanker, K., Joseph, A., Ganapathy, V.: An evaluation of methods to port legacy code to sgx enclaves. In: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. pp. 1077-1088 (2020)
31. Shen, Y., Tian, H., Chen, Y., Chen, K., Wang, R., Xu, Y., Xia, Y., Yan, S.: Occlum: Secure and efficient multitasking inside a single enclave of intel sgx. In: Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems. ACM (mar 2020). https://doi.org/10.1145/3373376.3378469, https://doi.org/ 10.1145\%2F3373376.3378469
32. Shinde, S., Le, D., Tople, S., Saxena, P.: Panoply: Low-tcb linux applications with sgx enclaves (01 2017). https://doi.org/10.14722/ndss.2017.23500
33. che Tsai, C., Porter, D.E., Vij, M.: Graphene-SGX: A practical library OS for unmodified applications on SGX. In: 2017 USENIX Annual Technical Conference (USENIX ATC 17). pp. 645-658. USENIX Association, Santa Clara, CA (Jul 2017), https://www.usenix.org/conference/atc17/technical-sessions/ presentation/tsai
34. Wang, H., Wang, P., Ding, Y., Sun, M., Jing, Y., Duan, R., Li, L., Zhang, Y., Wei, T., Lin, Z.: Towards memory safe enclave programming with rust-sgx. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. p. 2333-2350. CCS '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319535.3354241, https://doi.org/10.1145/3319535.3354241
35. Wang, J., Wang, J., Fan, C., Yan, F., Cheng, Y., Zhang, Y., Zhang, W., Yang, M., Hu, H.: Svtpm: Sgx-based virtual trusted platform modules for cloud computing. IEEE Transactions on Cloud Computing (2023)

# BTFuzzer: a profile-based fuzzing framework for Bluetooth protocols 

Min Jang ${ }^{1,2}$, Yuna Hwang ${ }^{2}$, Yonghwi Kwon ${ }^{3}$, and Hyoungshick Kim ${ }^{1}$<br>${ }^{1}$ Sungkyunkwan University, Suwon, South Korea<br>\{min.jang, hyoung\}@skku.edu<br>2 Samsung Electronics, Suwon, South Korea<br>\{min.s.jang, yuna.hwang\}@samsung.com<br>${ }^{3}$ University of Maryland, College Park, Maryland, USA<br>yongkwon@umd.edu


#### Abstract

Bluetooth vulnerabilities have become increasingly popular in recent years due to, in part, the remote exploitability of Bluetooth. Unfortunately, in practice, security analysts often rely on manual analysis to identify these vulnerabilities, which is challenging. Specifically, testing various workloads while maintaining reliable Bluetooth connections between devices requires complicated network configuration settings. This paper introduces BTFuzzer, a profile-based fuzzing framework for Bluetooth devices. BTFuzzer eliminates the need for complex network configurations by feeding Bluetooth packets directly into the target device's Bluetooth library without going through the Over-TheAir (OTA) transmissions. BTFuzzer carefully crafts test inputs based on protocol profiles and specifications to maximize code coverage efficiently. Our evaluation results show that BTFuzzer is highly effective. In particular, the framework has identified two security bugs in the latest Android versions (i.e., 10 and later): CVE-2020-27024 and a publicly unknown information leak vulnerability. The first is an out-of-bounds read vulnerability (CVE-2020-27024). The second vulnerability allows attackers to connect to a victim's device and leak sensitive data without the user's awareness, as the adversary is not shown in the list of connected Bluetooth devices.


Keywords: Bluetooth • Protocol • Fuzzing • Memory Corruption • Remote Code Execution.

## 1 Introduction

Recent Bluetooth vulnerabilities such as BlueBorne [16] have sparked interest in finding Bluetooth-related security bugs due to, in part, its broad impact across multiple platforms. For example, BlueBorne affects Bluetooth implementations across multiple platforms: Android, iOS, Windows, and Linux. As of September 2023, 720 CVEs have been registered as Bluetooth-related vulnerabilities [1], where they are remotely exploitable. For instance, CVE-2017-0781 is a vulnerability in the Android's BNEP service. It allows attackers to compromise Bluetooth devices [16,25] remotely. Due to Bluetooth vulnerabilities' high and broad
security impact, security testing of the systems using Bluetooth is particularly important and critical.

Fuzzing is an automated testing approach that injects randomized inputs into a system under test to reveal vulnerabilities. For software testing, fuzzing has been successful over the years for various software systems, from OS kernels $[6,7,15]$ to robotics systems [8-11]. However, unfortunately, fuzzing network protocols such as Bluetooth is still challenging. Specifically, the Bluetooth protocol is highly dependent on complex network configurations. Conducting various tests while preserving the same network configurations and states after each test requires non-trivial effort. In addition, practical challenges such as synchronization and delay of network communication further complicate the testing process. Worse, the root causes of many vulnerabilities stem from flaws in the Bluetooth chipset firmware rather than the software stack. Hence, various firmware implementations should be taken into consideration as well. Unfortunately, existing fuzzing approaches have difficulty thoroughly testing various layers of the system such as the Bluetooth protocol layer and the application layer. For example, many existing fuzzers generate test inputs targeting device drivers, which may not even reach the application layer, which may contains various potential vulnerabilities. In other words, existing techniques may underexplore a non-trivial amount of space for Bluetooth-related vulnerabilities.

This paper introduces BTFuzzer, a fuzzing framework that automatically identifies Bluetooth vulnerabilities. While there exist approaches for identifying Bluetooth security bugs $[12,13]$, they suffer from various challenges such as (1) obtaining and maintaining complex network configurations during the test and (2) crafting complex test inputs that can penetrate various software layers without violating the constraints from device drivers, network protocol, and applications. Our approach, BTFuzzer, addresses these challenges by creating an interface to inject Bluetooth packets into the library directly. It maximizes code coverage by carefully crafting specific test inputs (e.g., Bluetooth packets) with respect to the protocol specifications such as Hand-Free Profile (HFP), Human Interface Device (HID), and Bluetooth Radio Frequency Communication (RFCOMM). The framework encompasses key components for comprehensive Bluetooth protocol fuzzing, including a packet generator, crash collector, and coverage analyzer.

To demonstrate the effectiveness of BTFuzzer, we conducted experiments on Android using open-source software. BTFuzzer found two previously unknown vulnerabilities that are exploitable in most Android devices: (1) An out-of-bounds read vulnerability (CVE-2020-27024 [24]), affecting systems running Android version 10 or later and (2) an information leak vulnerability that allows attackers to connect to a victim's device and leak data without the user's awareness as it is not visible in the list of connected Bluetooth devices.

Organization. The remainder of the paper is organized as follows: Section 2 provides background on Bluetooth and fuzzing. Section 3 introduces our proposed fuzzing framework. Section 4 presents our experimental results. Section 5 discusses related work. Section 6 concludes the paper.

## 2 Background

This section outlines the structure of Bluetooth that is essential for understanding BTFuzzer. We also provide an overview of the Bluetooth stack, Bluetooth profiles, and a generic fuzzing environment for Bluetooth protocols.

### 2.1 Bluetooth components

Figure 1 illustrates a generic Bluetooth stack. Bluetooth packets move from the baseband to Logical Link Control and Adaptation Protocol (L2CAP) via the Host Controller Interface (HCI). L2CAP then routes these packets to the next appropriate stack for each channel. The HCI packet encapsulates data for the upper protocols and profiles, including L2CAP, and the path to the upper layer varies depending on the configuration of the HCI packet. If packets can be fed directly to the HCI, a security evaluation of the Bluetooth stack can be performed without the need for complex wireless configurations.


Fig. 1: Generic Bluetooth stack.

A Bluetooth profile is a protocol that aims to provide compatibility across various devices, allowing diverse Bluetooth devices to interact with each other. While the operation method may vary among devices, functions are implemented according to specific Bluetooth profiles, enabling communication between devices with different operating systems. Packet configurations differ for each profile and conform to the forms defined in their respective specifications [2]. Vulnerabilities may arise from improper profile implementations, making generating and transmitting packets tailored to each profile crucial for effective vulnerability discovery through fuzzing.

L2CAP operates based on the channel. A channel identifier (CID) [3] is the local name representing a logical channel endpoint on the device. When a Bluetooth device makes a connection, a channel is created and a CID is assigned. Communication with the device is possible through the assigned CID and channel. CID has a namespace designated according to its purpose. The CID namespace is $0 x 0000-0 x F F F F$. In the namespace, the null identifier ( $0 x 0000$ ) is not
used, and the identifiers from $0 x 0001$ to $0 \times 003 \mathrm{~F}$ are reserved for a specific L2CAP function, which is called fixed channels. Therefore, when connected to a generic Bluetooth device, CIDs are allocated within the range of 0x0040-0xFFFF, which are called dynamically allocated channels.

L2CAP's upper layers support various protocols. Radio Frequency Communications (RFCOMM) replaces the traditional wired RS232 serial port and shares characteristics with the TCP protocol. Currently, The Headset Profile (HSP) and Handsfree Profile (HFP) are the popular profiles that use it. The Generic Attribute Profile (GATT), or often referred to as GATT/ATT, outlines how to exchange data between BLE devices using services and characteristics. It represents the highest-level implementation of the Attribute protocol (ATT). Each attribute has a 128-bit UUID and ATT-defined attributes determine characteristics and services. The Bluetooth Network Encapsulation Protocol (BNEP) enables the transmission of common networking protocols over Bluetooth and offers functionalities similar to Ethernet's. Running on BNEP, the Personal Area Networking Profile (PAN) specifies how two or more Bluetooth-enabled devices can form an ad-hoc network and access a remote network via a network access point.

### 2.2 Generic fuzzing environment for Bluetooth protocols

The fuzz testing technique is widely employed to discover security vulnerabilities [4] automatically. A fuzzer can be specialized for a specific target (e.g., a particular protocol or class of applications) or designed for a generic purpose such as AFL. To conduct a successful vulnerability discovery, understanding the characteristics of various fuzzers and selecting the most suitable one based on the target and scope of the analysis is critical.


Fig. 2: Generic fuzzing environment for Bluetooth protocols.

Traditional Bluetooth fuzz testing requires two Bluetooth-capable devices: an attacker device that sends malformed packets and a victim device that processes the packets and potentially exposes vulnerabilities. The attacker device must maintain a state where it can send and receive packets. It must also implement a fuzzing engine with three functions: (1) Generating malformed packets (1), (2) Establishing a Bluetooth connection (2), and (3) Sending the malformed packets (3). The victim device must process packets (4) and detect crashes (5).

Setting up this environment is time-consuming and complex, as it essentially requires constructing the entire system, including the network environment.

The Bluetooth software stack processes packets sent over-the-air (OTA) via the Bluetooth firmware on the target device. In OTA-based fuzzing, whether specific packets reach the Bluetooth software stack may depend on the firmware configuration of the Bluetooth chipset. This environment is more suited for Bluetooth firmware code analysis and has limitations for Bluetooth software stack vulnerability analysis.

BTFuzzer simplifies the fuzz testing process by directly transmitting packets to the victim device, bypassing the wireless environment. This approach allows quicker fuzz testing and eliminates the need for the packets to go through the Bluetooth firmware before reaching the software stack. BTFuzzer proposes an automated method to identify logical errors within the Bluetooth software stack.

## 3 Proposed system

In this section, we explain how to fuzz the Bluetooth stack using the proposed fuzzing framework, BTFuzzer.

### 3.1 Overview



Fig. 3: Overview of BTFuzzer.

We propose a new fuzzing framework, BTFuzzer, which directly feeds packets into the target device, bypassing OTA. BTFuzzer generates packets and defines an interface for direct input into the device's HCI layer. Figure 3 provides an overview of the proposed system. This configuration allows direct access to the Bluetooth software stack for fuzz testing on profiles and protocols with independent specifications. We note that the framework is highly configurable, meaning that it can easily customized to support fuzz testing on diverse profiles and even other protocols of interest.

### 3.2 Fuzzing interface

We create a specialized fuzzing interface to feed packets directly into the device. In particular, based on our analysis of the Android Open Source Project (AOSP) Bluetooth stack, we implement our fuzzing interface in libbluetooth.so. ${ }^{4}$

The hci_initialize function within hci_layer_android.cc initializes the HCI and creates (1) a fuzzing interface thread and (2) a socket for communication with the fuzzing client. This client then feeds commands and packets from the fuzzing server into the interface through the socket.

HCI Handles and L2CAP CIDs are essential for generating valid Bluetooth packets. The interface receives and processes predefined commands from the client to obtain these values. The currently connected Handles and CIDs are saved, and the gathered Handles and CIDs are used for packet creation. Additionally, HCI packets fed into the interface are categorized into four types for processing: COMMAND, ACL, SCO, and EVENT. Figure 4 illustrates the architecture of the fuzzing interface within the AOSP device.


Fig. 4: Composition of fuzzing interface.

### 3.3 Fuzzing server

The fuzzing server consists of the following three modules:

- Packet generator: This module creates a large corpus of malformed packets by randomly injecting errors into valid packets. This addresses performance degradation when feeding individual packets to the Android device via ADB. This ensures that the fuzzing process covers a wide range of possible inputs. The corpus is transferred to the Android device using the adb push command.

[^8]- Crash collector: This module collects crashes that occur during fuzzing.
- Coverage analyzer: This module analyzes the coverage of the Bluetooth software stack during fuzzing.

HCI handles, and L2CAP CIDs are assigned when a Bluetooth device is connected. However, these values may change if the device is reconnected after a crash. This requires the packet generator to regenerate the packets. Additionally, the device's Bluetooth settings may change due to previous packets. To mitigate these issues, the fuzzing server initializes the Bluetooth stack before starting the fuzzing process. This ensures that HCI handles and L2CAP CIDs remain constant, allowing the use of pre-made packets even after a crash.

### 3.4 Fuzzing client

The fuzzing client is specialized for interaction with the fuzzing interface, implemented in the libbluetooth.so library. This client is an executable file that establishes a connection to the fuzzing interface's socket. It reads from the corpus file located at a predefined path and sequentially sends packets into the Bluetooth stack via this socket. Essentially, the fuzzing client is responsible for sending malformed packets to the Android device for testing.

Figure 5 illustrates the architecture of the fuzzing client, showcasing its various components and their interaction with the fuzzing interface. This helps to understand the role of the fuzzing client in the overall architecture of BTFuzzer, highlighting its critical role in injecting malformed packets into the system to identify vulnerabilities.


Fig. 5: Composition of the fuzzing client.

### 3.5 Packet generator

This paper focuses on fuzzing three key Bluetooth protocols commonly used in smartphones: RFCOMM, HFP, and HID. These were selected because they are essential for core smartphone functions and have significant security implications.

- RFCOMM is a simple, reliable data stream to which other applications can connect as if they were serial ports. It is one of the foundational profiles used in most Bluetooth devices, meaning that it is an essential test subject.
- HFP is crucial for enabling smartphone call functionalities. Given that calling is a core function of smartphones and a profile used daily by many users, any vulnerabilities in HFP could have significant security implications, such as the potential for eavesdropping.
- HID is related to input devices such as keyboard and mouse. Vulnerabilities in HID could allow an attacker to remotely control the victim's device, making it critical for security analysis.

To generate test cases for these profiles, we have implemented two different types of packet generation techniques: mutation-based and profile-based.

First, the mutation-based packet generator takes existing valid Bluetooth packets and modifies them in various ways to create malformed packets. These malformed packets are then used to test how well the Bluetooth stack can handle unexpected or non-standard data.

Second, the profile-based packet generator creates packets according to the specifications of the target Bluetooth profiles (RFCOMM, HFP, and HID). By adhering closely to the specifications, we can test for vulnerabilities caused by wrong implementations of the protocols.

By combining the two different packet generation techniques, BTFuzzer aims to achieve a comprehensive set of test cases that can thoroughly evaluate the robustness and security of Bluetooth implementations in Android devices.


Fig. 6: Composition of the packet generator.

Mutation-based packet generation. Mutation-based packet generation creates new packets through mutation, using packets transmitted and received between devices to enhance code coverage. Base packets are obtained from Android Bluetooth snoop logs [18]. Bluetooth HCI Snoop is specified in RFC 1761 [17]. A simple script was developed to parse these Snoop logs into a mutational hex
format. Pyradamsa is used to mutate the parsed packets. A base packet is selected for mutation. A packet must be generated with a matching HCI Handle and L2CAP CID to facilitate normal communication and data processing. In mutation-based generation, packets are created using two methods. The first method sequentially writes and mutates the entire set of recorded packets. The second method randomly selects a packet for mutation.

Profile-based packet generation. Profile-based packet generation produces packets tailored for specific Bluetooth profiles and protocols. Target profiles and protocols were selected, and their specifications were analyzed. We examined the specifications for three items: HFP [19], HID [20], and RFCOMM [21]. Payloads for each item are generated using Python's random library. Like in mutationbased generation, the HCI and L2CAP portions, excluding the payload, utilize the allocated HCI Handle and L2CAP CID. Packets, including the generated payload, are generated with matching HCI and L2CAP lengths.


Fig. 7: Structure of HCI and L2CAP packets.

Figure 7 illustrates the basic structure of HCI and L2CAP packets. The type field in HCI packets consists of one octet and classifies COMMAND, ACL, SCO, and EVENT types. The handle field, comprising two octets, holds connection information between devices. The Length field, also of two octets, specifies the total length of the HCI packet. If the length field value does not match the packet length, Android Bluetooth HCI will immediately abort the connection. Therefore, it is crucial to calculate and set the correct length and handle values when generating a packet. Detailed specifications for HID, HFP, and RFCOMM, along with their implementation in BTFuzzer, are outlined below.

Figure 8 depicts the packet structure of HID. The Header field contains HID Header information in one octet. Only HANDSHAKE, HID_CONTROL, and DATA Message types are used for packet generation. These types facilitate data transmission from HID to the smartphone. The payload part consists of randomly generated data, varying in size from $0 x 00$ to $0 x F F$.


Fig. 8: Packet structure of HID.

Figure 9 shows the packet structure of RFCOMM. The Address field, consisting of one octet, contains the DLCI (Data Link Connection Identifier) or the connection information for RFCOMM. To transmit data correctly, this address value must be set accurately, which can be retrieved from Bluetooth logs. The Control field is one octet and includes frame type and poll/final bit information. Depending on the payload size, the Length field consists of one or two octets. If the payload size exceeds 127 bytes, two octets are used. The Payload field is filled with random values, and its size determines the Length field. Finally, the FCS field, comprised of one octet, is used for CRC (Cyclic Redundancy Check). It is calculated based on predefined CRC table values, Address, and Control fields.


Fig. 9: Packet structure of RFCOMM.

Figure 10 presents the packet structure of HFP. The payload field is the only variable part based on AT Commands from the RFCOMM packet structure. We extracted a list of usable AT Commands from Android Bluetooth code and configured the system to randomly generate payloads for each AT Command.

### 3.6 Crash collector

When a crash occurs during fuzzing, the crash collector gathers and stores relevant information. On Android devices, Signals 6 and 11 automatically generate tombstone files. The crash collector checks whether a tombstone file is created during fuzzing. If created, it collects the tombstone file from the Android device. The generated corpus, handle, and CID information are stored to facilitate crash reproduction. Figure 11 illustrates the components of the crash collector.


Fig. 10: Packet structure of HFP.

| AT Command List |  |  |
| :---: | :---: | :---: |
| AT+VGS | AT+VGM | AT |
| AT+CHLD | AT+CHUP | AT+CIND |
| AT+CLIP | AT+CMER | AT+VTS |
| AT+BINP | AT+BLDN | AT+BVRA |
| AT+BRSF | AT+NREC | AT+CNUM |
| AT+BTRH | AT+CLCC | AT + COPS |
| AT+CMEE | AT+BIA | AT+CBC |
| AT +BCC | AT+BCS | AT+BIND |
| AT+BIEV | AT +BAC |  |

Table 1: List of AT Commands used for packet generation.


Fig. 11: Composition of the crash collector.

### 3.7 Coverage analyzer

To measure the coverage of the code, the coverage analyzer inserts log codes into all AOSP Bluetooth stack files. To avoid duplicates, the log format is set as FUZZ_COVERAGE _FileName_Count. For automated log insertion, we developed a Python script. Once log code insertion is complete, the number of logs added to


Fig. 12: Composition of the coverage analyzer.
each file and the total log count are recorded. By comparing the number of output logs during fuzzing with the total number of logs, we can assess the extent of code execution. Logs are inserted to identify most branching statements, allowing efficient code coverage measurement for libbluetooth.so. Figure 12 illustrates the structure of the coverage analyzer.

## 4 Evaluation

BTFuzzer was tested on a Pixel 3a device running Android 10. During the evaluation, it was paired with a Galaxy Watch, Galaxy Buds, a Bluetooth keyboard, and a Bluetooth mouse. Fuzzing was conducted after analyzing the packets obtained during basic interactions between the Pixel 3a and each Bluetooth device. After that, random packets were generated for fuzzing. The profiles evaluated were RFCOMM, HID, and HFP. To assess BTFuzzer's effectiveness, we applied it to the binary code before patching the vulnerability known as BlueFrag (CVE-2020-0022) [22,23], one of the most critical Android Bluetooth vulnerabilities of 2020.

BTFuzzer discovered two vulnerabilities that could affect most Android devices, including the latest version. One was reported to the Google Android Security Team and recognized as a new vulnerability under the identifier A-182388143. The other was reported as A-182164132 but was marked as a duplicate of A-162327732, which has been assigned CVE-2020-27024 [24]. The BlueFrag vulnerability, for which the patch had been removed, was also detected.

The code coverage of BTFuzzer was assessed using the coverage analyzer. When delivering packets generated specifically for a particular profile, it was observed that the code coverage corresponding to that profile increased significantly. This observation validates the effectiveness of profile-based fuzz testing.

### 4.1 Hiding the list of malicious Bluetooth devices

We discovered a new vulnerability in the Bluetooth stack of Android devices. This vulnerability allows attackers to manipulate the list of Bluetooth-connected devices on a victim's device. The vulnerability, which is assigned to the identifier A-182388143. It was discovered by using RFCOMM profile-based fuzz testing with BTFuzzer. The Google Android Security Team has confirmed it as a security vulnerability.

The vulnerability can be exploited on most Android devices, including the latest version. An attacker could use this flaw to hide a malicious Bluetooth device connected to the user's device, making it undetectable to the user. Consequently, the attacker could access contacts and SMS messages or intercept calls without the user noticing the attacker's activities. The vulnerability can be exploited by sending just one malicious packet to the user's device.


Fig. 13: Result of the attack that exploited the A-182388143 vulnerability on Google Pixel 3a. The connected devices list is shown before (a) and after (b) the attack. Galaxy Buds are initially displayed in the connected list before the attack but not in the connected list after the attack. This is because the attacker was able to remove Galaxy Buds from the list by exploiting the vulnerability.

As shown in Figure 13, we can see the Bluetooth device in the connected list before the attack is performed. However, after the attack is performed, the Bluetooth device is not visible in the connected device list even though the device can still maintain the connection with the victim device. This vulnerability was discovered while fuzzing RFCOMM. It was possible to trigger the vulnerability through a specific packet generated by BTFuzzer's Profile-based. This attack can hide the device by sending only one simple packet. We received a 2,000 USD reward from the Google Android Security Team for reporting this vulnerability. However, this vulnerability has not been patched yet and detailed information cannot be disclosed to prevent malicious exploitation.

14 Jang et al.

### 4.2 Buffer overflow vulnerabilities

CVE-2020-0022. To demonstrate BTFuzzer's effectiveness, we conducted fuzzing tests on a binary containing the BlueFrag vulnerability, a significant Android Bluetooth vulnerability from 2020. Our goal is to evaluate whether BTFuzzer can find a known vulnerability effectively. Just less than 5 minutes, BTFuzzer detected the CVE-2020-0022 vulnerability. Figure 14 displays the crash $\log$ for this vulnerability, triggered by BTFuzzer.

```
pid: 7221, tid: 10924, name: bt_hci_thread >>> com.android.bluetooth <<<
uid: 1002
signal 11 (SIGSEGV), code 2 (SEGV_ACCERR), fault addr 0x6ff73ffff0
backtrace:
    #00 memcpy+104
    #01 reassemble_and_dispatch(BT_HDR*) [clone .cfi]+948
```

Fig. 14: CVE-2020-0022 crash log.

A-182164132. The out-of-bounds vulnerability was discovered through the BTFuzzer, and the vulnerability was reported to A-182164132. However, it was already reported as a vulnerability with A-162327732. This vulnerability has been assigned CVE-2020-27024. CVE-2020-27024 is a vulnerability that can cause out-of-bounds read due to a missing boundary check in smp_br_state_machine_ event() of smp_br_main.cc, Figure 15 shows the CVE-2020-27024 vulnerability crash log triggered via BTFuzzer. The vulnerability (i.e., related to the missing boundary check) is mitigated through Bounds Sanitizer, which is supported from Android 10. However, it can be still exploited in the previous Android versions or customized/specialized Android systems forked from the previous Android versions. This vulnerability can be attacked when the connection handle is $0 x 02$. Figure 16 shows packets that can reproduce CVE-2020-27024. Sending these two packets could trigger the CVE-2020-27024 vulnerability.

### 4.3 Coverage

Code coverage was measured using a log-based approach, in which $31,997 \operatorname{logs}$ were instrumented into the Android Bluetooth-related code. Fuzzing was carried out for 24 hours for each of the three methods used to generate packets: mutationbased, profile-based, and RFCOMM, HFP, and HID. The code coverage was then measured after each fuzzing run.

Figure 17 shows the change in code coverage over time during fuzzing. Figure 17 (a) shows the total coverage for the 24 hours, which reveals an initial rapid increase followed by a slower growth rate. Figure 17(b) focuses on the first 10

```
pid: 3753, tid: 3802, name: bt_main_thread >>> com.android.bluetooth <<<
uid: 1002
signal 6 (SIGABRT), code -1 (SI_QUEUE), fault addr --------
Abort message: 'ubsan: out-of-bounds'
backtrace:
    #00 abort+160
    #01 abort_with_message(char const*)+20
    #02 __ubsan_handle_out_of_bounds_minimal_abort+24
    #03 smp_br_state_machine__event(tSMP_CB*, unsigned char, tSMP_INT_DATA*)+1212
```

Fig. 15: CVE-2020-27024 crash log.
[Packet 1]
0202206900650006000befbb0057fe410e98007b22726573f3a0819e756c74223a22737563
63657373222c22726561736f6e223a302c226d73674964223a226d757369632d7175657565
6368616e6765642d696e64222c22636f756e74223a302c226c697374223a5b5d7dce249a
[Packet 2]
0202201500110007000bff1702000503c01af001c08007c07a86

Fig. 16: CVE-2020-27024 trigger packets.
minutes of this period, demonstrating a similar trend: an initial swift rise in coverage that eventually plateaus.


Fig. 17: Code coverage changes over time. (a) represents the 24-hour coverage for mutation, RFCOMM, HID, and HFP methods. (b) represents the coverage changes during the first 10 minutes of the 24 -hour period.

Figure 18(a) and (b) present the coverage results of mutation-based and profile-based (HFP, RFCOMM, HID) fuzzing, respectively. Figure 18(a) illustrates the outcomes of profile-based fuzzing, where "Total" denotes the combined
log results for HID, HFP, and RFCOMM, exceeding the individual log count for RFCOMM, the highest among them. Each method executed distinct code segments. Figure 18(b) contrasts mutation-based and profile-based fuzzing. Tests conducted on the same three types of Bluetooth devices (Galaxy Watch, Galaxy Buds, and a Bluetooth keyboard and mouse) showed that profile-based fuzzing achieved more code coverage than mutation-based fuzzing. Although profilebased fuzzing offers more code coverage, it requires understanding the profile and creating a packet structure code that aligns with the profile. Conversely, mutation-based fuzzing, while achieving less code coverage than profile-based fuzzing, allows fuzzing without profile comprehension. More importantly, each method executed different code segments, indicating that the two methods are complementary and could maximize fuzzing code coverage when combined.

Out of the 31,997 logs instrumented, 6,914 were recorded, representing approximately $21.6 \%$ of the total code coverage. Enhanced results are expected with further profile/protocol testing.


Fig. 18: Coverage results for 31,997 instrumented logs. (a) represents the coverage of profile-based fuzzing, and (b) compares mutation-based and profile-based methods.

### 4.4 Summary of evaluation results

BTFuzzer is an effective tool for finding vulnerabilities in Android Bluetooth stacks. It found two vulnerabilities in the Pixel 3a, one of which was a new vulnerability that allowed attackers to hide a Bluetooth device in the list of Bluetooth-connected devices on a victim's device. BTFuzzer also detected the CVE-2020-0022 vulnerability, a significant Android Bluetooth vulnerability from 2020, in less than 5 minutes.

Our evaluation also shows that BTFuzzer's profile-based fuzzing is more effective than mutation-based fuzzing at achieving more code coverage. However,
each approach targeted different code segments, meaning that they are complementary. We believe that combining both techniques could maximize fuzzing code coverage.

## 5 Related work

Research on Bluetooth security is diverse, covering topics such as attacks via malicious devices, vulnerabilities in protocol implementations, and methodologies for vulnerability analysis, including active fuzzing studies.

One approach focuses on exploiting the Bluetooth function by taking control of Bluetooth communication authority. Xu et al. [5] describe an attack that leverages a device's inherent trust in an already-connected Bluetooth device. This research suggests that devices better manage Bluetooth function authority, pairing conditions, and the intent of paired devices. A more straightforward method of identifying vulnerabilities is to analyze Bluetooth protocol implementations. A notable example is BlueBorne [16], published by ARMIS Lab in 2017, which examined Bluetooth specifications and identified vulnerabilities and logical errors. However, auditing the code for the entire Bluetooth specification and its various profiles is challenging.

Another technique to consider is fuzzing. Mantz et al. [13] introduced a versatile framework for finding vulnerabilities in Bluetooth firmware. Ruge et al. [12] proposed an advanced, firmware emulation-based fuzzing framework for undisclosed Bluetooth implementations and firmware. However, these studies focus on chipset firmware-level security evaluation, not the Bluetooth software stack. Heinze et al. [14] recently suggested a fuzzing approach targeting specific L2CAP Channels in Apple's private Bluetooth stack.

We propose a new approach: a profile-based fuzzing framework for the Bluetooth stack. This framework facilitates creating and fuzzing packets for each Bluetooth profile, enabling comprehensive coverage of various protocols and profiles within the Bluetooth stack.

## 6 Conclusions

As Bluetooth technology becomes ubiquitous and its applications span multiple devices and functionalities, vulnerabilities in Bluetooth technology have become high-impact security risks. Despite ongoing research to enhance Bluetooth security, new vulnerabilities continue to be discovered and exploited, demanding a systematic approach to search for vulnerabilities effectively.

We introduce BTFuzzer, a scalable, profile-based fuzzing framework for Bluetooth devices. BTFuzzer implements in-device packet transmission, eliminating the need for complex environment setup. It generates packets according to specific Bluetooth profiles to maximize code coverage. BTFuzzer has identified a new vulnerability that allows an attacker's Bluetooth device to remain concealed while connected to a victim's device. Additionally, BTFuzzer has demonstrated its efficacy by detecting previously disclosed Bluetooth vulnerabilities.

BTFuzzer is a generic approach and not limited to Android. It is highly configurable, meaning that it can be easily configured to support other operating systems and protocols. Our preliminary results indicate that BTFuzzer is compatible with Linux Bluez, making it a viable tool for evaluating vulnerabilities in the Linux Bluetooth software stack. Further experimentation with the multitude of Bluetooth profiles will enhance code coverage and enable the discovery of additional vulnerabilities. We plan to expand our research to other operating systems, Bluetooth profiles, and other wireless technologies such as NFC, Wi-Fi, and Zigbee to improve wireless network security.

Acknowledgements We thank our anonymous shepherd and reviewers for their valuable feedback and insights. Hyoungshick Kim is the corresponding author. This work was supported by Institute for Information \& communication Technology Planning \& Evaluation grant funded by the Korea government (No.2018-0-00532, Development of High-Assurance (>=EAL6) Secure Microkernel (50\%), No.2022-0-00495 (30\%), and No.2022-0-01199 (20\%)).

## References

1. MITRE, https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=bluetooth. Accessed on September. 15, 2023
2. Bluetooth Specifications, https://www.bluetooth.com/specifications/specs/. Accessed on September. 15, 2023
3. BLUETOOTH CORE SPECIFICATION Version 5.2, page 1026-1028, https: //www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc \_id=478726. Accessed on September. 15, 2023
4. Valentin Jean Marie Manès and HyungSeok Han and Choongwoo Han and Sang Kil Cha and Manuel Egele and Edward J. Schwartz and Maverick Woo.: The Art, Science, and Engineering of Fuzzing: A Survey. IEEE Transactions on Software Engineering (2019)
5. Fenghao Xu, Wenrui Diao, Zhou Li, Jiongyi Chen and Kehuan Zhang.: BadBluetooth: Breaking Android Security Mechanisms via Malicious Bluetooth Peripherals. Network and Distributed System Security Symposium (NDSS 2019)
6. Seulbae Kim, Meng Xu, Sanidhya Kashyap, Jungyeon Yoon, Wen Xu and Taesoo Kim.: Finding Semantic Bugs in File Systems with an Extensible Fuzzing Framework. ACM Symposium on Operating Systems Principles (SOSP 2019)
7. Sergej Schumilo, Cornelius Aschermann, and Robert Gawlik, Ruhr-Universität Bochum; Sebastian Schinzel, Münster University of Applied Sciences; Thorsten Holz, Ruhr-Universität Bochum.: kAFL: Hardware-Assisted Feedback Fuzzing for OS Kernels. USENIX Security Symposium (USENIX Security '17)
8. Hyungsub Kim, Muslum Ozgur Ozmen, Antonio Bianchi, Z. Berkay Celik, and Dongyan Xu.: PGFUZZ: Policy-Guided Fuzzing for Robotic Vehicles. Network and Distributed System Security Symposium (NDSS 2021)
9. Taegyu Kim, Purdue University; Chung Hwan Kim and Junghwan Rhee, NEC Laboratories America; Fan Fei, Zhan Tu, Gregory Walkup, Xiangyu Zhang, Xinyan Deng, and Dongyan Xu, Purdue University.: RVFuzzer: Finding Input Validation Bugs in Robotic Vehicles through Control-Guided Testing. USENIX Security Symposium (USENIX Security '19)
10. Seulbae Kim, and Taesoo Kim.: RoboFuzz: fuzzing robotic systems over robot operating system (ROS) for finding correctness bugs. ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2022)
11. Chijung Jung, Ali Ahad, Yuseok Jeon and Yonghwi Kwon.: SWARMFLAWFINDER: Discovering and Exploiting Logic Flaws of Swarm Algorithms. IEEE Symposium on Security and Privacy (SP 2022)
12. Jan Ruge, Jiska Classen, Francesco Gringoli and Matthias Hollick.: Frankenstein: Advanced Wireless Fuzzing to Exploit New Bluetooth Escalation Targets. USENIX Security Symposium (USENIX Security '20)
13. Dennis Mantz, Jiska Classen, Matthias Schulz and Matthias Hollick.: InternalBlue - Bluetooth Binary Patching and Experimentation Framework. ACM International Conference on Mobile Systems, Applications, and Services (MobiSys 2019)
14. Dennis Heinze, Matthias Hollick and Jiska Classen.: ToothPicker: Apple Picking in the iOS Bluetooth Stack. USENIX Workshop on Offensive Technologies (WOOT '20)
15. syzkaller, https://github.com/google/syzkaller. Accessed on September. 15, 2023
16. BlueBorne, https://www.armis.com/blueborne/. Accessed on September. 15, 2023
17. RFC 1761, https://tools.ietf.org/html/rfc1761. Accessed on September. 15, 2023
18. Android Bluetooth Verifying and Debugging, https://source.android.com/ devices/bluetooth/verifying_debugging\#debugging-with-logs. Accessed on September. 15, 2023
19. Hands-Free Profile 1.8, https://www.bluetooth.com/specifications/specs/ hands-free-profile-1-8/. Accessed on September. 15, 2023
20. Human Interface Device Profile 1.1.1, https://www.bluetooth.com/ specifications/specs/human-interface-device-profile-1-1-1/. Accessed on September. 15, 2023
21. RFCOMM 1.2, https://www.bluetooth.com/specifications/specs/rfcomm-12/. Accessed on September. 15, 2023
22. BlueFrag, https://insinuator.net/2020/04/cve-2020-0022-an-android-8-0-9-0-bluetooth-zero-click-rce-bluefrag/. Accessed on September. 15, 2023
23. CVE-2020-0022, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-0022. Accessed on September. 15, 2023
24. CVE-2020-27024, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-27024. Accessed on September. 15, 2023
25. CVE-2017-0781, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0781. Accessed on September. 15, 2023

# mdTLS: How to make middlebox-aware TLS more efficient? 

Taehyun Ahn ${ }^{[0009-0007-2339-286 X]}$, Jiwon Kwak ${ }^{[0009-0008-1560-7442]}$, and Seungjoo Kim ${ }^{\star}[0000-0002-2157-0403]$<br>School of Cybersecurity, Korea University, Seoul 02841, South Korea<br>\{thyun_ahn,jwkwak4031,skim71\}@korea.ac.kr


#### Abstract

Recently, many organizations have been installing middleboxes in their networks in large numbers to provide various services to their customers. Although middleboxes have the advantage of not being dependent on specific hardware and being able to provide a variety of services, they can become a new attack target for hackers. Therefore, many researchers have proposed security-enchanced TLS protocols, but their results have some limitations. In this paper, we proposed a middlebox-delegated TLS (mdTLS) protocol that not only achieves the same security level but also requires relatively less computation compared to recent research results. mdTLS is a TLS protocol designed based on the proxy signature scheme, which requires about $39 \%$ less computation than middlebox-aware TLS (maTLS), which is the best in security and performance among existing research results. In order to substantiate the enhanced security of mdTLS, we conducted a formal verification using the Tamarin. Our verification demonstrates that mdTLS not only satisfies the security properties set forth by maTLS but also complies with the essential security properties required for proxy signature scheme. ${ }^{1}$


Keywords: maTLS • Middlebox • Proxy signature • Formal verification

## 1 Introduction

The advent of the COVID-19 pandemic has instigated substantial transformations in the business landscape. Notably, a significant proportion of enterprises have transitioned from conventional in-office working arrangements to facilitating remote work options for their workforce. Concurrently, the pandemic has spurred innovative shifts in operational methodologies, exemplified by the substitution of face-to-face business procedures, historically reliant on in-person meetings, with video conferencing solutions. As a result of these shifts, there has been a discernible escalation in network traffic, with notable statistics from the Telegraph indicating a remarkable $47 \%$ surge in internet traffic between 2019 and 2020 [28].

[^9]Especially during the COVID-19 pandemic, the security of confidential information of various companies and individuals has been emphasized as most social activities, including business, are conducted remotely over the network. Among the most prominent and widely adopted technologies addressing network security concerns during this period is HTTPS (HyperText Transfer Protocol Secure) [36].

HTTPS represents a communication protocol that integrates the HTTP (HyperText Transfer Protocol) [13] to the TLS (Transport Layer Security) protocol [10], with the overarching objective of ensuring the confidentiality and integrity of data transmitted over networks. This protocol finds utility not only in desktops but extends its application domain to encompass a diverse array of embedded devices, including IoT (Internet of Things) devices. HTTPS offers several fundamental security attributes, including the following:

- Encryption: It serves as a pivotal mechanism within HTTPS, facilitating the obfuscation of sensitive information by encoding the data exchanged between communicating entities. Commonly employed encryption algorithms encompass symmetric key algorithms like Advanced Encryption Standard (AES) [19].
- Authentication: It constitutes an integral component of HTTPS, operating to ascertain the identity of entities by utilizing digital certificates.
- Integrity: It is another crucial facet of HTTPS, operating as a mechanism to detect unauthorized tampering or forgery of messages. Conventional algorithms used to maintain message integrity involve the implementation of Message Authentication Codes (MACs), such as the Secure Hash Algorithm (SHA) [9], to uphold the veracity and unaltered state of a network connection.

According to the Google transparency report, there has been a consistent increase in the loading speed of HTTPS pages in the chrome browser since 2014 [17]. Moreover, among the top 100 non-Google websites on the internet, which collectively constitute approximately $25 \%$ of global website traffic, 96 websites have embraced HTTPS, with 90 of them making HTTPS their default protocol. Additionally, according to Gartner's article [34], edge computing technology is anticipated to evolve into a core IT technology. This technology facilitates the secure communications of data collected through embedded systems deployed across various domains, relying on TLS protocols. Consequently, TLS communication is expected to assume an increasingly pivotal role. However, the robust encryption mechanisms employed by TLS to protect data can also be exploited by attackers to hide malware within network traffic, thereby evading detection by conventional security measures. In fact, according to research by Cisco and Sophos, TLS is vulnerable to detecting malicious traffic, and the number of such cases continues to increase [5, 14]. As a result, TLS cannot be considered a complete solution against cybersecurity threats.

For this reason, numerous organizations have deployed specialized middleboxes with distinct functionalities designed to enhance security for their clients, such as firewall and intrusion detection [39]. For instance, some companies have integrated Transport Layer Security Inspection (TLSI) [30] capabilities into middleboxes to identify and intercept malicious traffic attempting to infiltrate their internal networks. TLSI represents a technology devised to thwart unauthorized actions perpetrated by hackers on encrypted network traffic, and numerous entities, including industry giants such as Microsoft, are actively leveraging this technology [27].

However, according to a survey conducted in the United States, more than $70 \%$ of employees still believe that hackers can exploit middleboxes. Also, $50 \%$ of the respondents answered that their personal information could be infringed by exploiting vulnerabilities in the middleboxes [33]. Ironically, middleboxes, initially installed to fortify data security within TLS communications, have emerged as potential targets for cyberattacks. Consequently, safeguarding data transmitted over TLS communications necessitates a holistic approach considering network components, such as middleboxes, from the inception of communication channel construction. This approach goes beyond simply installing securityhardened components into an existing network.

As a consequence, numerous researchers have proposed a range of TLS extension protocols to enhance security during communication via the TLS protocol. However, prior research endeavors, driven primarily by a pursuit of security, have inadvertently encountered performance-related challenges. In this study, we will introduce the mdTLS protocol, which is meticulously designed based on the proxy signature scheme. The mdTLS is subject to comparative evaluation against maTLS [24], widely recognized as the most exemplary among prior researches in terms of both security and performance. First, we investigated the amount of arithmetic operations that must be performed for each designed protocol to compare the performance of the mdTLS and maTLS protocols. We then formally verified that the mdTLS satisfies not only the security properties verified in maTLS, but also three other security properties related to the proxy signature scheme. To ensure methodological consistency in our experimental setup, we employed the Tamarin $[26,37,40]$, utilized in prior maTLS research, during the security analysis.

The remainder of the paper is organized as follows. First, we analyzed the strengths and weaknesses of related works (Section 2). Next, we introduced our mdTLS protocol (Section 3). After that, we compare the performance between maTLS and mdTLS (Section 4). In Section 5, we verified our protocol using Tamarin (Section 5). We showed that the performance can be further improved when the Schnorr digital signature is used in the protocol (Section 6). Finally, we present our concluding remarks (Section 7).

## 2 Related works

Many researches have been conducted to improve TLS protocol. They are categorized into two types. One is the TLS-encryption extension-based approach. Their research is to improve the mechanism itself inside the protocol. The other one is the Trusted Execution Environment (TEE) based approach. Their research is to improve the protocol by using specific hardware.

### 2.1 TEE based approaches

A typical example of the Trusted Execution Environment (TEE) based approach is SGX-Box [18]. It utilized the remote attestation of Intel SGX. The server performs remote attestation to verify the integrity of the SGX-Box module in middleboxes. If remote attestation succeeds, they create a secure channel to prevent sensitive information from leaking between them. However, it is limited in that it is too dependent on its specific hardware (Intel SGX). Besides SGXBox, there are many researches such as STYX [42], EndBox [16], and ShieldBox [41]. However, they also had the same limitations mentioned above.

### 2.2 TLS-extension based approaches

A typical example of the TLS-extension approach is SplitTLS [20]. In SplitTLS, middleboxes act as servers and clients at the same time. This feature gives them too many privileges. It can cause some security incidents. For example, middleboxes such as CDN service providers could receive the private key to act as a server. It accidentally exposes the private key during the key-exchange phase. The worst thing is that when the middleboxes become compromised, malicious users (attackers) could abuse their privileges. Unlike SplitTLS, mcTLS [32] provides the least privilege to middleboxes. Middleboxes can read or write the TLS payload by obtaining MAC key pair from each endpoint. For example, they can only read the TLS packets when they get a unique key for reading. The advantage of mcTLS is that it does not force middleboxes to create or install further objects. Since the mcTLS uses only one key when creating a session, it is considered insecure. In the performance view, it has a limitation in that additional latency occurs when establishing the first connection. Furthermore, it does not follow TLS standards. David Naylor, who had proposed mcTLS, proposed an extended version of mcTLS called mbTLS [31]. mbTLS was created to improve compatibility with TLS standards. mbTLS establishes two types of sessions. One is the mbTLS session, and the other is the standard TLS session. If one of the endpoints does not use mbTLS, then traditional TLS sessions are activated. Overall, mbTLS offers improvements over mcTLS, which causes latency when adding a secondary session. maTLS [24] is another extended protocol to address security issues in SplitTLS. It treats middleboxes as equivalent entities to the server and includes them in the TLS session. As the server's certificate, middleboxes' certificates are issued by the Certificate Authority (CA), and by
introducing the Middlebox Transparency (MT) log server, the middleboxes certificate contains a Signed Certificate Timestamp (SCT) [2,23]. This guarantees middleboxes' audition and improves the reliability of the middleboxes' certificates. Also, unlike SplitTLS, this procedure shows middleboxes can create their own official certificates without using custom root certificates or server certificates. However, these security elements entail performance issues. To make every session in each section, maTLS handshakes are essential between every entity. This is why maTLS's initial handshake takes more time than the original version of TLS.

## 3 mdTLS: middlebox-delegated TLS protocol with proxy signature scheme

In this section, we described the mdTLS protocol. At first, we defined the adversary model and security goals related to the mdTLS. After that, we described each phase in the protocol in detail.

### 3.1 Adversary model

We considered the attacker's capability under the Dolev-Yao model [11]. Attackers can obtain and analyze messages in the network. Furthermore, they can get public keys. They aim to obtain certificates, perform an impersonation attack via forged certificates, and reveal private keys.

### 3.2 Security goal

TLS currently provides the following properties in multi-party cases. Among them, we define "secure" for mdTLS by extending three security properties to cover the "delegation" concept.

Authentication: The notion of authentication was defined as that every entity must be able to verify whether they are talking to the "right person". This goal was divided into two sub-goals. First, each entity(client or server) can verify whether the other endpoint is operated by the expected middleboxes. It is called entity authentication. Second, If a session between two endpoints consists of an ordered set of middleboxes $M B_{1} \ldots M B_{n-1}$, then any data received by $M B_{j}$ must be a prefix of the data sent by $M B_{j-1}$ or $M B_{j+1}$, where $1<j<n-1$. It is called data authentication. We refined entity authentication into two security goals. First, the client ensures the delegated middleboxes by verifying the warrant in signature. It is called verifiability. Second, each middlebox can be identified as an appropriately delegated middlebox by checking its public key from the proxy signature. It is called strong-identifiability.

Secrecy: The notion of secrecy can be defined as that adversaries should learn nothing more from observing ciphertext in network connections. This goal is divided into two sub-goals. First, each mdTLS segment sent from entities should be encrypted with a strong ciphersuite. It is called segment secrecy. Second, each segment should have its own security parameters, such as a unique session key, to prevent the data from being reused. It is called individual secrecy.

Integrity: The notion of integrity means that only authorized or delegated entities can make or modify messages under their permissions. This goal is divided into two sub-goals. First, the entity can confirm which middleboxes have made each modification to the message. It is called modification accountability. Second, endpoints can determine the list and order of middleboxes that messages pass through. It is called path integrity. In mdTLS, we defined one security goal additionally. Delegated middleboxes can generate valid signatures. It means, in converse, undelegated entities cannot modify messages because they cannot generate and verify the signatures. Hence, it is called strong-unforgeability.

### 3.3 Overview of mdTLS protocol

The mdTLS applies a proxy signature scheme based on the partial delegation with warrant $[6,22,25]$ to improve performance while having the same security level as maTLS.

Proxy signature scheme [25] is a technique in which a proxy signer electronically signs on behalf of the original signer. When the original signer is temporarily absent, a proxy signer receives signature authority from the original signer and performs the proxy signing. This signing authority delegation technique can be used in various distributed systems, such as edge computing. There are four types of delegation in the proxy signature scheme: full delegation, partial delegation, delegation by warrant, and partial delegation with warrant [22,25].

- Full delegation: The proxy signer uses the original signer's private key to generate the proxy signature.
- Partial delegation: This method generates a proxy signing key using the private keys of both the original and the proxy signers. The advantage is that it can prevent the original signer from arbitrarily proxy signing, but there is no way to revoke or limit proxy signing authority.
- Delegation by warrant: This method uses a warrant that specifies the proxy delegation period and message space to limit proxy signing authority. It can compensate for the shortcomings of partial delegation, but performance in verification deteriorates because the verifier must additionally verify the warrant when verifying the proxy signature.
- Partial delegation with warrant: Kim et al. [22] first introduced this type of delegation. This method utilizes the advantages of both partial delegation and delegation by warrant. Proxy signing authority can be restricted or revoked through a warrant. Additionally, since this method only verifies the proxy signature, the verification efficiency can be improved.

The details of the mdTLS are shown in Figure 1, 2. For reader's convenience, notation definitions are listed in Table 1. mdTLS is divided into 3 phases.

- Generating certificates phase: Before negotiation, server certificates are generated.
- Handshake phase: Negotiation between two endpoints on a network - such as a client and a server - to establish the details of their connection. During handshake, ECDH and ECDSA $[21,29]$ are used in key exchange and digital signature, respectively.
- Record phase: Data communications are encrypted between the two entities.

The following statements below Table 1 are detailed sequences in which each entity establishes a secure communication channel based on the mdTLS.

Table 1: Notations in mdTLS

|  | Notation | Meaning |
| :---: | :---: | :---: |
| Entities | $\begin{gathered} C \\ S \\ M B_{i} \\ e_{i} \\ \hline \end{gathered}$ | Client <br> Server <br> i-th middlebox $(0<i<n)$ <br> i-th entity ( $e_{0}$ : client, $e_{n}$ : server) |
| ECDH | $\left(d_{e_{i}}^{e x}, Q_{e_{i}}^{e x}\right)$ | $e_{i}$ 's ECDH key pairs |
| ECDSA | $p$ $E$ $q$ $G$ $d_{e_{i}}$ $Q_{e_{i}}$ $H$ $S^{H}\left(d_{e_{i}}, m\right)$ $V^{H}\left(Q_{e_{i}}, m, \sigma\right)$ | A prime number <br> An elliptic curve on $\mathbb{F}_{p}$ <br> A field size (prime number) <br> A base point on $E$ having prime order $q$ <br> A private key with $0<d e_{i}<q$ <br> A public key with $d_{e_{i}} \cdot G$ on $E$ <br> Cryptographic hash function $\left(\{0,1\}^{*} \rightarrow \mathbb{F}_{q}\right)$ <br> Sign message $m$ with private key $d_{e_{i}}$ using $H$ <br> Verify signature $\sigma$ generated by $S^{H}\left(d_{e_{i}}, m\right)$ |
| Proxysignature | $\begin{gathered} P S(s k p, m) \\ P V\left(Q_{e_{i}}, m, \sigma_{p}\right) \end{gathered}$ | Proxy signing the message $m$ with proxy signing key skp <br> Proxy verification for proxy signature $\sigma_{p}$, with $Q_{e_{i}}$ |

## Phase 0. Generating certificates

1. Server sends Certificate Signing Request (CSR) to Certificate Authority (CA).
2. CA verifies CSR, creates pre-certificates, and submits to the Certificate Transparency (CT) log server to get SCTs [2].
3. After the CT log server adds pre-certificates to the logs, it returns SCTs to CA. Due to the Certificate Transparency policy [2,23], at least 2 SCTs from different CT log servers are required for certificates.
4. Using the X. 509 v3 [7] extension, CA attaches SCTs to the certificate and issues the certificate to the server.


Fig. 1: Handshake phase of mdTLS

## Phase 1. Handshake

1. Client generates ECDH key pair, and the public key $Q_{C}^{e x}$ will be sent by ClientHello message.
2. Middleboxes attach their two types of keys to the ClientHello message. One is ECDH public key, $Q_{M B_{i}}^{e x}$, and the other is ECDSA public key, $Q_{M B_{i}}$, which will be used in the proxy signature scheme.
3. Server, the original signer, also creates its ECDH and ECDSA key pairs as middleboxes. When the server receives a ClientHello message, it operates the designation process to delegate middleboxes as proxy signers. Outputs of this process are called signed delegations $\sigma_{d_{-M B_{i}}}$. For delegation, the server has to sign the hash value of the delegation message. This message consists of $Q_{S}$, the identity of proxy signer $I D_{M B_{i}}, Q_{M B_{i}}$, and a warrant $\omega$ containing the message space and delegation period. In addition, 0 is prepended to represent that it is for the proxy signature scheme. $\sigma_{d_{-M}}$ can be represented as $\left(x_{Y_{d}}, s_{d}\right)$ according to ECDSA form. Signed delegations will be sent by ServerHello message with $Q_{S}^{e x}$.
$-\sigma_{d-M B_{i}} \leftarrow S^{H}\left(d_{S}, 0\left\|Q_{S}\right\| I D_{M B_{i}}\left\|Q_{M B_{i}}\right\| \omega\right)$

- random value $y_{d}\left(0<y_{d}<q\right)$
- $Y_{d} \leftarrow y_{d} \cdot G$
- $x_{Y_{d}} \leftarrow \mathrm{x}$-coordinate of $Y_{d}$
- $c \leftarrow H\left(m_{d}\right)\left(m_{d}=0\left\|Q_{S}\right\| I D_{M B_{i}}\left\|Q_{M B_{i}}\right\| \omega\right)$
- $s_{d} \leftarrow\left(c+d_{S} \cdot x_{Y_{d}}\right) \cdot y_{d}^{-1} \bmod q$
- $\therefore \sigma_{d-M B_{i}}=\left(x_{Y_{d}}, s_{d}\right)=$ signed delegation

4. Middleboxes attach their own ECDH public key $Q_{M B_{i}}^{e x}$ to the ServerHello message. Then, middleboxes check whether signed delegations from the server are valid. If validation succeeds, middleboxes generate their proxy signing key $s k p_{M B_{i}}$.

$$
\begin{aligned}
-s k p_{M B_{i}} & \leftarrow\left(Q_{S}\left\|I D_{M B_{i}}\right\| Q_{M B_{i}} \| \omega, x_{Y_{d}}, t\right) \\
\bullet & c \leftarrow H\left(m_{d}\right)\left(m_{d}=0\left\|Q_{S}\right\| I D_{M B_{i}}\left\|Q_{M B_{i}}\right\| \omega\right) \\
\bullet & r \leftarrow H\left(Q_{S}\left\|I D_{M B_{i}}\right\| Q_{M B_{i}}\|\omega\| c\right) \\
\bullet & t \leftarrow r+d_{M B_{i}} \cdot H\left(Y_{d} \| \omega\right) \bmod q \\
& * Y_{d} \leftarrow y_{d} \cdot G=s_{d}^{-1} \cdot\left(c+d_{S} \cdot x_{Y_{d}}\right) \cdot G
\end{aligned}
$$

5. Due to the ServerCertificate message, the server sends its certificate Cert $_{S}$ to the client and middleboxes. Middleboxes generate their own certificates $C^{C r} t_{M B_{i}}$ by proxy signing the received server's certificate. Then, their certificates are sent to the client by appending to the ServerCertificate message.

- PS (skp MBi, $\left.\operatorname{Cert}_{S}\right)$ returns $\operatorname{Cert}_{M B_{i}}$, which can be shown as below:
- $\left(I D_{M B_{i}}, Q_{M B_{i}}, \omega,\left(x_{Y_{d}}, s_{d}\right), S^{H}\left(t, 0\left\|\operatorname{Cert}_{S}\right\| Q_{S}\left\|I D_{M B_{i}}\right\| Q_{M B_{i}}\|\omega\| x_{Y_{d}}\left\|s_{d}\right\| r\right)\right)$

$$
*\left(x_{Y_{p}}, s_{p}\right) \leftarrow S^{H}\left(t, 0 \| \text { Cert }_{S}\left\|Q_{S}\right\| I D_{M B_{i}}\left\|Q_{M B_{i}}\right\| \omega\left\|x_{Y_{d}}\right\| s_{d} \| r\right)
$$

6. The client, a verifier, verifies certificates to authenticate entities in TLS session. Unlike $C^{e r t} t_{S}$, the client has to use proxy verification, $P V$, to verify $\operatorname{Cert}_{M B_{i}}$, which requires the client to generate proxy public keys $P K P_{M B_{i}}$ corresponding to each middleboxes. With $P K P_{M B_{i}}$, the client verifies $\operatorname{Cert}_{M B_{i}}$.

- PV $\left(Q_{S}\right.$, Cert $_{S}$, Cert $\left._{M B_{i}}\right)$
- $\operatorname{Cert}_{M B_{i}} \leftarrow\left(I D_{M B_{i}}, Q_{M B_{i}}, \omega,\left(x_{Y_{d}}, s_{d}\right),\left(x_{Y_{p}}, s_{p}\right)\right)$
- If Certs $_{S} \notin \omega$ then return false;
- Else $P K P_{M B_{i}} \leftarrow r \cdot G+H\left(s_{d}^{-1} \cdot\left(c \cdot G+x_{Y_{d}} \cdot Q_{S}\right) \| \omega\right) \cdot Q_{M B_{i}}$; $* c \leftarrow H\left(0\left\|Q_{S}\right\| I D_{M B_{i}}\left\|Q_{M B_{i}}\right\| \omega\right), r \leftarrow H\left(Q_{S}\left\|I D_{M B_{i}}\right\| Q_{M B_{i}}\|\omega\| c\right)$
- $V^{H}\left(\right.$ PK $_{M B_{i}}, 0 \|$ Cert $\left._{S}\left\|Q_{S}\right\| I D_{M B_{i}}\left\|Q_{M B_{i}}\right\| \omega\left\|x_{Y_{d}}\right\| s_{d} \| r,\left(x_{Y_{p}}, s_{p}\right)\right)$

7. Server sends ServerFinished message with security parameter block (SPB). These blocks consist of signatures of $H M A C$. This $H M A C$ generates authentication code from security parameters such as ciphersuite and handshake messages. For middleboxes, they have to proxy sign their blocks with their generated $s k p_{M B_{i}}$. For a client, it must verify middleboxes' signed blocks with its generated proxy public keys $P K P_{M B_{i}}$.

## Phase 2. Record

- Modification log is attached to the message and helps to check whether a message is modified. Besides, endpoints can also check whether unauthorized entities modify messages without permission.


Fig. 2: Record phase of mdTLS

## 4 Performance analysis for mdTLS

In this section, we analyzed the performance of the mdTLS by conducting a comparative analysis with maTLS, which we consider to be among the best of the existing TLS-extension protocols. Our performance analysis is focused on the number of computations in protocols. Both mdTLS and maTLS rely on ECDSA for the generation of security parameters. ECDSA, being based on the Elliptic Curve Discrete Logarithm Problem (ECDLP), involves a substantial number of point multiplication operations. These operations can significantly influence the performance of both protocols. Therefore, we conducted a performance analysis employing algorithms capable of measuring the number of point multiplication operations. It is important to note that this analysis is based on server-only authenticated TLS version 1.2 and assumes that 3 SCTs are created for each certificate through the Certificate Transparency policy [1-3,23].

### 4.1 Preliminaries for performance analysis

To facilitate performance comparisons between two protocols that offer the same 128 -bit security strength, we have set the elements within the protocols, as shown below [12].

- Types of elliptic curve: Secp256r1
- Private key size: 256 bits
- Hash size: 256 bits


### 4.2 Analyzing the performance between maTLS and mdTLS

To measure the number of point multiplication operations, we employed the double-and-add algorithm, which averages 1 point doubling and 0.5 point additions per bit. Therefore, we considered an average of 1.5 point multiplication operations per bit. Following this, we divided the protocol into two segments and measured the number of point multiplication operations. The first segment corresponds to the generation and verification of certificates for utilization in the handshake phase. The number of computations for each protocol in this segment is detailed in Table 3 and 4 below. The second segment is where entities (server, client, middlebox) create and verify security parameters to be exchanged at the handshake phase. The number of computations for each protocol in this segment is detailed in Table 2 below.

Table 2: Computational analysis for security parameter blocks

| Descriptions | maTLS | mdTLS |
| :--- | :---: | :---: |
| Server generates security parameter blocks. | 384 | 384 |
| Middlebox generates security parameter blocks. | 384 N | 384 N |
| Client verifies blocks from the server. | 768 | 768 |
| Client verifies blocks from the middleboxes. | 768 N | 768 N |

Table 3: Computational analysis for generating certificates

| Descriptions | maTLS | mdTLS |
| :--- | :---: | :---: |
| - Server side |  |  |
| Server generates keys and signature for CSR to CA. | 768 | 768 |
| CA verifies CSR signature. | 768 | 768 |
| CT log servers generate keys and signatures for 3 SCTs. | 2,304 | 2,304 |
| CA generates keys and signs for server's certificate. | 768 | 768 |
| - Middlebox side for maTLS | 768 N | - |
| Middleboxes generate keys and signature for CSR to CA. | 768 N | - |
| CA verifies CSR signature. | $2,304 \mathrm{~N}$ | - |
| MT log servers generate keys and signatures for 3 SCTs. | 768 N | - |
| CA generates keys and signs for middleboxes' certificate. |  |  |
| - Middlebox side for mdTLS | - | 384 N |
| Each middlebox generates its keys. | - | 384 N |
| Server generates signed delegations to assign proxy signers. | - | 768 N |
| Middlebox verifies signed delegation and generate proxy signing key. | - | 384 N |
| Middleboxes generate certificates with proxy signing key. |  |  |

Table 4: Computational analysis for certificates verification

| Descriptions | maTLS | mdTLS |
| :--- | :---: | :---: |
| Client verifies the signature and 3 SCTs in the server's certificate. | 3,072 | 3,072 |
| Client verifies the middleboxes' certificates. | $3,072 \mathrm{~N}$ | $2,304 \mathrm{~N}$ |



Fig. 3: Performance of protocols when using ECDSA

We have implemented certain components essential for the functionality of mdTLS. We mainly implemented internal functions for computing data required during the handshake phase, such as key or signature generation and verification. We implemented and analyzed its performance within a virtual environment, specifically using docker container. The rest of our testbed in docker image is as follows:

- Ubuntu 22.04.3 LTS
- Intel(R) Core(TM) i5-10400 CPU @ 2.90GHz
- 2GiB RAM

Table 5: Average execution time in implementation

| Features | maTLS | mdTLS |
| :---: | :---: | :---: |
| ECDSA signing | 1.4 ms | 1.4 ms |
| ECDSA verification | 2.5 ms | 2.5 ms |
| Proxy signing | - | 1.6 ms |
| Proxy verification | - | 8.9 ms |

Table 5 shows the time spent when signing and verifying the CSR files. Since the proxy signature scheme requires additional keys, the execution time of mdTLS is longer than maTLS. However, by reusing these keys when processing the security parameter block, the execution time of mdTLS can become similar to maTLS.

## 5 Security analysis for mdTLS

In this section, we conducted a security analysis of mdTLS using an approach similar to the one employed for maTLS [24], involving formal specification and verification through the Tamarin [40]. Tamarin is an automated formal verification tool based on multiset rewriting rules in the theory of equations. It has been continuously updated to maintain its effectiveness. Using this tool, maTLS formally verified six security lemmas: server authentication, middlebox authentication, data authentication, path integrity, path secrecy, and modification accountability. In the case of mdTLS, we successfully verified not only the same lemmas as previously done in maTLS but also three novel lemmas related to the proxy signature scheme following the same approach and tools: verifiability, strong-unforgeability, and strong-identifiability. However, in this paper, we only described three novel lemmas related to the proxy signature scheme, taking into consideration the maximum page limit imposed by the conference guidelines. The rest can be found on our GitHub [4].

### 5.1 Experimental setup

To analyze the security of the mdTLS, we established an experimental environment, as illustrated below. Our goal was to confirm that the formal model of mdTLS aligns with the security lemmas within our testing environment.

- Amazon Elastic Compute Cloud (Amazon EC2) c5a.24xlarge instance
- 96 vCPUs, 192 GiB RAM
- Ubuntu 22.04.2 LTS


### 5.2 Formal specification

We have formalized the mdTLS, specifying the detailed operations conducted by each entity during the handshake and record phases in the form of rules. For cryptographic primitives like hash, signature, and PRF (Pseudo-Random Function) [15], we used the built-in functions provided by Tamarin. Details of all rules can be found in the spthy file uploaded to our github [4]. The script below illustrates an example of the detailed operations concerning ServerHello messages. In the handshake phase, when the server receives a ClientHello message from the client, it responds by sending a ServerHello message to initiate mutual authentication. In this process, mdTLS sends a ServerKeyExchange message, a signed delegation, a Diffie-Hellman public key, and a ServerCertificate message. The delegation in this context consists of the server's public key, the middlebox's public key and identification information, and a warrant providing an explanation of the delegation.

```
rule Server_Hello:
    let
        server_hello_msg
            = < 'server_hello', ~ns, server_chosen_details >
        server_key_exchange = s_dhe_pub
        server_key_exchange_signed
            = < server_key_exchange, sign(h(server_key_exchange)
                , ltk) >
        server_cert = < $S, pk(ltk) >
        warrant = ~warrant_fresh
        proxy_delegation = < pk(ltk), $M, mb_pubkey, warrant >
        proxy_delegation_signed = sign(h(proxy_delegation), ltk)
        Y_d = calcY_d(~ y, 'G_skp')
        y_d_x = pointx(Y_d)
        c = h(proxy_delegation)
        s_d = multp( plus(multp(ltk, y_d_x), c), inv(~y) )
        proxy_delegation_signed_pair
        = < proxy_delegation, proxy_delegation_signed, <y_d_x, s_d> >
    in
    [ In( <mb_client_hello_msg, c_mb_extension> )
        , !PrivateKey('server', $S, ltk) ]
    --[
        ServerSendDelegation(ltk_pub, mb_pubkey, warrant, proxy_delegation)
    ]->
    [ Out( <server_hello_msg, server_key_exchange_signed
        , proxy_delegation_signed_pair, s_extension
            , server_cert> ) ... ]
```


### 5.3 Formal verification

A Tamarin-based formal model is a set of multiple rules, and these single rules are made up of three basic components. Facts represent detailed information about the current execution in the model. States are multisets of facts. During formal verification, user-defined functions called rules can add or remove facts from the state. This is often denoted as $\mathrm{l} \rightarrow[\mathrm{a}] \rightarrow \mathrm{r}$, indicating that fact " l " is removed from the state and replaced by fact "r," with this process traced through the action denoted as "a." Tamarin, following these principles, can verify whether a lemma, which is desired to be satisfied throughout the protocol, holds even as the state changes in the operation. Tamarin's verification process is based on tracing the protocol's state through actions. To evaluate the security of our protocol, we defined nine security lemmas and one source lemma. Among them, security lemmas consist of six security lemmas of maTLS and three security lemmas related to the proxy signature scheme. As previously noted, we described three security lemmas associated with proxy signatures. Prior to describing them, we described an additional description of a source lemma designed to assist Tamarin in accurately verifying the formal specifications.

Source lemma. A source lemma is a concept used for formally verifying the security lemmas that a security protocol must adhere to during its execution. When conducting formal verification of an overall protocol, Tamarin adopts a strategy of deconstructing the protocol into smaller, more manageable components for analysis. The verification outcomes for these individual subsets are then used as supporting evidence to confirm that the entire protocol operates correctly and meets its prescribed security lemmas. However, during the verification process of these subsets, if Tamarin encounters difficulties in distinguishing between variables as nonce values or ciphertexts, it may face challenges in completing the verification. This is commonly referred to as a "partial deconstruction". To address such issues, it becomes necessary to establish a source lemma that precisely specifies the origin of these variables. From this source lemma, a refined source is generated, comprising a new set of sources. All security lemmas are subsequently verified using these refined sources, underscoring the importance of validating the source lemma to ensure the accurate computation of these refined sources $[8,40]$. When we initially omitted the definition of source lemmas, the formally specified mdTLS model yielded 120 partial deconstructions. Consequently, we defined source lemmas to enable Tamarin to discern the origins of these problematic variables. Upon closer analysis, it was determined that the issue of partial deconstruction occurred in 4 distinct segments, one of which pertained to the scenario where a middlebox received an encrypted request message sent by the client. To resolve this particular issue, we formulated a source lemma indicating that the encrypted message enc received by the middlebox had been transmitted from the client through the OutClientRequest () action, as shown below. By employing this approach, we could generate refined sources in a state of "deconstructions completed". This strategic use of source lemmas proved in-
strumental in addressing the partial deconstruction challenge and facilitating the successful verification process within the mdTLS model.

```
All enc msg #i.
    InMbClientRequest( enc, msg ) @ i
    ==> (Ex #j. KU(msg) @ j & j < i)
        | (Ex #j. OutClientRequest( enc ) @ j & j<i)
```

Security lemma. After resolving the partial deconstruction issue, we verified that our protocol meets the nine security lemmas outlined in Section 3.2. In this section, we define three of the nine security goals related to proxy signature scheme. We also defined detailed information about the formulas that convert informal definitions into mathematical formulas called lemmas.

- Verifiability: The client must verify whether the middlebox's certificate, the proxy signature, was created with the consent of the server. To verify this lemma, we have to check whether the middlebox generated its certificate based on delegation and warrant sent by the server through the ServerHello message, as specified in rule Server_Hello.

```
All warrant mbLtk mbCert #tc.
    ClientReceivedProxySign(warrant, pk(mbLtk), mbCert) @tc
    ==> Ex delegation gy #tmb.
        MbGenerateProxySign(delegation, mbLtk, gy, warrant, mbCert)
        @tmb & KU(gy) @tmb & not(Ex #tmb. KU(mbLtk) @tmb)
        ==> Ex sPub #ts
            ServerSendDelegation(sPub, pk(mbLtk), warrant, delegation)
            @ts & (#ts < #tmb) & KU(sPub) @ts
```

- Strong-unforgeability: The proxy signer's private key, which is used to generate the proxy signature, must not be revealed. Otherwise, the proxy signature can be forged by an adversary.

```
All warrant mbLtk mbCert #tc.
    ClientReceivedProxySign(warrant, pk(mbLtk), mbCert) @tc
    ==> All delegation gy sPub #tmb.
        (MbGenerateProxySign(delegation, mbLtk, gy, warrant, mbCert) @tmb
        & KU(gy)@tmb & not(Ex #tmb.KU(mbLtk) @tmb))
        & (MbReceiveProxyDelegation(sPub, pk(mbLtk), delegation) @tmb)
        ==> All #ts.
            ServerSendDelegation(sPub, pk(mbLtk), warrant, delegation)@ts & KU(sPub)@ts
            ==> Ex #tmbclient. MbSendPublicKey(pk(mbLtk)) @tmbclient
                & KU(pk(mbLtk)) @tmbclient
```

- Strong-identifiability: The identification of a proxy signer can be proved by its public key. The public key of the middlebox included in the proxy signature sent to the client must be the same as the public key of the middlebox sent to the server for proxy delegation.


## 16 Ahn et al.

```
All warrant mbPub mbCert #tc.
    ClientReceivedProxySign(warrant, mbPub, mbCert)@tc
    ==> All delegation mbLtk gy sPub #tmb.
        (MbGenerateProxySign(delegation, mbLtk, gy, warrant, mbCert)
        @tmb & KU(gy)@tmb & not(Ex #tmb. KU(mbLtk) @tmb))
        & (MbReceiveProxyDelegation(sPub, pk(mbLtk), delegation) @tmb)
        ==> All #ts
            ServerSendDelegation(sPub, pk(mbLtk), warrant, delegation)
            @ts & KU(sPub)@ts
            ==> Ex #tmbclient. MbSendPublicKey(pk(mbLtk)) @tmbclient
                & KU(pk(mbLtk)) @tmbclient & (mbPub = pk(mbLtk))
```

Results of verification The overall result of formal verification is shown in Figure 4. Figure 4 illustrates that our mdTLS protocol not only satisfies the three security lemmas introduced above but also aligns with the lemmas validated for maTLS. Furthermore, Figure 5 shows mathematical proofs (verification process) demonstrating the consistent validity of the verifiability lemma within our mdTLS protocol among the security lemmas outlined in Figure 4.

```
/* All well-formedness checks were successful. */
end
```

summary of summaries:
analyzed: mdTLS_ecdsa.spthy

```
    source lemma (all-traces): verified (5660 steps)
    server_authentication (all-traces): verified (10 steps)
    middlebox authentication (all-traces): verified (12 steps)
    middlebox_path_integrity (all-traces): verified (8 steps)
    path_secrecy (all-traces): verified (2 steps)
    modification_accountability (all-traces): verified (6 steps)
    data_authentication (all-traces): verified (2 steps)
    proxy verifiability (all-traces): verified (10 steps)
    proxy_strong_unforgeability (all-traces): verified (12 steps)
    proxy_strong_identifiability (all-traces): verified (12 steps)
```

Fig. 4: Overview of formal verification results

As mentioned earlier, Tamarin formally verifies whether the rules always satisfy the lemma, called validity. A typical approach to verifying validity is negating the formulas and checking for inconsistencies. Figure 5 shows the negated lemma for verifiability, followed by verifying whether this formulation leads to contradictions. Following this process, we have validated all six security lemmas mentioned earlier.

```
lemma proxy_verifiability:
    all-traces
    "(\forall warrant mbLtk mbCert #tc.
        (ClientReceivedProxySign( warrant, pk(mbLtk), mbCert ) @ #tc) =
        (\forall delegation ydx sd #tmb.
        (()(MbGenerateProxySign( delegation, mbLtk, ydx, sd, warrant,
                        mbCert
                    ) @ #tmb) ^
            (!KU( ydx ) @ #tmb)) ^
            (!KU( sd ) @ #tmb)) ^
                (\neg(\exists #tmb.1. ! KU( mbLtk) @ #tmb.l))) =
                (\exists sPub #ts.
                    ((ServerSendDelegation( sPub, pk(mbLtk), warrant, delegation
                    ) @ #ts) ^
                    (#ts < #tmb)) ^
            (!KU( sPub ) @ #ts)))) ^
        (\forall warrant mbLtk mbSign #tc.
        (ClientReceivedProxySignForSpb( warrant, pk(mbLtk), mbSign
        ) @ #tc) =>
        (}\forall\mathrm{ delegation ydx sd #tmb.
            (()(MbGenerateProxySignForSpb( delegation, mbLtk, ydx, sd, warrant,
                        mbSign
                    )@ #tmb)^
            (!KU(ydx ) @ #tmb)) ^
            (!KU( sd ) @ #tmb)) ^
        (\neg(\exists #tmb.1. ! KU(mbLtk) @ #tmb.1))) =
        (\exists sPub #ts.
            ((ServerSendDelegation( sPub, pk(mbLtk), warrant, delegation
            ) @ #ts) ^
            (#ts < #tmb)) ^
            (!KU( sPub) @ #ts))))"
```

Fig. 5: Proof of verifiability lemmas in Tamarin

## 6 Discussion

We proposed an ECDSA-based cryptographic protocol. However, during the research, we found new insights for improvement. The insight is to use the Schnorr algorithm instead of ECDSA for the algorithm that generates the digital signature. Boldyreva et al.'s research [6] used Schnorr signature, and they shows better outcomes in terms of both performance and security than ECDSA.

- Performance: Schnorr does not have modular inverse calculations that significantly affect performance.
- Security: Since Schnorr is strongly unforgeable under chosen message attack (SUF-CMA), Schnorr is provably secure in the random oracle model [35].

So we compared the performance of the maTLS and mdTLS protocols assumed that both protocols use the Schnorr signature. To measure the performance of Schnorr, the number of modular multiplication operations was calculated using the square-and-multiply algorithm. This algorithm requires 1.5 modular multiplications per bit on average. Besides, as mentioned in Schnorr's paper [38], we calculated the modular multiplications of the Schnorr verification equation by multiplying by 1.75 per bit. When the security level is set to 128 -bit, the related parameters' sizes can be shown below [12].

- Public key size: 3,072 bits
- Private key size: 256 bits
- Hash size: 256 bits

Ahn et al.

Table 6 shows the number of modular multiplications at each stage. Here, N represents the number of middleboxes. The mdTLS reduces the number of modular multiplications by $51.8 \%$ compared to maTLS, demonstrating better performance when using Schnorr than when using ECDSA. Nevertheless, the TLS standard mandates the utilization of the ECDSA algorithm for digital signature creation, rendering the adoption of the Schnorr signature algorithm impractical now.

Table 6: Modular multiplications in maTLS and mdTLS

| Stages | maTLS | mdTLS |
| :---: | :---: | :---: |
| Certificate generation | $4,293 \mathrm{~N}+4,293$ | $1,603 \mathrm{~N}+4,293$ |
| Certificate verification | $1,792 \mathrm{~N}+1,792$ | $897 \mathrm{~N}+1,792$ |
| Security parameter blocks | $833 \mathrm{~N}+833$ | $833 \mathrm{~N}+833$ |
| Overall | $6,918 \mathrm{~N}+6,918$ | $3,333 \mathrm{~N}+6,918$ |



Fig. 6: Performance of protocols when using Schnorr

## 7 Conclusion

In this paper, we proposed a middlebox-delegated TLS protocol in which only middleboxes that have been permitted can participate in the network. To demonstrate the excellence of our proposed protocol, we verified our protocol from two aspects of view: performance and security. In the performance view, we calculated the number of computations in the protocol. We found that the mdTLS reduces about $39 \%$ of the computations compared to maTLS. Also, we formally verified that our proposal achieved nine security lemmas: server/middlebox/data authentication, path integrity, path secrecy, modification accountability, verifiability, strong-unforgeability, and strong-identifiability. Especially among them, the latter three security lemmas are newly defined for our protocol by extending existing concepts. The primary contribution of this work is to show that using the proxy signature scheme can enhance performance efficiency and maintain its security level.

Acknowledgements This work was partly supported by Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2018-0-00532, Development of High-Assurance (EAL6) Secure Microkernel, 100) and supported by Korea University.

## References

1. Apple's Certificate Transparency policy Homepage, https://support.apple.com/enng/HT205280, Last accessed 21 May 2023
2. Certificate Transparency Homepage, https://certificate.transparency.dev, Last accessed 21 May 2023
3. Chrome Certificate Transparency Policy Homepage, https://googlechrome.github.io/CertificateTransparency/ct_policy.html, Last accessed 21 May 2023
4. Hackproof github Homepage, https://github.com/HackProof/mdTLS, Last accessed 26 May 2023
5. Anderson, B.: Detecting Encrypted Malware Traffic (Without Decryption), https://blogs.cisco.com/security/detecting-encrypted-malware-traffic-withoutdecryption, Last accessed 26 September 2023
6. Boldyreva, A., Palacio, A., Warinschi, B.: Secure Proxy Signature Schemes for Delegation of Signing Rights. Journal of Cryptology 25, 57-115 (2012)
7. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Internet X. 509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) profile (2008), https://www.rfc-editor.org/rfc/rfc5280.txt, Last accessed 23 September 2023
8. Cortier, V., Delaune, S., Dreier, J.: Automatic generation of sources lemmas in TAMARIN: towards automatic proofs of security protocols. In: Computer Security-ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part II. pp. 3-22. Springer (2020)
9. Dang, Q.H.: Secure Hash Standard (2015), https://nvlpubs.nist.gov/nistpubs/FIPS /NIST.FIPS.180-4.pdf, Last accessed 23 September 2023
10. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2 (2008), https://www.rfc-editor.org/rfc/rfc5246.txt, Last accessed 23 September 2023
11. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Transactions on information theory 29(2), 198-208 (1983)
12. Elaine, B.: Recommendation for Key Management: Part 1 - General (2020), https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf, Last accessed 23 September 2023
13. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext Transfer Protocol-HTTP/1.1 (1999), https://www.rfceditor.org/rfc/rfc2616.txt, Last accessed 23 September 2023
14. Gallagher, S.: Nearly half of malware now use TLS to conceal communications, https://news.sophos.com/en-us/2021/04/21/nearly-half-of-malware-now-use-tls-to-conceal-communications, Last accessed 26 September 2023
15. Goldreich, O., Goldwasser, S., Micali, S.: How to Construct Random Functions. Journal of the ACM (JACM) 33(4), 792-807 (1986)
16. Goltzsche, D., Rüsch, S., Nieke, M., Vaucher, S., Weichbrodt, N., Schiavoni, V., Aublin, P.L., Cosa, P., Fetzer, C., Felber, P., et al.: EndBox: Scalable Middlebox Functions Using Client-Side Trusted Execution. In: 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). pp. 386397. IEEE (2018)
17. Google: Google Transparency Homepage, https://transparencyreport.google.com/ https/overview?hl=en, Last accessed 9 May 2023
18. Han, J., Kim, S., Ha, J., Han, D.: Sgx-Box: Enabling Visibility on Encrypted Traffic using a Secure Middlebox Module. In: Proceedings of the First Asia-Pacific Workshop on Networking. pp. 99-105 (2017)
19. Heron, S.: Advanced Encryption Standard (AES). Network Security 2009(12), 812 (2009)
20. Jarmoc, J., Unit, D.: SSL/TLS Interception Proxies and Transitive Trust. Black Hat Europe (2012)
21. Johnson, D., Menezes, A., Vanstone, S.: The Elliptic Curve Digital Signature Algorithm (ECDSA). International journal of information security 1, 36-63 (2001)
22. Kim, S., Park, S., Won, D.: Proxy Signatures, Revisited. In: Information and Communications Security: First International Conference, ICIS'97 Beijing, China, November 11-14, 1997 Proceedings 1. pp. 223-232. Springer (1997)
23. Laurie, B., Langley, A., Kasper, E.: RFC 6962: Certificate Transparency (2013), https://www.rfc-editor.org/rfc/rfc6962.txt, Last accessed 23 September 2023
24. Lee, H., Smith, Z., Lim, J., Choi, G., Chun, S., Chung, T., Kwon, T.T.: maTLS: How to Make TLS middlebox-aware? In: NDSS (2019)
25. Mambo, M., Usuda, K., Okamoto, E.: Proxy Signatures: Delegation of the Power to Sign Messages. IEICE transactions on fundamentals of electronics, communications and computer sciences $79(9)$, 1338-1354 (1996)
26. Meier, S., Schmidt, B., Cremers, C., Basin, D.: The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In: Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings 25. pp. 696-701. Springer (2013)
27. Microsoft: Microsoft Azure firewall Homepage, https://learn.microsoft.com/ko-kr/azure/firewall/premium-features, Last accessed 9 May 2023
28. Miller, J.: Telegeography Homepage, https://blog.telegeography.com/2021-global-internet-map-tracks-global-capacity-traffic-and-cloud-infrastructure, Last accessed 9 May 2023
29. National Institute of Standards and Technology: Digital Signature Standard (DSS) (2023), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf, Last accessed 23 September 2023
30. National Security Agency: Cybersecurity and Infrastructure Security Agency Homepage, https://www.us-cert.gov/ncas/current-activity/2019/11/19/nsa-releases-cyber-advisory-managing-risk-transport-layer-security, Last accessed 9 May 2023
31. Naylor, D., Li, R., Gkantsidis, C., Karagiannis, T., Steenkiste, P.: And Then There Were More: Secure Communication for More Than Two Parties. In: Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies. pp. 88-100 (2017)
32. Naylor, D., Schomp, K., Varvello, M., Leontiadis, I., Blackburn, J., López, D.R., Papagiannaki, K., Rodriguez Rodriguez, P., Steenkiste, P.: Multi-context TLS (mcTLS): Enabling Secure In-Network Functionality in TLS. ACM SIGCOMM Computer Communication Review 45(4), 199-212 (2015)
33. O'Neill, M., Ruoti, S., Seamons, K., Zappala, D.: TLS Inspection: How Often and Who Cares? IEEE Internet Computing 21(3), 22-29 (2017)
34. Panetta, K.: Gartner Homepage, https://www.gartner.com/smarterwithgartner/ gartner-top-10-strategic-technology-trends-for-2019, Last accessed 9 May 2023
35. Pointcheval, D., Stern, J.: Security Arguments for Digital Signatures and Blind Signatures. Journal of cryptology 13, 361-396 (2000)
36. Rescorla, E.: HTTP Over TLS (2000), https://www.rfc-editor.org/rfc/rfc2818.txt, Last accessed 23 September 2023
37. Schmidt, B., Meier, S., Cremers, C., Basin, D.: Automated Analysis of DiffieHellman Protocols and Advanced Security Properties. In: 2012 IEEE 25th Computer Security Foundations Symposium. pp. 78-94. IEEE (2012)
38. Schnorr, C.P.: EFFICIENT IDENTIFICATION AND SIGANTRUES FOR SMART CARDS. In: Advances in Cryptology-CRYPTO'89 Proceedings 9. pp. 239-252. Springer (1990)
39. Sherry, J., Hasan, S., Scott, C., Krishnamurthy, A., Ratnasamy, S., Sekar, V.: Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service. ACM SIGCOMM Computer Communication Review 42(4), 13-24 (2012)
40. The-Tamarin-Team: Tamarin-Prover Manual, https://tamarin-prover.github.io/manual/master/tex/tamarin-manual.pdf, Last accessed 17 May 2023
41. Trach, B., Krohmer, A., Gregor, F., Arnautov, S., Bhatotia, P., Fetzer, C.: ShieldBox: Secure Middleboxes using Shielded Execution. In: Proceedings of the Symposium on SDN Research. pp. 1-14 (2018)
42. Wei, C., Li, J., Li, W., Yu, P., Guan, H.: STYX: A Trusted and Accelerated Hierarchical SSL Key Management and Distribution System for Cloud Based CDN Application. In: Proceedings of the 2017 Symposium on Cloud Computing. pp. 201-213 (2017)

# PHI: Pseudo-HAL Identification for Scalable Firmware Fuzzing 

Seyeon Jeong ${ }^{1,2 \star}$, Eunbi Hwang ${ }^{1, *}$, Yeongpil Cho ${ }^{3}$, and Taekyoung Kwon ${ }^{1 \star \star}$<br>${ }^{1}$ Graduate School of Information, Yonsei University, Seoul 03722, South Korea<br>${ }^{2}$ Suresofttech Inc., Seongnam-si, Gyeonggi 13453, South Korea<br>best6653@gmail.com, \{ebhwang95, taekyoung\}@yonsei.ac.kr<br>${ }^{3}$ Hanyang University, Seoul 08826, South Korea<br>ypcho@hanyang.ac.kr


#### Abstract

Firmware fuzzing aims to detect vulnerabilities in firmware by emulating peripherals at different levels: hardware, register, and function. HAL-FuZZ, which emulates peripherals through HAL function handling, is a remarkable firmware fuzzer. However, its effectiveness is confined to firmware solely relying on HAL functions, and it necessitates intricate firmware information for best outcomes, thereby limiting its target firmware range. Notably, in commercial firmware, both HAL and non-HAL (which we call "pseudo-HAL") functions are prevalent. Identifying and addressing both is crucial for comprehensive peripheral control in fuzzing. In this paper, we present PHI, a tool designed to identify HAL and pseudo-HAL functions at the register level. Using PHI, we develop PHI-Fuzz, an enhanced firmware fuzzer operating at the function level. This fuzzer efficiently manages HAL and pseudo-HAL functions, demanding minimal prior knowledge yet delivering substantial results. Our evaluation demonstrates that PHI identifies HAL functions accessing the MMIO range as effectively as LibMatch of HAL-Fuzz, while overcoming its constraints in detecting pseudo-HAL functions. Significantly, when benchmarked against HAL-Fuzz, PHI-Fuzz showcases superior bug-finding capabilities, uncovering crashes that HAL-FUZZ missed.


Keywords: Security, Firmware, Fuzzing, Hardware Abstraction Layer

## 1 Introduction

Embedded devices play a crucial role in various applications, including the Internet of Things (IoT), aviation, and weapons systems. According to State of IoT-Spring 2023 [1] report, there was an $18 \%$ growth in the number of global IoT connections during 2022, resulting in a total of 14.3 billion active IoT endpoints. However, when compared to the total vulnerabilities discovered, firmware vulnerabilities have consistently accounted for about $2 \%$ each year since 2017, and as of 2023, $2.41 \%$ of firmware vulnerabilities have been identified [2].

[^10]Firmware vulnerabilities, which can result from system crashes, reboots, and hangs, are exploitable by attackers aiming to compromise embedded devices. This poses a significant risk to society, thus necessitating dynamic analysis and proactive detection through firmware fuzzing [14, 19, 20].

Fuzzing, a dynamic bug-finding technique, provides random input values to a program and monitors its executions. AFL (American Fuzz Lop) [24] is a coverage-guided fuzzer that has demonstrated high performance in general software fuzzing and can also be utilized for firmware fuzzing on microcontroller units (MCU) [16, 20, 25]. However, exploring firmware vulnerabilities through fuzzing techniques can be challenging, particularly for embedded devices with inherent limitations. To address these challenges, recent firmware fuzzing research has proposed emulation-based fuzzing [10-13,18,26]. Firmware emulation enables fuzzing on devices with sufficient power and capacity. Nonetheless, using a general emulator like QEMU [9] can lead to execution failures due to undefined peripheral access during firmware fuzzing. Consequently, how emulators handle peripherals is crucial for successful firmware emulation and fuzzing. Emulation through Hardware-In-The-Loop (HITL) method can result in performance degradation due to communication between hardware and the emulator [19]. Recent studies have focused on peripheral modeling as a way to overcome this limitation. Peripheral modeling techniques can be classified into three types: hardware-level, function-level, and register-level modeling. Function-level and register-level modeling do not require hardware during the modeling phase, resulting in better performance for firmware emulation and fuzzing.

Function-level peripheral modeling involves emulating firmware by hooking a function during emulation and connecting pre-made handlers. Register-level peripheral modeling handles each register during emulation. Compared to registerlevel modeling, function-level modeling boasts faster processing, as peripheral functions accessing Memory-mapped I/O (MMIO) are processed with a handler. HALucinator, a firmware emulator, implements function-level peripheral modeling using Python handlers achieved through Hardware Abstraction Layer (HAL) function hooking [11]. Building upon this concept, HAL-Fuzz, a firmware fuzzer, integrates HALucinator with UnicornAFL [3]. HALucinator and HALFuzz identify functions to be hooked using LibMatch [4], a HAL function identification tool. Although LibMatch can identify HAL functions, it requires a software development kit (SDK) containing HAL function object files compiled in the same environment as the target firmware. As a result, LibMatch needs extensive information about the firmware despite its limited capabilities in identifying functions.

Many modern firmware implementations utilize not only HAL but also pseudoHAL functions. Consequently, LibMatch may not fully identify all functions in the firmware, limiting the effectiveness of HALucinator and HAL-Fuzz. Additionally, obtaining detailed information about firmware compilation options can be challenging, and the scripts used in LibMatch are often not openly available. This makes it difficult to use LibMatch in an ideal operating environment. To overcome these limitations, we propose the Pseudo-HAL Identification
(PHI) program, which leverages symbolic execution to identify HAL and pseudoHAL functions at the register level without relying on specific firmware compilation environments or firmware stripping. Furthermore, we introduce PHI-Fuzz, a function-level firmware fuzzer based on HAL-Fuzz that utilizes PHI's results. With the scalability provided by PHI, PHI-Fuzz can perform more efficient and effective fuzzing compared to existing function-level firmware fuzzers.
Contribution. This paper makes the following contributions.

- Pseudo-HAL Identification We propose PHI, a register-level function identification method for more scalable function-level peripheral modeling.
- PHI-Fuzz We propose PHI-Fuzz, an enhanced and scalable firmware fuzzer operating at the function level by leveraging PHI.
- For further research, we will release our tool at publication time.

Organization. This paper is organized as follows. Section II provides the necessary background and discusses the existing problems. Section III presents the design of the proposed system. Section IV describes the implementation of the system. Section V presents the evaluation of the system. Section VI provides a discussion of the results and limitations. Section VII reviews the related work. Finally, Section VIII concludes the paper.

## 2 Motivation

In this section, we briefly discuss the background of firmware fuzzing, identify the challenges of existing techniques, and demonstrate their limitations through a series of experiments.

### 2.1 Background

Firmware in Embedded Devices Firmware is a type of software that offers low-level control over hardware components, including on-chip and off-chip peripherals, as well as MCUs integrated into embedded systems. Muench et al. [19] classified embedded devices into three categories based on firmware: general OS-based firmware, embedded OS-based firmware, or monolithic firmware. Monolithic firmware (also known as bare-metal firmware) is present in approximately $81 \%$ of embedded devices as of 2019 [5]. This firmware type operates by executing simple functions in a continuous loop and is commonly used in smallscale embedded systems. Our study focuses on developing a firmware fuzzing technique that specifically targets monolithic firmware.

Firmware Fuzzing Traditional fuzzing techniques for general software often require instrumentation to observe and analyze the behavior of the tested program. However, firmware fuzzing presents additional challenges due to the high dependency on heterogeneous peripherals and the lack of reliable emulation techniques. Fully emulating firmware, including both the processor and peripherals,


Fig. 1. STM32 firmware architecture
can be a complex and time-consuming process owing to the wide variety of peripherals available. For firmware testing, partial emulation using the hardware-in-the-loop (HITL) method may be slower than the peripheral modeling method, as it may cause a bottleneck in the communication process between the emulator and the actual hardware being emulated [19, 22]. Recently, emulation techniques utilizing peripheral modeling have gained popularity for effective firmware fuzzing [10-13, 18, 21, 26].

HAL(Hardware Abstraction Layer) HAL is a library provided by manufacturers to enhance the convenience of firmware development. By abstracting common functionality for specific devices, HAL makes developers program without relying on a specific hardware target [6]. Since many manufacturers produce various types of hardware, developing firmware based on specific hardware requires a significant loss of productivity to develop firmware that directly accesses the hardware. Using HAL has the advantage of facilitating the development of essential functions when creating firmware. It is presented as higher-layer functions rather than register units, enabling convenient usage through function calls without the need for direct register access. For instance, in implementing the functionality to send data over UART, developers can simply call the HAL_UART_Transmit() function without directly manipulating the Data Register. HALucinator [11] leveraged the characteristics of this HAL in firmware emulation. Identified HAL function calls and handled them with pre-made handlers, HALucinator improved emulation efficiency. Unlike HALucinator, which identified HAL functions at the function level, the PHI proposed in this paper detects not only HAL functions but also various library functions for peripherals HAL functions at the register level.

### 2.2 Problem Definition

A central question this study aimed to address is whether function-level fuzzing, as a peripheral modeling method, is more efficient than register-level fuzzing. We also examined the scalability of current function-level emulation techniques. To answer these questions, we conducted several experiments as part of our research.

```
Example 1 Firmware execution code
    int main()\{
    char a[5];
    char \(\mathrm{b}=\) HAL_uart_getc();
    \(\mathrm{a}[\mathrm{b}]=1\);
\}
Example 2 Firmware execution code
    int main()\{
    char a[5];
    data = HAL_UART_Receive_IT(huart, pData, Size);
    strcpy(a, data);
\}
```

Efficiency of function-level emulation for fuzzing This paper investigates the use of different levels of peripheral modeling for firmware fuzzing, including hardware, function, and register levels. While hardware-level modeling necessitates physical devices, function-level and register-level modeling can be achieved through emulation. To compare the performance of firmware fuzzing at the function and register levels, we conducted an experiment using recent fuzzers, including HAL-Fuzz, P ${ }^{2}$ IM, Fuzzware, and HEFF. HAL-Fuzz employs function-level modeling, while $\mathrm{P}^{2} \mathrm{IM}$ and Fuzzware utilize register modeling. HEFF uses duallevel modeling at both functional and register levels [15]. We tested these fuzzers on the Drone firmware [12], and the results are presented in Table 9. The experiment indicates that the fuzzing speed of register-level fuzzers (including duallevel fuzzers) is approximately half as fast as the fuzzing speed of HAL-Fuzz, a function-level fuzzer. These results suggest that function-level fuzzing is a more efficient approach.

The difference in fuzzing speed between function-level and register-level fuzzing (including dual-level) is due to the additional processing overhead incurred by register-level fuzzing as it handles all accessed registers (also partially handles accessed registers). Firmware vulnerabilities can arise from processing inputs received through peripherals. We provide two examples of vulnerabilities resulting from buffer overflow in this paper. In Example 1, a vulnerability occurs in line 4, where an external input is received through the HAL function and stored as a variable. In Example 2, an external input is saved as a variable, leading to a vulnerability. While both examples use HAL functions, the vulnerabilities arise outside of the HAL function, not within it. In the above-mentioned case, register-level emulation handles all accesses made inside the HAL function, whereas function-level emulation handles functions with pre-made handlers, thus avoiding any processing overhead.

The necessity of identifying pseudo-HAL functions. Figure 1 illustrates the structure of STM32 firmware, where the HAL acts as an intermediate layer

Table 1. Peripheral related functions in CNC firmware

| Firmware | Pseudo-HAL | HAL |
| :---: | :---: | :---: |
| CNC | dirn_wr enable_tim_clock enable_tim_interrupt enable_usart_clock g540_timer_init g540_timer_start g540_timer_stop gpio_clr gpio_init gpio_rd gpio_set gpio_toggle mc_dwell set_step_period set_step_pulse_delay set_step_pulse_time step_isr_disable step_isr_enable step_timer_init step_wr SystemClock_Config SystemCoreClockUpdate SystemInit TIM2_IRQHandler usart_getc usart_init usart_putc usart_tstc | HAL_DeInit <br> HAL_DisableCompensationCell <br> HAL_EnableCompensationCell <br> HAL_GPIO_DeInit <br> HAL_GPIO_EXTI_IRQHandler <br> HAL_GPIO_Init <br> HAL_GPIO_ReadPin <br> HAL_GPIO_TogglePin <br> HAL_GPIO_WritePin <br> HAL_Init <br> HAL_RCC_ClockConfig <br> HAL_RCC_DeInit <br> HAL_RCC_GetHCLKFreq <br> HAL_RCC_GetOscConfig <br> HAL_RCC_GetPCLK1Freq <br> HAL_RCC_GetPCLK2Freq <br> HAL_RCC_GetSysClockFreq <br> HAL_RCC_MCOConfig <br> HAL_RCC_NMI_IRQHandler <br> HAL_RCC_OscConfig |
| $\underline{\operatorname{Total}(\#) \mid}$ | 28 | 20 |

between hardware and software, directly writing values to MCU registers or controlling peripheral devices. The HAL is a universal library commonly employed by developers to manage peripheral devices in firmware implementation. Tools like HALucinator and HAL-Fuzz are used to identify and hook these HAL functions for handling. The HAL function identification program proposed in [11], called LibMatch, is currently employed for this purpose. This enables firmware to operate without requiring physical peripheral devices or separate peripheral emulations. However, LibMatch has two significant limitations due to its reliance on a context-matching technique between the target firmware and the HAL function object file to extract HAL function information.
A lot of information is required. The first limitation of LibMatch is that it necessitates the SDK (object file of the HAL functions) to be compiled in an


Fig. 2. Result of libmatch HAL function identification according to SDK and firmware combination by compile optimization level
environment with the same compiler version and optimization level as the target firmware. Figure 2 displays the LibMatch function identification results for six types of compile optimization levels of the target firmware and their corresponding SDKs. The x-axis represents the optimization options for firmware, while the $y$-axis represents the optimization options for the SDK. For example, in Figure 2, the matrix $(0,0)$ represents $96.4 \%$ of the matching HAL function ratio when the firmware is built with the -O0 option and the SDK is built with the -O0 option, using the libmatch extraction method. When the optimization levels match ( 6 out of 36 ), a high matching rate ranging from $67.9 \%$ to $96.4 \%$ is achieved. However, in most cases where the optimization levels do not match (30 out of 36 ), function search is either impossible or, even if a match is identified, the matching rate is below $20 \%$. This indicates that Libmatch has a high dependency on the SDK files. If it fails to find an SDK that matches the optimization options of the target firmware, the matching ratio of HAL functions decreases.
Unidentified functions exist. The second limitation of LibMatch is that it can only identify HAL functions, as the required SDK file contains only HAL function information. Consequently, functions other than HAL functions cannot be identified by LibMatch. However, as demonstrated in the CNC [7] firmware example in Table 1, not all firmware exclusively depends on HAL functions to control their peripherals. In such cases, developers define and utilize functions that behave like HAL but can be controlled in smaller units for convenience. These functions, referred to as pseudo-HAL functions in this study, perform functions using registers assigned to peripheral devices while accessing within the range of the HAL functions and MMIO. Therefore, for scalable function-level firmware fuzzing, it is crucial to identify both HAL and pseudo-HAL functions.


Fig. 3. PHI system flow

### 2.3 Our approach

We propose the use of pseudo-HAL function identification for effective and scalable firmware fuzzing at the function level. Pseudo-HAL functions are identified based on register access patterns at the register level. This can be accomplished through symbolic execution of MMIO and identifying characteristic offset information for each function. This approach reduces the reliance on the SDK compilation environment and enables fuzzing of a wider range of firmware than HAL-Fuzz. In the next section, we will provide a detailed description of our PHI system.

## 3 System Design

### 3.1 System Overview

In this section, we present an overview of the PHI (Pseudo-HAL Identification) system, which involves a two-input, three-step process, as illustrated in Figure 3. The user provides the target firmware and the corresponding MCU (Microcontroller Unit) name as inputs. The MCU name is used for selecting the appropriate DB (Database) file, while the firmware is utilized for feature extraction to identify functions related to peripheral devices. The PHI process comprises three steps: DB configuration, feature extraction, and feature comparison. DB configuration (Section 3.2) is the first step, which involves creating a DB for each MCU prior to the PHI operation and selecting the appropriate DB based on the input MCU name. The second step, feature extraction (Section 3.3 ), extracts the function features from the firmware using symbolic execution. This step is the most computationally intensive and involves the extraction of three features for each peripheral access. In the final step, feature comparison (Section 3.4), the functions used in the firmware are identified by matching the extracted features with the DB. The extracted files in this step are utilized for fuzzing.

### 3.2 DB Configuration

The process of configuring the DB includes two primary steps: DB creation and DB selection. DB creation involves extracting the features of peripheral

|  |  |  |  |
| :--- | :---: | :--- | :--- |
| ADC_GetResolution | 0 | $0 \times 40012000$ | $0 \times 4$ |
| ADC_IsEnabled | 0 | $0 \times 40012000$ | $0 \times 8$ |
| ADC_INJ_SetOffset | 1 | $0 \times 40012000$ | $0 \times 14$ |
|  | $\ldots$ |  |  |
| USART_IsEnabledIT_TXT | 0 | $0 \times 40011000$ | $0 \times c$ |
| USART_SetStopBitsLength | 1 | $0 \times 40011000$ | $0 \times 10$ |
| USART_TransmitData9 | 1 | $0 \times 40011000$ | $0 \times 4$ |
|  | $\ldots$ |  |  |

Fig. 4. Example of DB
functions used in each MCU from MMIO (Memory-Mapped Input/Output) and offset that can be called from the embedded board, and converting them into a database. This process is essential for obtaining the necessary information to accurately map the functions used in the firmware to the MCU. It involves analyzing the registers used and their corresponding states, as well as dividing the base address and offset of each peripheral device to enable further classification. As a result, the database structure can be represented as <func_i, state_rw, peri_addr, offset>. Figure 4 illustrates an example of a database (DB). DB includes the name of low-level functions (func_i), whether the function involves reading or writing to the MMIO registers (state_rw), the MMIO address associated with the function (peri_addr), and the register access offset (offset). For the indication of reading or writing to MMIO registers, 0 represents the state of reading from the MMIO register, and 1 represents the state of writing to the MMIO register. In the first row of the Figure 4, ADC_GetResolution represents the function name, 0 indicates reading from the MMIO register, $0 \times 40012000$ indicates the base address of peripheral and $0 \times 4$ indicates the offset for accessing the MMIO register.

DB selection, on the other hand, is the process of selecting the appropriate DB based on the MCU name input for PHI. This step is crucial for effective and accurate PHI operation. These DBs are stored in a single folder, and DB selection is the process of selecting a DB corresponding to the entered MCU name. The reason for configuring various DBs is that the register addresses used for each MCU are different, and selecting the correct DB ensures the proper mapping of peripheral functions to the specific MCU.

### 3.3 Feature Extraction

The feature extraction step extracts the features of functions called when the target firmware is executed using symbolic execution, a static analysis technique. Typically, to identify functions at the function-level, an object file containing function information is necessary, as in the case of LibMatch results. However, this paper proposes a register-level function detection approach that extracts function features from all register-level accesses without requiring detailed function information, such as function names. As a result, we leverage symbolic

Table 2. Information of USART

| Register |  | Offset |
| :--- | :--- | :--- |
| Status Register | SR | $0 \times 00$ |
| Data Register | DR | $0 \times 04$ |
| Baud Rate Register | BRR | $0 \times 08$ |
|  | CR1 | $0 \times 0 \mathrm{C}$ |
| Control Register | CR2 | $0 \times 10$ |
|  | CR3 | $0 \times 14$ |
| Guard Time and Prescalar Register GTPR | $0 \times 18$ |  |

execution to identify functions at the register level without relying on detailed information, instead of using a matching method that requires such information. This approach is possible because peripheral registers in firmware are assigned to specific memory ranges, such as the MMIO range of $0 x 40000000-0 x 5 f f f f f f f$ for ARM Cortex-M4 MCUs, for example.

Consider the case of USART, which manages asynchronous serial communication between computers. In an ARM Cortex-M4 MCU, the peripheral base address for USART is $0 \times 40011000$, and offsets such as SR, DR, BRR, CR, and GTPR are allocated to it, as shown in Table 2. By utilizing these offsets and their corresponding USART functions, which control USART using the related registers, it is possible to identify functions at the register level without the need for detailed information, such as function names. MMIO ranges, peripheral base addresses, and offset information can be obtained from the datasheet for each MCU, facilitating the construction of this information. Therefore, to extract the features of functions related to firmware peripherals, the following steps are performed:

1. List the functions that access the MMIO range.
2. Check the base address and offset used by each function.
3. Record whether the function reads or writes to that memory.

To accomplish this, the top-level parent node is first extracted from the target firmware. Then, the function call flow within the firmware is checked, starting from all parent nodes. All accesses that read or write memory information within the MMIO address range are recorded. These accesses are listed by creating the tuple <instruction address (ins_addr), block address (block_addr), state_rw, peri_addr, and offset>. Typically, functions can access the MMIO range multiple times, and memory reads/writes can occur sequentially. If a function has a continuous sequence of the same type of operation, such as read/read/read/... or write/write/write/..., the sequence of accesses is summarized into a single input. However, if both read and write operations occur in the same function with the same offset, they are summarized as a write operation because the same offset is read and written when writing to a specific register for a function.

### 3.4 Feature Comparison

In the feature comparison step, a list of functions for fuzzing is extracted by matching the feature extraction results, which consist of instruction address, block address, status (read or write), peripheral base address, and offset, with the previously constructed database. These function names are used as keys when connecting to a function handler after function hooking. The corresponding function search result field is the same as that of $<f u n c \_i$, ins_addr $>$. In this step, the corresponding results are extracted to a file and used for function hooking during fuzzing.

## 4 Implementation

In this study, we implemented PHI, PHI-Fuzz, and a handler. PHI takes the firmware binary and the name of the MCU on which the firmware is loaded as input, then selects the DB corresponding to the MCU name. The PHI is implemented as a Python script consisting of 479 lines, which configures the function information DB , totaling 972 lines of code.

To configure the DB and identify the pseudo-HAL, PHI utilizes angr [8], a symbolic execution tool. The angr functions used include Control-Flow Graph (CFG) analysis and Data Dependency Graph (DDG) results. The CFG functions were divided into CFGFast and CFGEmulated. CFGFast was employed to extract the parent node, while CFGEmulated (with a call depth of 7) was used to extract the DDG.

PHI-Fuzz is implemented based on HAL-Fuzz and receives the PHI result as an addr.yaml file, saves it, and fuzzes the target firmware through a modified handler. The essential handler functions for fuzzing were implemented by adding them to the existing HAL function handler file. Specifically, the existing HAL function handler was connected with the pseudo-HAL function, which played a similar role, to enable fuzzing. Functions discovered through PHI that could not be replaced with existing functions were implemented and added to the existing handler file.

## 5 Evaluation

The evaluation of PHI-Fuzz was experimentally conducted to answer the following research questions:

- RQ1: How scalable is a PHI that uses only firmware images for identification?
- RQ2: How effective is the PHI in terms of function identification?
- RQ3: How good is the PHI-Fuzz in Bug finding?

Table 3. Firmware tested in Section 5.2, 5.3, 5.4

| Firmware | MCU | OS | Library |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | Peripherals |  |  |
| GPIO UART I2C SPI |  |  |  |

### 5.1 Experimental Setup

Experimental environment. Experiments for PHI and PHI-Fuzz evaluation were conducted in an Intel® Core ${ }^{T M}$ i7-8700 CPU @ 3.20 GHz , 8GB RAM, and Ubuntu 18.04.4 LTS (VM) environment.

## Experiment data.

Table 3 presents the information on the firmware used to evaluate PHI and PHI-Fuzz. The firmware was based on STM32F469NI and STM32F103RB, with the source code collected from an open-source project on GitHub and then ported for use. The per firmware included GPIO, UART, I2C, and SPI for evaluation. In total, four HAL-based firmware and ten pseudo-HAL-based firmware were created and used for the experiments. Additionally, one HALucinator benchmark firmware and two $\mathrm{P}^{2} \mathrm{IM}$ benchmark firmware were used in the experiment. The firmware was compiled without optimization using the 2018_q4 (gcc8) version. The HAL object file required for Libmatch, a program that compares with PHI, was also compiled with the 2018_q4 (gcc8) version and without optimization.

PHI: Pseudo-HAL Identification for Scalable Firmware Fuzzing
Table 4. PHI result of UART_Hyperterminal_IT by Optimization level

| Optimization level | Total(\#) Result(\%) |  |
| :--- | :---: | ---: |
| -00 No optimization | 16 | 69 |
| -01 Reduced code size, execution time | 15 | 75 |
| -03 Optimization of inline functions and registers | 15 | 75 |
| -Os Omit optimizations that increase code size | 15 | 75 |
| -Og Remove optimizations that confuse debugging | 15 | 75 |

### 5.2 Scalability of PHI (RQ1)

To demonstrate PHI's scalability, this study shows that identifying pseudoHAL functions is feasible with only the MCU name, without relying on detailed firmware information. To validate this claim, function identification experiments were conducted on compiled firmware at various optimization levels, and the function identification rates were compared with LibMatch's HAL function identify results when the compiler versions of the SDK file and the target firmware differed. The reason for demonstrating scalability through results obtained with different compilation options is that LibMATCH, which uses the specific SDK, exhibits varying results depending on compilation options, as shown in Figure 2. Therefore, by achieving consistent results without using the SDK, PHI establishes its scalability. TABLE 4 presents the PHI results for the UART_Hyperterminal_IT [11] firmware compiled at different optimization levels using the same source code. Optimization led to a reduction of one in the total number of peripheral-related functions (HAL functions), but at all optimization levels, 15 identical pseudo-HAL function identifications were possible. In comparison, LibMATCH's identification rate varies depending on the compilation level of the SDK and firmware, unlike PHI, which not only requires the SDK but also shows consistent identification results in target firmware compiled at each optimization level.

As an additional experiment, a comparison experiment was conducted by detecting with a different compiler. While the original experimental firmware and SDK files were compiled with 2018_q4 (gcc8), for this experiment, only the experimental firmware was compiled with $2016 \_q 4$ (gcc6) to compare the results in the unideal environment. Figure 5 and 6 show the results of PHI and LibMATCH with four types of firmware that utilize HAL functions and compiled with 2018_q4 (gcc8) and 2016_q4 (gcc6) each. Figure 5 represents the identification results in an ideal environment for using LibMatch. As a result, PHI exhibited an average exploration rate of around $69 \%$, while LibMATCH showed an average exploration rate of approximately $75 \%$. Figure 6 illustrates the results of experiments conducted using firmware compiled with 2016_q4 (gcc6), which did not occur in an ideal environment. PHI, since it doesn't rely on the SDK, produced the same results as the exploration with the firmware compiled with 2018_q4 (gcc8). However, LibMatch did not achieve the same results. LibMatch detected only NVIC-related functions, resulting in detection performance of up to


Fig. 5. Comparison of HAL function identification rates between PHI and LibMatch. The figure shows the execution outcome of the LIBMATCH with the ideal compiler version.


Fig. 6. Comparison of HAL function identification rates between PHI and LibMatch. The figure shows the execution outcome of the LibMatch without the ideal compiler version.
$17 \%$ or less. As a result of these experiments, it was confirmed that PHI can explore functions consistently across various compilation optimization options and compiler versions, demonstrating its scalability as a program. With this scalable feature of PHI, it is possible to detect peripheral-related functions in commercially available firmware without prior information. These detection results can subsequently be used for vulnerability exploration through PHI-Fuzz. The experimental results related to this will be presented in Section 5.4.

### 5.3 Effectiveness of PHI (RQ2)

In Section 5.2, it was observed that LibMatch's identification rate is favorable when the SDK is in an ideal environment. Therefore, in this section, we compare LibMatch and our approach in the ideal environment. Generally, the HAL function identification rate of PHI closely resembled LibMatch's

Table 5. HAL function identification result for SPI_receive firmware

| Function | Libmatch PHI |  |
| :--- | :---: | :---: |
| HAL_GPIO_Init |  | $\checkmark$ |
| HAL_NVIC_SetPriority | $\checkmark$ |  |
| HAL_NVIC_SetPriorityGroup | $\checkmark$ |  |
| HAL_RCC_ClockConfig | $\checkmark$ | $\checkmark$ |
| HAL_RCC_GetHCLKFreq | $\checkmark$ |  |
| HAL_RCC_GetPCLK1Freq | $\checkmark$ | $\checkmark$ |
| HAL_RCC_GetPCLK2Freq | $\checkmark$ | $\checkmark$ |
| HAL_RCC_GetSysClockFreq | $\checkmark$ | $\checkmark$ |
| HAL_RCC_OscConfig | $\checkmark$ | $\checkmark$ |
| HAL_SPI_Init |  | $\checkmark$ |
| HAL_SPI_MspInit |  | $\checkmark$ |
| HAL_SPI_Receive |  | $\checkmark$ |
| HAL_SPI_Transmit |  | $\checkmark$ |
| HAL_SPI_TransmitReceive |  | $\checkmark$ |
| HAL_UART_Init | $\checkmark$ |  |
| HAL_UART_MspInit | $\checkmark$ | $\checkmark$ |
| HAL_UART_Transmit | 10 | 12 |
| Total |  |  |

rate (as shown in Figure 5). However, for UART_transmit, UART_receive, and I2C_receive firmware, LibMatch displayed a higher search rate than PHI. What could be the reason? The functions identified by LibMatch but not by PHI were NVIC-related functions, specifically HAL_NVIC_SetPriority and HAL_NVI C_SetPriorityGrouping. PHI failed to identify these functions because the NVICrelated DB configuration was not established in PHI since the access address was outside the MMIO range. Conversely, for SPI_receive firmware, PHI exhibited a higher search rate than LibMatch. In Table 5, while PHI did not identify two NVIC-related functions, LibMatch could not identify four other SPI-related functions. This confirms that LibMatch cannot identify all HAL functions, whereas PHI can identify functions that LibMatch cannot.

Additionally, Table 6 shows the results of another function identification experiment using 10 firmware that call pseudo-HAL functions instead of HAL functions. While LibMatch had a detection rate of $0 \%$, PHI could identify functions at a significantly high rate of $92.3 \%$. As a result, PHI can identify HAL functions with performance similar to or even superior to LibMatch, which has access to all SDK information, even without utilizing the SDK. Additionally, PHI can also identify pseudo-HAL functions that were previously inaccessible for exploration with LibMatch. Furthermore, similar to the results in Section 5.2 , PHI's effectiveness in detecting a wider range of peripheral-related functions allows for more efficient fuzzing, making it beneficial.

Table 6. Pseudo-HAL function identification(\%)

| Firmware | Libmatch PHI |  |
| :--- | :---: | :---: |
| Baremetal_I2C | 0 | 68.1 |
| FreeRTOS_I2C | 0 | 63.6 |
| Baremetal_UART | 0 | 64.2 |
| FreeRTOS_UART | 0 | 64.2 |
| RIOT_I2C_receive | 0 | 60 |
| RIOT_I2C_transmit | 0 | 62.5 |
| RIOT_SPI_receive | 0 | 60 |
| RIOT_UART | 0 | 64.7 |
| RIOT_SPI | 0 | 92.3 |
| RIOT_I2C | 0 | 84.2 |

Table 7. Fuzzing experiment

| Firmware | HAL-Fuzz PHI-Fuzz |  |
| :--- | :---: | :---: |
| UART_receive | O | O |
| I2C_receive | O | O |
| UART_HyperTerminal_IT | O | O |
| Drone | O | O |
| CNC | X | O |
| Baremetal_I2C | X | O |
| FreeRTOS_I2C | X | O |
| Baremetal_UART | X | O |
| FreeRTOS_UART | X | O |

### 5.4 Effectiveness of PHI-Fuzz in bug finding (RQ3)

To demonstrate the effectiveness of PHI-Fuzz, the fuzzing results of PHI-Fuzz and HAL-Fuzz were compared. Table 7 represents the results of testing the feasibility of fuzzing on nine firmware, using HAL-Fuzz and PHI-Fuzz. Among the experimental firmware, UART_receive, I2C_receive, UART_HyperTerminal_IT, and Drone contain HAL functions, and both HAL-Fuzz and PHI-Fuzz can be used to fuzz these samples. However, CNC, Baremetal_ I2C, FreeRTOS_I2C, Baremetal_UART, and FreeRTOS_UART contain pseudo-HAL functions, and can only be fuzzed using PHI-Fuzz.

Table 8 shows the execution results of HAL-Fuzz and PHI-Fuzz on Drone and CNC. The experimental results reveal that both fuzzers could run on Drone, but only PHI-Fuzz was capable of running on CNC. PHI-Fuzz outperformed in terms of fuzzing execution speed and execution path on Drone, as more functions were identified and handled. Furthermore, PHI-Fuzz discovered six unique crashes not detected by HAL-Fuzz, indicating that PHI-Fuzz demonstrated superior performance in finding bugs.

PHI: Pseudo-HAL Identification for Scalable Firmware Fuzzing
Table 8. Fuzzing experiment with Drone and CNC firmware

| Firmware | HAL-Fuzz <br> Exec. \#Path \#Crash |  | PHI-Fuzz <br> Exec. \#Path \#Crash |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Drone | 2,981,648 | 473 | X 3,511,621 | 491 | $x$ |
| CNC | $x$ | $x$ | X 4,020,289 | 958 | 6 |

Table 9. Drone firmware fuzzing Performance Comparison in terms of execution speed \& a number of basic blocks.

|  | HAL-Fuzz [3] | P $^{2}$ IM [12] | HEFF [15] | Fuzzware [20] | PHI-Fuzz |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Modeling level | Function | Register | Dual | Register | Function |
| Function scalable | HAL | HAL | HAL | HAL | HAL |
| Speed(exec/s) | 49 | Pseudo-HAL Pseudo-HAL | Pseudo-HAL | Pseudo-HAL |  |
| Executed BB (\#) | 254 | 20 | 21 | 23 | 53 |

## 6 Discussion \& Limitation

The results presented in Section 5.2 demonstrate that PHI can effectively identify both pseudo-HAL and HAL functions independently of firmware information, as shown in Section 5.3. Moreover, due to its scalability, PHI can efficiently find bugs, as discussed in Section 5.4. Furthermore, the HAL function identification results in Table 5 reveal that PHI outperforms LibMatch, since it identified four out of the five SPI-related functions that LibMatch failed to identify. However, LibMatch has not yet identified HAL_RCC_GetHCLKFreq and HAL_UART_MspInit. Therefore, to achieve high function coverage during fuzzing, a dual identification technique can be employed. This approach involves first identifying function information through LibMatch and then executing PHI to identify functions related to all peripheral devices within the MMIO range.

Table 9 compares the fuzzing performance of firmware fuzzers at various levels. As seen in the table, PHI-Fuzz exhibits more than twice the speed compared to register-level fuzzers and is $8 \%$ faster than the function-level firmware fuzzer HAL-Fuzz, achieving the best results in terms of fuzzing speed. However, it also obtained the lowest number of executed basic blocks. This is because register-level firmware fuzzers process all registers, resulting in a larger number of executed basic blocks. On the other hand, function-level firmware fuzzers execute a relatively smaller number of basic blocks since they have predefined handlers for each function call. In this context, PHI explored and handled more functions than HAL-FUZZ, leading to the execution of the fewest basic blocks.

## 7 Related Work

Firmware fuzzing for an MCU target requires firmware emulation. Unlike general software, firmware depends on various peripheral devices, making peripheral device emulation the core of firmware emulation. To address this dependency problem of peripheral devices, various firmware emulation studies have been conducted. In this section, we introduce the firmware emulation technique and the latest fuzzers that utilize it.

### 7.1 Firmware Emulation

In WYCINWYC [19], firmware emulation is divided into two categories: full emulation, which emulates both the core and peripheral devices of the firmware, and partial emulation, which emulates only the core device and handles peripheral device emulation through physical hardware or peripheral modeling. Full emulation requires significant engineering effort, as all peripherals must be directly configured into the emulator. In particular, in the case of MCUs, which can have various manufacturers and peripheral devices, directly emulating all of them incurs high costs. On the other hand, partial emulation is proposed to mitigate the inefficient development effort of peripheral devices required during full emulation. This method was studied using hardware-in-the-loop (HITL) and peripheral modeling techniques.

The hardware-in-the-loop emulation handles peripheral access by using real peripheral hardware [17,23]. This approach performs firmware emulation by communicating with peripherals not supported by the emulator using actual peripheral hardware. However, its availability is limited due to the requirement of actual peripheral hardware. On the other hand, peripheral modeling emulates I/O processing for peripheral devices through a model of the peripheral device $[10-13,26]$. This method does not use actual peripheral devices, making it easier to use and reducing engineering efforts. Muench et al. [19] demonstrated that emulation through peripheral modeling is more effective than the HITL method and improves emulation performance.

### 7.2 Hardware-Level Emulation

Peripheral modeling can be categorized into hardware-level, function-level, and register-level modeling based on the modeling level of the peripheral device. Pretender [13] models a peripheral device based on hardware values obtained by inputting values for the actual device. The modeling process uses machine learning, and firmware fuzzing is performed using the implemented model. This is different from the HITL method in that the hardware is used only during the peripheral modeling phase. Thus, fuzzing can proceed without an actual device, relying solely on the modeled result. However, a drawback of this approach is that various hardware is eventually required for the peripheral modeling phase. In contrast, PHI makes it possible to identify functions related to peripheral devices using only firmware binary images and MCU names, without the need for
actual hardware at any stage. This enables more scalable fuzzing than Pretender and other peripheral modeling-based approaches.

### 7.3 Function-Level Emulation

HALucinator [11] is an emulator that allows developers to model peripheral devices of MCU devices directly using the Hardware Abstraction Layer (HAL). Compared to full emulation, which requires detailed modeling of the register unit, HALucinator reduces overhead by allowing developers to directly model the HAL, which is commonly used in many MCU target operating systems. When HAL functions are called, HALucinator handles them by using modeled function handlers. Moreover, HALucinator provides emulation for each peripheral device in the HAL layer, making it possible to fuzz without emulating complex hardware. PHI-Fuzz uses a self-modified HAL-Fuzz function handler for fuzzing. Furthermore, PHI's ability to identify pseudo-HAL functions addresses the limitation of HALucinator, which could only identify HAL functions.

### 7.4 Register-Level Emulation

Compared to HALucinator, which focuses on handling functions, P2IM [12] is designed for dynamic testing and fuzzing of individual I/O devices at the register level. When the firmware is executed in the emulator, P2IM classifies the access pattern of the peripheral's MMIO registers into categories such as $\mathrm{CR}, \mathrm{SR}, \mathrm{DR}$, and C\&SR using a proposed heuristic and performs peripheral device modeling with each register handling method. As a result, P2IM does not require prior knowledge of which specific peripheral devices are connected to the MCU since peripheral device handling is performed automatically. PHI leverages P2IM's register access pattern classification to identify peripheral functions. By analyzing the MMIO information output through DDG, PHI classifies peripherals and calculates the used offset, categorizing them into memories such as SR, DR, and CR. Through this classification process, PHI identifies the accesses performed by the HAL and pseudo-HAL functions. In contrast to P2IM, which automatically creates and operates a handler during fuzzing, PHI-Fuzz requires only a pre-written function handler for the identified function, enabling faster fuzzing.

Laelaps [10] performed firmware emulation through dynamic symbolic execution when an undefined peripheral device access occurred in the emulator while being emulated through QEMU. $\mu E m u$ [26] analyzed register access patterns for peripheral access via symbolic execution, prior to firmware fuzzing. During symbolic execution, rules for responding to unknown peripheral accesses are inferred, stored in the Knowledge Base (KB), and referenced in the firmware analysis. To address the limitations of Laelaps and $\mu \mathrm{Emu}$, Fuzzware [20] proposes a solution for limiting fuzzing coverage expansion through path removal during symbolic execution and partial input overhead. PHI also leverages symbolic execution to extract the called functions. Function identification information is provided through Angr, a symbolic execution tool. The offset used when the address of
the called function is in the MMIO range is extracted, and function matching is performed through this information.

## 8 Conclusion

This study aims to improve firmware fuzzing efficiency by identifying both HAL and pseudo-HAL functions at the register level and implementing PHI and PHI-Fuzz as firmware fuzzers based on HAL-Fuzz. The proposed method was able to identify HAL functions accessing the MMIO range at a comparable level to LibMatch, while also addressing the limitation of LibMatch in identifying pseudo-HAL functions. PHI-Fuzz proved to be more effective in bug finding than HAL-FuzZ, as it discovered additional crashes not found by HAL-Fuzz. However, there are still some functions that LibMatch can identify but PHI cannot. To address this, future work will involve conducting a study that combines LibMatch and PHI to increase the function identification rate.

Acknowledgement. This work was supported by Institute of Information \& communications Technology Planning \& Evaluation (IITP) grant funded by the Korea government(MSIT) (RS-2023-00229400, Development of user authentication and privacy preserving technology for a secure metaverse environment) and by Institute of Information \& communications Technology Planning \& Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2023-00230337, Advanced and Proactive AI Platform Research and Development Against Malicious Deepfakes)

## References

1. State of IoT_spring-2023: https://iot-analytics.com/product/ state-of-iot-spring-2023/
2. National vulnerability database. Accessed: Jun. 1, 2021. [Online]. Available: https: //nvd.nist.gov/vuln/
3. HAL_Fuzz. Accessed: Jun. 1, 2021. [Online]. Available: https://github.com/ ucsb-seclab/hal-fuzz
4. Libmatch. Accessed: Jun. 1, 2021. [Online]. Available: https://github.com/ subwire/libmatch
5. 2019 Embedded markets study.(2019) Accessed: Jun. 1, 2021. [Online]. Available: https://www.embedded.com/wp-content/uploads/2019/11/EETimes_ Embedded_2019_Embedded_Markets_Study.pdf
6. Description of STM32F4 HAL and low-layer drivers: https://www.st.com/resource/en/user_manual/ dm00105879-description-of-stm32f4-hal-and-ll-drivers-stmicroelectronics. pdf
7. $\mathrm{P}^{2} \mathrm{IM}$ real-world firmware samples. Accessed: Jun. 1, 2021. [Online] Available: https://github.com/RiS3-Lab/p2im-real_firmware/tree/ d4c7456574ce2c2ed038e6f14fea8e3142b3c1f7
8. Angr : https://github.com/angr/angr
9. Bellard, F.: QEMU, a fast and portable dynamic translator. In: Proc. USENIX annu. technical conf., FREENIX Track. Berkeley, CA, USA (Apr 2005)
10. Cao, C., Guan, L., Ming, J., Liu, P.: Device-agnostic firmware execution is possible: A concolic execution approach for peripheral emulation. In: Annual Computer Security Applications Conference. pp. 746-759 (2020)
11. Clements, A.A., Gustafson, E., Scharnowski, T., Grosen, P., Fritz, D., Kruegel, C., Vigna, G., Bagchi, S., Payer, M.: Halucinator: Firmware re-hosting through abstraction layer emulation. In: Proc. USENIX Secur. Symp. pp. 1201-1218 (2020)
12. Feng, B., Mera, A., Lu, L.: P2im: Scalable and hardware-independent firmware testing via automatic peripheral interface modeling. In: Proc. USENIX Secur. Symp. pp. 1237-1254 (2020)
13. Gustafson, E., Muench, M., Spensky, C., Redini, N., Machiry, A., Fratantonio, Y., Balzarotti, D., Francillon, A., Choe, Y.R., Kruegel, C., et al.: Toward the analysis of embedded firmware through automated re-hosting. In: Proc. Int. Symp. Research in Attacks, Intrusions and Defenses (RAID). pp. 135-150. Beijing, China (Sep 2019)
14. He, Y., Zou, Z., Sun, K., Liu, Z., Xu, K., Wang, Q., Shen, C., Wang, Z., Li, Q.: Rapidpatch: Firmware hotpatching for real-time embedded devices. In: 31th USENIX Security Symposium (USENIX Security 22) (2022)
15. Hwang, E., Lee, H., Jeong, S., Cho, M., Kwon, T.: Toward fast and scalable firmware fuzzing with dual-level peripheral modeling. IEEE Access 9, 141790141799 (2021)
16. Klees, G., Ruef, A., Cooper, B., Wei, S., Hicks, M.: Evaluating fuzz testing. In: Proc. ACM Conf. Comput. Commun. Secur. (CCS). pp. 2123-2138. Toronto, Canada (Oct 2018)
17. Koscher, K., Kohno, T., Molnar, D.: SURROGATES: Enabling near-real-time dynamic analyses of embedded systems. In: Proc. USENIX Workshop Offensive Technol. Washington, DC, USA (Aug 2015)
18. Mera, A., Feng, B., Lu, L., Kirda, E., Robertson, W.: DICE: Automatic Emulation of DMA Input Channels for Dynamic Firmware Analysis. In: Proc. IEEE Symp. Secur. and Privacy (SP). pp. 302-318. Los Alamitos, CA, USA (may 2021)
19. Muench, M., Stijohann, J., Kargl, F., Francillon, A., Balzarotti, D.: What You Corrupt Is Not What You Crash: Challenges in fuzzing embedded devices. In: Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS). San Diego, CA, USA (January 2018)
20. Scharnowski, T., Bars, N., Schloegel, M., Gustafson, E., Muench, M., Vigna, G., Kruegel, C., Holz, T., Abbasi, A.: Fuzzware: Using precise mmio modeling for effective firmware fuzzing. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 1239-1256 (2022)
21. Spensky, C., Machiry, A., Redini, N., Unger, C., Foster, G., Blasband, E., Okhravi, H., Kruegel, C., Vigna, G.: Conware: Automated modeling of hardware peripherals. In: Proc. ACM Asia Conf. on Comput. Commun. Secur. (Asia CCS). pp. 95-109 (2021)
22. Wright, C., Moeglein, W.A., Bagchi, S., Kulkarni, M., Clements, A.A.: Challenges in firmware re-hosting, emulation, and analysis. J. ACM Computing Surveys (CSUR) 54(1), 1-36 (2021)
23. Zaddach, J., Bruno, L., Francillon, A., Balzarotti, D., et al.: AVATAR: A framework to support dynamic security analysis of embedded systems' firmwares. In: Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS). vol. 23, pp. 1-16. San Diego, CA, USA (February 2014)
24. Zalewski, M.: American funzz lop. Accessed: Jun. 1, 2021. [Online].Available: https://lcamtuf.coredump.cx/afl/
25. Zheng, Y., Davanian, A., Yin, H., Song, C., Zhu, H., Sun, L.: FIRM-AFL: highthroughput greybox fuzzing of IoT firmware via augmented process emulation. In: Proc. USENIX Secur. Symp. pp. 1099-1114. Santa Clara, CA, USA (August 2019)
26. Zhou, W., Guan, L., Liu, P., Zhang, Y.: Automatic firmware emulation through invalidity-guided knowledge inference. In: Proc. USENIX Secur. Symp. (2021)

# Lightweight Anomaly Detection Mechanism based on Machine Learning Using Low-Cost Surveillance Cameras 

Yeon-Ji Lee ${ }^{1[0000-0002-0482-2381]}$, Na-Eun Park ${ }^{1[0009-0008-0939-5762]}$ and Il-Gu Lee ${ }^{1[0000-0002-}$ 5777-4029]<br>${ }^{1}$ Sungshin Women's University, Seongbuk-Gu, Seoul, 02844, Korea<br>\{cselab.lyj, nepark.cse, iglee19\}@gmail.com


#### Abstract

As the need for on-site monitoring using surveillance cameras increases, there has been a growing interest in automation research incorporating machine learning. However, traditional research has not resolved the performance and resource efficiency trade-offs. Therefore, we proposed a lightweight learning model that is more efficient and with minimal performance degradation. The proposed model reduces the resolution of the image until the performance is maintained, finding where the trade-off is resolved for each dataset. Using this, we suggested a real-time lightweight fire detection algorithm. The proposed mechanism is approximately 30 times more memory efficient while maintaining the detection performance of traditional methods.


Keywords: surveillance camera, abnormal detection, CNN

## 1 Introduction

The need for on-site monitoring using surveillance cameras for public management, security, and safety has recently increased. However, interpreting surveillance camera footage is a human task, and as individuals monitor multiple cameras simultaneously, there are clear limitations regarding efficiency and accuracy[1]. When humans manage surveillance cameras, issues arise related to human resources, maintenance costs for installing and managing cameras, and other associated costs[2].

Research on anomaly detection using machine learning is being actively pursued to address these issues. [3-6]. Deep learning, a subset of machine learning, allows training without human intervention and delivers high-level results in object detection, data classification, and natural language processing[3]. In particular, the CNN (convolutional neural networks) model, which directly learns features from datasets, is being utilized for anomaly detection in various fields ranging from medicine to agriculture[4]. However, traditional research has been increasing the resolution of images to the maximum, using ultra-high-resolution images as datasets or relying on high-quality images to enhance the accuracy of CNN models[5]. IoT devices, including surveillance cameras, are constrained in energy, memory, and cost[13-14]. High-quality datasets can maintain high model performance but are unsuitable for real-time surveillance camera detection [6-7]. Various research has been conducted for lightweight CNN learning[8-

11]. However, traditional studies have not effectively addressed the trade-off between model performance and cost, underscoring the need for further research in this area.

The proposed model identifies the optimal resolution where the CNN model can maintain its performance on the fire dataset and suggests a more efficient fire detection mechanism. The suggested mechanism maintained a lowered camera resolution and switched to a clearer quality when the likelihood of fire detection exceeded a threshold. The contributions of this study are as follows.

- By classifying the fire dataset based on the fire size and adjusting the resolution to identify the point at which accuracy is maintained, we have addressed the trade-off issue between performance and cost, a limitation of previous research.
- We proposed a universal mechanism not limited to surveillance cameras, making it easier for lightweight CNN learning to be applied across various research and environments.
- We proposed a lightweight fire detection mechanism that maintains the performance of fire detection while reducing memory consumption by 31.8 times.
The structure of this study is as follows: In Section 2, we investigate and analyze research aimed at improving overheads in deep learning training and memory consumption. Section 3 introduces the proposed model and suggests a lightweight fire detection mechanism. Section 4 analyzes the experimental environment, content, and results, and Section 5 concludes with an introduction to future research.


## 2 Related Works

Various studies are being conducted to address issues like training time and memory consumption in data learning using images. This section compares and analyzes previous research, describing the limitations of past studies and the contributions of our proposed research.

In the study proposed by [8], a lightweight, intelligent CNN model was designed to reduce the computational cost of the model. The research addressed power consumption limitations when converting analog signals to digital signals and the computational cost aspects of the image sensor module. Two lightweight CNN models were implemented by reducing the bit precision of the analog-digital converter (ADC) to save power and reduce the number of parameters. The paper experimented with the designed pipeline in MobileNetv2 and GhostNet architectures to assess their generalization capability and performance. While the study demonstrated the generalization ability and reduced power consumption of the model, it could not resolve the slight decrease in model accuracy when reducing ADC bit precision. Additionally, there were limitations related to the dataset, this paper uses high-quality, high-capacity, advanced datasets to improve model performance, making it unsuitable for use in lightweight models. In the study [9] aimed at addressing power consumption in image and video processing and computational cost issues of computer vision applications, an intelligent compression system was proposed to solve the power consumption problem during wireless capsule endoscopy video processing. A deep learning-based classification feedback loop was proposed to determine the importance of images. Important images were enhanced to
include additional content, while the less important ones were compressed into lower quality for storage. In this study, we conducted compression and classification experiments on wireless capsule endoscopy(WCE) videos to evaluate the performance of the proposed model, verify the gain of the intelligent compression system, and predict the number of additional transmittable images. The experiments demonstrated the contributions of the study by verifying that achieving high compression rates and classification accuracy is possible while maintaining video quality. However, we did not consider the processing time complexity, and the learning and experiments were limited to specific gastrointestinal organs and lesion presence in the data, making it unclear whether we could achieve the same performance in other learning scenarios.

Study [10] aimed at enhancing the speed of predicting anomalies to detect fire situations. It is emphasized that while recognizing patterns with high accuracy is vital, optimization for real-time execution is also critical. The research adopts the capabilities of Deeplabv3+ and the OpenVINO toolkit to propose an approach close to real-time detection, with experiments and evaluations focusing on process acceleration. The results showed an achieved inference process acceleration of $70.46 \%$ to $93.46 \%$. When using a GPU with FP16 precision, the inference process speed was approximately double compared to FP32. This study contributes by considering the accuracy of the detection model and process acceleration and speed in time complexity. However, its limitation lies in analyzing only the impact from a temporal perspective without considering memory availability and accuracy.

In a study [11] using a CNN model trained on actual fire incident images, a custom framework for fire detection was presented using transfer learning. The gradientweighted class activation mapping (Grad-CAM) method was employed to visualize the fire and pinpoint its location. Experiments were conducted using a composite largescale dataset formed by merging the fire detection dataset, DeepQuestAI, Saied, Carlo, and Bansal datasets, and the detection performance was evaluated. Experimental results revealed that while the detection accuracies of GoogLeNet, VGG16, and ResNet50 were $88.01 \%, 64.48 \%$, and $92.54 \%$, respectively, the proposed EfficientNetB0 model exhibited an improved accuracy of $92.68 \%$. However, while traditional research analyses considered model lightweightness and computational costs, this study did not further analyze other metrics besides accuracy. Moreover, while the study introduced EfficientNetB0 as a better method, supposedly lighter than the similarly performing ResNet50, it does not provide concrete evidence to confirm the lightweight nature of the model.

Table 1 summarizes the preceding research that was analyzed.
Table 1. Related research summary table.

| Ref. | Features | Limitation |
| :---: | :--- | :--- |
| [8] | - Research on lightweight, intelligent <br> CNN models for reduced computational <br> cost | - Uses high-quality, high-capacity <br> datasets |
|  |  | - It unsuitable for use in light- |
|  |  |  |


|  | - Proposed method to reduce power consumption by decreasing the bit precision of ADC | - Failed to address the decrease in model accuracy when ADC bit precision is reduced |
| :---: | :---: | :---: |
| [9] | - Research on intelligent compression systems to address power consumption issues during wireless capsule endoscopy video processing | - Time complexity was not considered |
|  | - Proposed deep learning-based classification feedback loop based on importance | - The data used for training and experiments was limited to specific conditions such as lesions and specific digestive organs |
| [10] | - Acceleration of the process speed for fire situation detection models | - Various complexities are mentioned, but only time complexity is considered, without accounting for spatial complexities like memory availability |
|  | - Research on optimization for real-time execution | - Did not conduct performance analysis |
| [11] | - Proposed fire detection framework using transfer learning | - While lightweight and computational cost aspects are mentioned, these metrics are not considered in the experiments |
|  | - Fire visualization and location identification using Grad-CAM | - No evidence is provided to support the claim of proposing a lightweight model |

This research showed that not many actively considered optimization among traditional image and video processing studies. Most previous studies either analyzed performance aspects alone or focused on optimization excluding performance, thereby conducting performance analyses limited to specific areas. Some studies that considered accuracy and complexity simultaneously couldn't resolve the trade-off relationship where an increase in accuracy led to increased complexity and improving the complexity aspect resulted in a decrease in accuracy. Therefore, we proposed a mechanism that detects fire by finding the optimal resolution point while maintaining the CNN model performance to address the trade-off issue and enhance fire detection efficiency.

## 3 Proposed Mechanism

This section details the proposed preprocessing steps and mechanism, elaborating on each stage in depth. First, we explained the criteria used to divide the fire dataset used in the experiment into large fires, medium fires, and small fires. We then discuss how adjusting the resolution helps determine two threshold values. Subsequently, based on the details mentioned above, we discussed the proposed lightweight fire detection mechanism.

### 3.1 Adjusting Resolution



Fig. 1. Flowchart of the mechanism for real-time fire detection.
This study aimed to reduce memory while maintaining performance by reducing image resolution to a point where accuracy is sustained. However, for data where the target object size being detected affects performance, performance differs based on that size. For example, in a medical imaging dataset for tumor detection, one can differentiate between early, middle, and terminal stages based on the tumor size. The early stage would require higher resolution compared to the advanced stage. For reliable experiments, it is necessary to measure the performance separately based on the size of the dataset.

The flow of image resolution adjustment is depicted on the left side of Fig. 1. The original image dataset has a dimension of 224 pixelsAfter adjusting it from 100 to 1 , we aimed to identify the resolution point N where performance remained close to the original. When fire is detected using the model with the lowest performance, we calculate the predicted probability estimates for fire classification to derive the average detection likelihood, denoted as M.

### 3.2 Lightweight fire detection model

After deriving N and M , we proposed a lightweight fire detection mechanism, the diagram shown on the right side of Fig. 1. N represents the threshold value for the minimum resolution, while M serves as the real-time fire detection threshold. In the proposed mechanism, surveillance cameras operate at resolution N , but if they detect a probability exceeding M , they update to a higher resolution. In this context, 'probability' refers to the model's estimation of the likelihood of a fire. When the resolution is ' N, , if the probability exceeds ' M ,' the model increases the resolution and performs the
detection again in the zone where all models converge in accuracy. If the probability surpasses the threshold 'M,' it is classified as an anomaly.

## 4 Evaluation

In this section, we describe the experimental environment for implementing and testing the proposed model, mention the content of the experiments, and discuss the results.

### 4.1 Experimental Environment

We conducted the experiments in an environment with an Intel(R) Core(TM) i910850K CPU, 32.0GB RAM, and 930GB Memory, running on the Windows 10 Pro operating system. The tools used were Anaconda3 and Python version 3.10.9. Table 2 provides information on the modules used.

Table 2. Table of used modules.

| Module Name | Version |
| :---: | :---: |
| keras | 2.10 .0 |
| sklearn | 1.0 .2 |
| numpy | 1.23 .5 |
| matplotlib | 3.5 .3 |
| tensorflow | 2.10 .0 |
| glob | 2.69 .1 |
| pandas | 1.4 .2 |
| seaborn | 0.11 .2 |

The fire-detection dataset is used [12], an image dataset for detecting fires. This study only used a portion of the dataset, and the fire images were manually verified and categorized into large, medium, and small fires. A large fire is where the fire occupies more than half of the image, a medium fire occupies less than half but more than a quarter of the image, and a small fire takes up less than a quarter of the image. Each large, medium, and small fire is trained separately, and the control group of normal images is used identically in all three models. Table 3 shows the ratio and number of images used in each experiment.

Table 3. Distribution of datasets used by experiment.

| Experiment | Image Type | Train | Test | Valid |
| :---: | :---: | :---: | :---: | :---: |
| Large-fire Classification | Large-fire image | 140 | 40 | 20 |
|  | normal image | 140 | 40 | 20 |


|  | medium-fire image | 140 | 40 | 20 |
| :---: | :---: | :---: | :---: | :---: |
| medium-fire Classification | normal image | 140 | 40 | 20 |
| small-fire Classification | small-fire image | 140 | 40 | 20 |
|  | normal image | 140 | 40 | 20 |


| xception_input | input: | [(None, None, None, 3)] |  |
| :---: | :---: | :---: | :---: |
| InputLayer | output: | [(None, None, None, 3)] |  |
| xception | input: | (None, None, None, 3) |  |
| Functional | output: | (None, 2048) |  |
| dropout_4 input: (None, 2048) <br> Dropout output: (None, 2048) |  |  |  |
| output input: (None, 2048) <br> Dense output: (None, 2) |  |  |  |

Fig. 2. Layers and input values for the exception model being used.
In the experiment, we used a transfer learning CNN model. Transfer learning models utilize pre-trained models, which can deliver good performance even with data. This made them frequently used models for training with limited images. Fig. 3 shows the operational scenario of the real-time fire detection mechanism. As for other parameter values, we used three channels, the Adam optimizer, and binary_crossentropy, for the loss function. We conducted the training for ten epochs.

### 4.2 Adjustment of fire image resolution



Fig. 3. Operational scenarios of real-time fire detection mechanism
The first experiment aimed to identify the image resolution range where performance is maintained. We reduced the image resolution from 100 to 1 and conducted a binary classification of fire and non-fire, after which we measured the performance. The experiment adjusted the resolution from 100 to 5 pixels in increments of 5 . However, since the performance converged from 20 to 100 pixels, we only visualized and analyzed from 5 to 20 pixels. Fig. 4 shows the graph depicting the accuracy according to image resolution.


Fig. 4. Evaluation results of detection accuracy by resolution.


Fig. 5. Evaluation results of memory usage by resolution.
For the large-fire category, the model maintains an accuracy of 99.3 up to a resolution of 5 pixels. The medium-fire maintains a high accuracy of 96.6 at 5 pixels. However, for the small-fire category, even though it sustains a high accuracy of 95 at a resolution of 20 pixels, it drops to a lower performance of 85 when the resolution is at 5 pixels. Therefore, for each dataset, the maximum points where the performance is maintained while reducing the image resolution are confirmed to be 5 for both largeand medium-fire and 20 for small-fire. Fig. 5 shows the evaluation results of the memory usage at each resolution. The original size of 224 pixcels consumes approximately 4.6 million bytes. At the performance retention point for large-fire and mediumfire, which is 5 pixcels, it uses 59,550~65,307 Bytes, while the small-fire at a resolution of 20 utilizes $405,496 \sim 416,853$ Bytes. This indicates that large- and medium-fire can reduce memory size by up to 70 times, whereas small-fire can save memory by a factor of 10 .

In the experiment mentioned above, the small-fire detection demonstrated the least effective performance. However, fires typically spread from small to larger ones, and detecting the fire when it is still a small flame is crucial. Therefore, in the subsequent experiment, we will detect fire using the small-fire dataset to devise an efficient and lightweight fire detection algorithm.

### 4.3 Evaluation of a lightweight fire detection model

The second experiment evaluated a lightweight fire detection model system for enhanced memory efficiency and effective detection. The proposed mechanism increased the resolution when the probability exceeded a certain threshold, up to a maximum of 20 pixels. The proposed model aimed to detect fires when they are small, so the experiment primarily focused on small-fire detection from the three tests previously conducted. Performance is assessed by measuring the probability, representing the
likelihood of matching a particular label. We used the predict function provided by scikit-learn for this purpose.


Fig. 6. Evaluation results of probability by resolution.
The probability based on the resolution for small fire is represented in Fig. 6. When the resolution was at 5 , it displayed a probability of $46 \%$ for fire data. At 10 , it showed $74 \%$, and at 20 , it converged to $99 \%$.

## 5 Conclusion

In this study, we proposed a lightweight fire detection model to address the conventional deep learning research limitation of balancing performance with cost. We adjusted the resolution of the images and evaluated the performance for each resolution to determine the threshold value of the proposed model. For the large-fire and mediumfire datasets, a $99.3 \%$ accuracy was demonstrated at a resolution of 5, proving 70 times more memory efficient than the original. Furthermore, the small-fire dataset exhibited a $95 \%$ accuracy at a resolution of 20 , demonstrating it to be ten times more memory efficient. Subsequently, we proposed a two-stage fire detection mechanism, focusing on the small-fire dataset with the lowest performance. This proposed mechanism adjusted the resolution based on the probability of deemed fire and used the measured probability from the small-fire dataset as its threshold. Ultimately, the proposed model proved to be approximately 31 times more memory efficient while maintaining fire detection performance.

However, this study utilized a limited dataset, and various variables may have influenced the experimental results. To derive more reliable results, repetitive testing with vast data is necessary. Therefore, in the future, we plan to conduct experiments
targeting a broader and more diverse dataset and aim to derive trustworthy outcomes through repeated experiments.

## Acknowledgements

This work is partly supported by the Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government(MOTIE) (P0008703, The Competency Development Program for Industry Specialist), and the MSIT under the ICAN (ICT Challenge and Advanced Network of HRD) program (No. IITP-2022-RS-202200156310) supervised by the Institute of Information \& Communication Technology Planning \& Evaluation (IITP).

## References

1. Soheil, V., Kin-Choong, Y.: A CNN-RNN Combined Structure for Real-World Violence Detection in Surveillance Cameras. applied sciences 12(3) (2022)
2. Trong, N., Jean, M.: Anomaly Detection in Video Sequence With Appearance-Motion Correspondence. In: 2019 IEEE/CVF International Conference on Computer Vision, ICCV, Seoul Korea (2019)
3. Asif, K., Quadri S.M.K., Saba, B., Junaid, L.: Deep diagnosis: A real-time apple leaf disease detection system based on deep learning. Computers and Electronics in Agriculture, vol. 298 (2022)
4. Amreen, A., Sweta, J., Mahesh, G., Swetha, V.: Tomato plant disease detection using transfer learning with C-GAN synthetic images. Computers and Electronics in Agriculture, Vol. 187 (2021)
5. Waqas, A., Umair, N., Muhammad, H. Y., Sergio, A. V.: Lightweight CNN and GRU Network for Real-Time Action Recognition. In: 2022 12th International Conference on Pattern Recognition Systems, ICPRS, Saint-Etienne, France (2022)
6. Daiki, A., Tomio, G.: Improving Image Quality Using Noise Removal Based on Learning Method for Surveillance Camera Images. In: 2022 IEEE 4th Global Conference on Life Sciences and Technologies, LifeTech, Osaka, Japan (2022)
7. Muhammad, A. K., Ridha, H., Aiman, E., Moncef, G.: Distributed Inference in ResourceConstrained IoT for Real-Time Video Surveillance. IEEE Systems Journal 17(1), 1512 1523 (2022)
8. Wilfred, K., Amandeep, K., Deepak, M.: An Intelligent System with Reduced Readout Power and Lightweight CNN for Vision Applications. IEEE Transactions on Circuits and Systems for Video Technology (2023)
9. Dallel, B., Said, B., Mohammed, K.: An intelligent compression system for wireless capsule endoscopy images. Biomedical Signal Processing and Control. Vol. 70 (2021)
10. Medina, K., Adnan, S., Amila, A., Emir, B.: Early Stage Flame Segmentation with Deep Learning and Intel's OpenVINO Toolkit. In: 2023 XXIX International Conference on Information, Communication and Automation Technologies, ICAT (2023)
11. Saima, M., Fayadh, A., Sarfaraz, M., Musheer, A., Emine, S. G., Kemal, P.: Attention based CNN model for fire detection and localization in real-world images. Expert Systems with Applications, Vol. 189 (2022)
12. kaggle, https://www.kaggle.com/datasets/tharakan684/urecamain/data, last accessed 2023/10/06
13. So-Eun, J., Ye-Sol, O., Ye-Seul, K., Yeon-Ji, L., Il-Gu, L.: Two-Step Feature Selection Technique for Secure and Lightweight Internet of Things. 2023 32nd International Conference on Computer Communications and Networks, ICCCN, Honolulu, HI, USA (2023)
14. Sun-Woo, Y., Na-Eun, P., Il-Gu, L.: Wake-Up Security: Effective Security Improvement Mechanism for Low Power Internet of Things. Intelligent Automation \& Soft Computing, 37(3), 2897-2917 (2023)

# Side-Channel Analysis on Lattice-Based KEM using Multi-feature Recognition - The Case Study of Kyber 

Yuan $\mathrm{Ma}^{1,2}$, Xinyue Yang ${ }^{1,2}$, An Wang ${ }^{3}$, Congming Wei $\left.{ }^{3(\boxed{ }}\right)$, Tianyu Chen ${ }^{1}$, and Haotong $\mathrm{Xu}^{3}$<br>${ }^{1}$ SKLOIS, Institute of Information Engineering, CAS, Beijing, China<br>${ }^{2}$ School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China<br>\{mayuan, yangxinyue, chentianyu\}@iie.ac.cn<br>${ }^{3}$ School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China<br>\{wanganl,weicm, 3120231257\}@bit.edu.cn


#### Abstract

Kyber, selected as the next-generation standard for key encapsulation mechanism in the third round of the NIST post-quantum cryptography standardization process, has naturally raised concerns regarding its resilience against side-channel analysis and other physical attacks. In this paper, we propose a method for profiling the secret key using multiple features extracted based on a binary plaintext-checking oracle. In addition, we incorporate deep learning into the power analysis attack and propose a convolutional neural network suitable for multifeature recognition. The experimental results demonstrate that our approach achieves an average key recovery success rate of $64.15 \%$ by establishing secret key templates. Compared to single-feature recovery, our approach bypasses the intermediate value recovery process and directly reconstructs the representation of the secret key. Our approach improves the correct key guess rate by $54 \%$ compared to single-feature recovery and is robust against invalid attacks caused by errors in single-feature recovery. Our approach was performed against the Kyber768 implementation from pqm4 running on STM32F429 M4-cortex CPU.


Keywords: Lattice-Based cryptography • Side-channel analysis • Plaintextchecking oracle • Kyber • Convolutional neural network.

## 1 Introduction

Classical public key cryptosystems rely on the intractability of certain mathematical problems. However, the rapid development of quantum algorithms and quantum computers poses a grave threat to these cryptographic schemes in use today. Integer factorization and discrete logarithm problems can be solved in polynomial time using Shor's algorithm [17]. Furthermore, a recent study estimated the possibility of factoring a 2048-bit RSA integer in 8 hours using "20 million noisy qubits" [5]. Therefore, it is necessary to develop novel, postquantum secure cryptographic primitives for long-term security.

In 2016, the National Institute of Standards and Technology (NIST) initiated a process [12] to select the best post-quantum cryptography (PQC) primitives for standardization. In July 2022, NIST announced the first group of winners from its six-year competition [2]. Lattice-based cryptography prevailed, with 3 out of 4 winners, demonstrating their foundational role in PQC standards. Among them, Kyber [16], the KEM part of the Cryptographic Suite for Algebraic Cipher Suite (CRYSTALS), was chosen by NIST as the only public key encryption or key encapsulation mechanism (KEM) algorithm for standardization [2]. Shortly after, the National Security Agency included Kyber in the suite of encryption algorithms recommended for national security systems [1]. Currently, the NIST PQC process has entered the fourth round.

In addition to other desired security properties, NIST has prioritized the resilience against side-channel attacks (SCAs), before deploying these PQC algorithms in real-world applications, particularly in scenarios where an attacker could physically access an embedded device.

SCAs were first introduced by Kocher in 1996 [9]. Research has shown that power consumption, electromagnetic emanations (EM), thermal signatures, or other physical phenomena are often correlated with encrypt and decrypt operations occurring on a device [10]. Thus enabling attackers to extract sensitive information such as the long-term secret key. Based on this approach, several SCAs against lattice-based KEMs in the NIST PQC standardization process have been proposed, such as $[3,6,15,18-21]$. Most of them are chosen-ciphertext attacks (CCAs) due to the fact that NIST PQC KEMs are always targeting CCA security.

The recovery goals of these CCAs can be categorized into two groups: one for decrypted messages recovery $[18,20]$ and the other for key recovery $[3,6,15,19,21]$. Since key recovery is more powerful than message recovery, we focus our study on key recovery SCAs. Guo et al. in [6] first proposed an oracle based on decryptionfailure and instantiated the attack model to complete a timing attack on Frodo KEM. Xu et al. presented a full-decryption-based oracle in [21]. They proved that an attacker only needs 8 traces to recover a specific implementation of Kyber512 compiled at the optimization level -O0. D'Anvers et al. [3] exploited the variable runtime information of its non-constant-time decapsulation implementation on the LAC and successfully recovered its long-term secret key. This key recovery attack, named plaintext-checking (PC) oracle in [14] which was defined as a message-recovery-type attack, finds a link between the long-term secret key and specifically chosen messages and recovers the key by recovering the message. Ravi et al. [15] continue this attack conception by exploiting the leaked side information in Fujisaki-Okamoto (FO) transformation [4] or error correcting codes to propose a generic EM chosen-ciphertext SCA. Qin et al. [13] optimized the approach of [15] in terms of query efficiency. Ueno et al. in [19] further investigated the attack methods against adversaries. More appealing is that they implemented a deep-learning-based distinguisher to assist PC oracle attacks.

Our contributions. In this paper, we proposed a novel multi-feature-based side-channel attack (Multi-feature-based SCA) by extracting profiling information from multi-features. Multi-feature-based SCA constructs templates of each secret key value based on a convolutional neural network (CNN) and successfully recovers the secret key of Kyber768. In addition to improving the success rate of recovering the secret key, our approach also eliminates the occurrence of invalid attacks. In summary, we make the following contributions:

- We propose a new profiling approach named Multi-feature-based SCA, which uses multiple features to build templates for the secret key. Our approach eliminates invalid attacks and can directly recover the secret key values, bypassing the intermediate step of recovering the decrypted message.
- We build a CNN to recognize secret keys. The experimental results prove the huge advantages of CNN in constructing templates, and its recognition accuracy reached around $90 \%$.
- Furthermore, we instantiate the described attack framework on Kyber768 and show the details in each step of the new procedure. Compared to Ueno et al.'s [19] method, our approach demonstrates an average success rate enhancement of $27.45 \%$. Additionally, when contrasted with Ravi et al.'s [15] method, our approach exhibits an average attack success rate improvement of $53.69 \%$.

Outline. The remainder of this paper is organized as follows. In Sect. 2, we examine the details of Kyber and the conception of binary PC oracle. Then we enumerate some previous SCAs on it. Sect. 3 outlines the basic idea of our approach, Multi-feature-based SCA. In Sect. 4, we detail our experimental setup and illustrate our attack method and the CNN construction we used. We further demonstrate the effect of our approach on improving the probability of attack success. Lastly, Sect. 5 concludes our work.

## 2 Background

### 2.1 Kyber and the binary PC oracle

KEM is a public key cryptographic primitive that encapsulates a secret key. Kyber is a chosen-ciphertext secure (CCA-secure) KEM based on the Modulelearning with error (M-LWE) problem. The M-LWE problem evolves from the Ring-LWE (R-LWE) problem, with their theoretical basis being to add noise to the $\mathbf{b}=\mathbf{A s}$ problem, making it difficult to recover $\mathbf{b}=\mathbf{A s}+\mathbf{e}$. However, in R-LWE problem, s and each column of $\mathbf{A}$ are chosen from a polynomial ring, while in M-LWE, s and each column of A are selected from a module. Therefore, the M-LWE problem offers more flexibility and computational efficiency.

In Kyber, define a polynomial ring $\mathcal{R}_{q}=\mathbb{Z}_{q}[x] /\left(x^{n}+1\right)$, where modulus $q=3329$ and $n=256$. For every polynomial $f(x)=a_{0}+a_{1} x+a_{2} x^{2}+\cdots+$ $a_{n-1} x^{n-1} \in \mathbb{R}_{q}$, each coefficient $a_{i} \in \mathbb{Z}_{q}(0 \leq i \leq n-1)$, represents a ring with
all elements are integers modulo $q$. Additions, subtractions, and multiplications of polynomials all require modulus $x^{n}+1$. We use bolded uppercase letters for matrices and bolded lowercase letters for polynomial vectors. Matrix $\mathbf{A} \in \mathcal{R}_{q}^{k \times k}$, where its vector $(\mathbf{A}[0], \cdots, \mathbf{A}[k-1])$ represent a polynomial. s, e $\in \mathcal{B}_{\eta}^{k}$, where $\mathcal{B}_{\eta}$ represents the centered binomial distribution with parameter $\eta$, and can be generated by $\sum_{i=1}^{\eta}\left(a_{i}-b_{i}\right)$. In Kyber, $a_{i}$ and $b_{i}$ are uniformly random samples independently selected from $\{0,1\}$.

Based on the above, Kyber provides three security levels with Kyber512 (NIST Security Level 1), Kyber768 (Level 3) and Kyber1024 (Level 5) with dimension $k=2,3$ and 4 respectively. In this paper, we focus on the implementation of Kyber768, but our approaches can also be applied to the other two sets. Parameters in Kyber768 are shown in Table 1. $k=3$ means secret key sk has 3 polynomials. $\left(\eta_{1}, \eta_{2}\right)=(2,2)$ means the coefficients in sk belong an integer between $[-2,2]$. $\left(d_{u}, d_{v}\right)$ were used in Compress and Decompress fuction.

Table 1. Parameters used in Kyber768

|  | Parameters |  |  |  |  |
| :--- | :---: | :---: | :---: | :---: | :---: |
|  | $n$ | $q$ | $k$ | $\left(\eta_{1}, \eta_{2}\right)$ | $\left(d_{u}, d_{v}\right)$ |
| values | 256 | 3329 | 3 | $(2,2)$ | $(10,4)$ |

Generally, a KEM consists of key generation, encapsulation, and decapsulation. But PC-based SCA is only against the decapsulation part. Thus, in Algorithm 1 and Algorithm 2, we only introduce the main parts of encapsulation and decapsulation of Kyber, ignoring details such as the Number Theoretic Transform (NTT).

Let $\lceil x\rfloor$ denotes the nearest integer to $x$. In the following, we first define two functions, Compress $_{q}(x, d)$ and Decompress $_{q}(x, d)$.

Definition 1. The Compression function is defined as: $\mathbb{Z}_{q} \rightarrow \mathbb{Z}_{2^{d}}$

$$
\begin{equation*}
\operatorname{Compress}_{q}(x, d)=\left\lceil\frac{2^{d}}{q} \cdot x\right\rfloor\left(\bmod 2^{d}\right) \tag{1}
\end{equation*}
$$

Definition 2. The Decompression function is defined as: $\mathbb{Z}_{2^{d}} \rightarrow \mathbb{Z}_{q}$

$$
\begin{equation*}
\operatorname{Decompress}_{q}(x, d)=\left[\frac{q}{2^{d}} \cdot x\right\rfloor \tag{2}
\end{equation*}
$$

We can get in [16], $\operatorname{Compress}_{q}(x, d)$ and Decompress $_{q}(x, d)$ need polynomials for their inputs. The above operation is separately done on each coefficient in the input polynomial. Kyber uses a version of the FO transformation to achieve its stated security goals, i.e., for the chosen-plaintext secure (CPA-secure) to CCA-secure. In the following two algorithms, $\mathcal{G}$ represents a hash operation to

```
Algorithm 1 CCA-secure Kyber KEM based on FO transformation (Encaps)
Input: Public key pk
Output: Ciphertext \(\mathbf{c}=\left(\mathbf{c}_{1}, \mathbf{c}_{2}\right)\), session key \(k\)
    \(\mathbf{m} \leftarrow\{0,1\}^{256}\)
    \((\bar{K}, r)=\mathcal{G}(\mathbf{m} \| \mathcal{H}(\mathbf{p k}))\)
    \(\triangleright c=\operatorname{CPA} . E n c r y p t(\mathbf{p k}, \mathbf{m}, r)\)
        \(\mathbf{A} \leftarrow \mathcal{R}_{q}^{k \times k}\)
        \(\mathbf{r} \leftarrow \mathcal{B}_{\eta_{1}}^{k}, \mathbf{e}_{1}, \mathbf{e}_{2} \leftarrow \mathcal{B}_{\eta_{2}}^{k}\)
        \(\mathbf{u}=\mathbf{A}^{T} \mathbf{r}+\mathbf{e}_{1}\)
        \(\mathbf{v}=\mathbf{p k}^{T} \mathbf{r}+\mathbf{e}_{2}+\operatorname{Decompress}_{q}(\mathbf{m}, 1)\)
        \(\mathbf{c}_{1}=\operatorname{Compress}_{q}\left(\mathbf{u}, d_{u}\right)\)
        \(\mathbf{c}_{2}=\operatorname{Compress}_{q}\left(\mathbf{v}, d_{v}\right)\)
    \(k=\operatorname{KDF}(\bar{K} \| \mathcal{H}(\mathbf{c}))\)
    return \(\mathbf{c}, k\)
```

get a 64 -byte variant meanwhile, $\mathcal{H}$ represents a hash operation to get a 32 -byte variant.

During Algorithm 1, the message generates a 32 -byte $\mathbf{m}$ from the 0,1 space. By mand $\mathcal{H}(\mathbf{p k})$, we can get the pre-shared secret $\bar{K}$ and a random coin $r$. In the encapsulation, a CPA-secure encryption operation is used to output $\mathbf{c}_{1}$ and $\mathbf{c}_{2}$. Then, the shared secret $k$ is calculated from $\bar{K}$ and $\mathcal{H}(\mathbf{c})$ through the key-derivation function (KDF).

```
Algorithm 2 CCA-secure Kyber KEM based on FO transformation(Decaps)
Input: Ciphertext c, secret key sk
Output: Session key \(k\)
    \(\mathbf{p k}, \mathcal{H}(\mathbf{p k}), z \leftarrow \operatorname{UnpackSK}(\mathbf{s k})\)
    \(\triangleright \mathbf{m}^{\prime} \leftarrow\) CPA.Decrypt(sk, c)
        \(\mathbf{u}^{\prime}=\operatorname{Decompress}_{q}\left(\mathbf{c}_{1}, d_{u}\right)\)
        \(\mathbf{v}^{\prime}=\operatorname{Decompress}_{q}\left(\mathbf{c}_{2}, d_{v}\right)\)
        \(\mathbf{m}^{\prime}=\operatorname{Compress}_{q}\left(\mathbf{v}^{\prime}-\mathbf{s k}^{T} \mathbf{u}^{\prime}, 1\right)\)
    \(\left(\bar{K}^{\prime}, r^{\prime}\right)=\mathcal{G}\left(\mathbf{m}^{\prime} \| \mathcal{H}(\mathbf{p k})\right) \quad / *\) Attack loaction */
    \(\mathbf{c}^{\prime} \leftarrow \operatorname{CPA} . \operatorname{Encrypt}\left(\mathbf{p k}, \mathbf{m}^{\prime}, r^{\prime}\right)\)
    if \(\mathbf{c}=\mathbf{c}^{\prime}\) then
        return \(k \leftarrow \operatorname{KDF}\left(\bar{K}^{\prime}, \mathbf{c}\right)\)
    else
        return \(k \leftarrow \operatorname{KDF}(z, \mathbf{c})\)
    end if
```

CCA.Decaps first performs the CPA decryption. In CPA-secure decryption, from $\mathbf{c}_{1}$ and $\mathbf{c}_{2}$ using Compress, we obtained the plaintext $\mathbf{m}^{\prime}$. Then, similar
to CCA.Encaps, CCA.Decaps generates $r^{\prime}$ and $\bar{K}^{\prime}$, and evaluates CPA.Encrypt $\left(\mathbf{p k}, \mathbf{m}^{\prime}, r^{\prime}\right)$. This procedure is called re-encryption. At Algorithm 2 line 8, the algorithm executes equality checking, namely, examines whether the re-encryption result $\mathbf{c}^{\prime}$ is equal to the ciphertext $\mathbf{c}$. If equals, the CCA.Decaps algorithm returns the shared secret $k$ as the ciphertext is valid; otherwise, the algorithm returns a pseudorandom number of $\operatorname{KDF}(z, \mathbf{c})$ (instead of $\perp$ ) as the ciphertext is invalid. Thus, the KEM scheme gives any active attacker no information about the PKE decryption result for invalid ciphertext.

The CPA-secure KEMs are vulnerable to chosen-ciphertext attacks when the secret key is reused. These attacks are generally operated in a key-mismatch or PC Oracle. The working principle of PC oracle is to recover one coefficient of the secret key polynomial at a time. Algorithm 3 depicts the PC oracle, in which the adversary sends ciphertext $\mathbf{c}$ and a reference message $\mathbf{m}$ to the oracle. The oracle tells whether $\mathbf{m}$ equals the CPA decryption result $\mathbf{m}^{\prime}$ or not.

```
Algorithm 3 PC oracle
Input: Ciphertext \(\mathbf{c}\), message \(\mathbf{m}\)
Output: 0 or 1
    \(\mathbf{m}^{\prime} \leftarrow\) CPA.Decrypt(sk, c)
    if \(\mathbf{m}=\mathbf{m}^{\prime}\) then
        return 1
    else
        return 0
    end if
```

The key recovery process is based on the recovery of message $\mathbf{m}^{\prime}$ in Algorithm 3. By constructing the selected ciphertext, we can combine every possible coefficient value in Kyber with a set of oracle response sequences. With multiple queries, we are able to recover this coefficient value. Using the rotation property of the polynomial ring, we are then able to recover the complete secret key polynomial of Kyber.

### 2.2 PC oracle-based SCA attacks

The LWE-based KEM in the CPA model can be upgraded to a CCA-secure KEM through FO transformation. As we described in Section 2.1, using FO transformation, the attacker cannot obtain any prompt information about the decapsulation failure when decapsulating. This theoretically provides a strong security guarantee for CPA security KEM, which can prevent selected ciphertext attacks.

However, with the help of side information, such as analyzing the power or electromagnetic waveforms of certain operations during the decapsulation process, an attacker can directly discover the CPA-secure operations inside the CCA-secure model and launch the same attack.

At CHES 2020, Ravi et al. launched a PC oracle-based SCA attack against NIST KEM by utilizing side information leaked from the re-encryption process in the FO transform [15]. Taking the attack against Kyber as an example, the attacker only needs to control $\mathbf{m}^{\prime}$ to be $\mathrm{O}=(0,0,0,0, \cdots)$ or $\mathrm{X}=(1,0,0,0,0, \cdots)$. In this way, they build a PC Oracle with a side-channel waveform distinguisher. In [15], Ravi et al. used simple Euclidean distances to create a recognizer with profiled waveform templates. More specifically, they first collected two sets of re-encrypted waveforms with $\mathbf{m}^{\prime}=\mathrm{O}$ and $\mathbf{m}^{\prime}=\mathrm{X}$. Then, they performed a Test Vector Leakage Assessment (TVLA) between the two sets to select the Point of Interest (PoI). In the attack phase, they achieve binary classification by computing the Euclidean distance between the collected PoI waveforms and the two waveform templates. If each PC oracle query is correct, then Ravi et al. need 5 queries to recover a coefficient. In total, they need $256 \times 2 \times 5=2560$ queries to recover Kyber512.

After that, Qin et al. improved the query efficiency by using an optimal binary tree similar to Hoffman tree encoding to reduce the average number of queries to recover Kyber512 to 1312, which can be found in [13].

We call all the above recovery methods single-feature recovery, and if the value of the private key cannot be found based on the private key identifier obtained from a set of oracle queries, we call this case an invalid attack.

This type of key recovery approach designed by them cannot always tell the truth due to the influence of ambient noise and the accuracy of the side channel distinguisher itself. And since we cannot determine the location of the error, the complexity of brute force cracking is quite high. Therefore, additional techniques are needed to enhance the recovery procedure or tolerate the error. One commonly used technique is majority voting, which was also used in the Ravi et al. attack. With multiple votes, we can obtain a more accurate Oracle.

### 2.3 Convolutional neural network in SCAs

Convolutional neural networks are a powerful class of neural networks designed for processing image data. It has achieved widespread success across domains, including side-channel analysis. It is not surprising, as deep learning excels at identifying patterns and relationships, which aids in extracting information from power consumption time series. This is especially useful for template attacks.

In [11], Maghrebi et al. first applied deep learning in a side-channel context. They found that against unprotected cryptographic algorithm implementations, DL-based attacks are more effective than machine learning-based and traditional template attacks. Notably, their experimental results show that the feature extraction-based model performed very well on both datasets. This could be explained by the fact that CNN applies a nice features extraction technique based on filters allowing dealing with the most informative samples from the processed traces. The work of [8] also proves this.

At CHES 2022, Ueno et al. used CNN to design a side-channel distinguisher and achieve a similar binary classification [19]. With the CNN distinguisher, they can get higher accuracy of single-feature recognition.

### 2.4 Open problem

We reproduce the method of Ravi and Ueno in [15] and [19] using energy analysis. As an example, 20 coefficient values are recovered, as shown in Fig. 1, the average success rate for recovering a single-feature (i.e., message m') using Ravi's method is $64.58 \%$. However, for recovering the complete label, the entire attack fails even if one-bit feature is incorrectly recovered. Hence, the average success rate of secret key recovery using the method in [15] is only $10.46 \%$.


Fig. 1. The success rates of using Ravi [15] and Ueno [19] methods in recovering message bit and a certain secret key coefficient respectively.

As illustrated in Fig. 1, using Ueno's method in [19], the CNN model leads to significant performance gains, with the average success rate of recovering message $\mathbf{m}^{\prime}$ directly improved from $64.58 \%$ to $81.4 \%$. However, the success rate of secret key recovery using the method in [19] remains only $36.7 \%$.

We also noticed that with both methods in [15] and [19], this attack approach of recovering the secret key value bit-by-bit according to the single-feature of $\mathbf{m}^{\prime}$ has a very large invalid attack space. That is, the recovered binary label string may represent neither the correct secret key value nor the wrong secret key value, but rather a meaningless label string. Shockingly, the average occurrence probability of invalid attacks at $75.17 \%$ in [15], shown in Fig. 2. Although using CNN in [19] reduces the occurrence of this event, the proportion of invalid attacks still reaches over $50 \%$.

So how to improve the success rate of attacks and avoid such invalid attacks?

## 3 Multi-feature-based SCA on Kyber

In this section, we elucidate in detail the methodology for constructing multifeatures of secret key and use it to recover Kyber768 using power analysis attacks. Using this approach, we eliminate the occurrence of invalid attacks.


Fig. 2. Invalid attack rate in Ravi [15] and Ueno [19].

### 3.1 Construction of multiple features

In this part, we describe the full-key recovery framework of the new attack.
All previous attack methods take recovering $\mathbf{m}^{\prime}$ as an intermediate step (including [15] and [19]), with the decrypted message value $\mathbf{m}^{\prime}$ as the profiling target. In contrast, our approach bypasses this intermediate process and directly builds templates for the key. The comparison between the two approaches is illustrated in Fig. 3.


Fig. 3. Our profiling strategy.

In order to eliminate the invalid attack presented above, we propose a new profiling method that builds templates for secret keys from multi-features. We integrate the modeling and matching of $\mathbf{m}^{\prime}$ and build a template for the secret key instead of the decrypted message $\mathbf{m}^{\prime}$. Instead of recovering the key's binary label bit by bit, the new key feature construction method stitches single-features
$\mathbf{m}^{\prime}$ together based on a specific ciphertext query result. Compared to singlefeature recovery, we absorb the invalid attack space into the guess space for the entire secret key value, avoiding such situation.

### 3.2 Our attack scenario

We denote the $i$-th coefficient of the private key polynomial as $\mathbf{s k}[i]$. The overall workflow of the profiling stage and attack stage are shown in Fig. 4 and Fig. 5 , respectively. We assume the adversary can manipulate the target device and collect the leaked power traces during cryptographic operations.


Fig. 4. Profiling stage of the Multi-feature-based SCA of key recovery. The NN model learns to find the combined message bit $\mathbf{m}^{\prime}$.

By querying the PC oracle with constructed ciphertexts multiple times, the attacker obtains a set of pre-modeled power traces with the decrypted message $\mathrm{m}^{\prime}$ being 0 or 1 . Based on the mapping between the chosen ciphertexts and the private key values, the adversary acquires the multivariate feature labels representing the coefficients of the private key polynomial. Using the multivariate feature identifiers for each private key value, we construct the modeled power traces for $\mathbf{s k}[i]$ and label these traces based on the value of $\mathbf{s k}[i]$. Finally, they are fed into the network for training.

During the attack stage, as shown in Fig. 5, the attacker replaces the ciphertext with five preset chosen ciphertexts and polls the decrypted messages $\mathbf{m}^{\prime}$ from the target device by decrypting these five chosen ciphertexts. After that, the five obtained traces are concatenated in order and preprocessed into the $\mathbf{s k}[i]$ template style during the modeling stage. Finally, the preprocessed power trace is fed into the trained network, which will directly output the value of this $\mathbf{s k}[j]$.


Fig. 5. Attack stage of the Multi-feature-based SCA of key recovery.

### 3.3 Generate qualified ciphertexts

The process of obtaining these five chosen ciphertexts is as follows:
In CCA.Decaps of Algorithm 2, an attacker can construct the ciphertext $\mathbf{c}=(\mathbf{u}, \mathbf{v})$. And set $\mathbf{u}=k_{u} \cdot x^{0}$ and $\mathbf{v}=k_{v} \cdot x^{0}$ where $\left(k_{u}, k_{v}\right) \in \mathbb{Z}_{q}$.

Let us take the example of recovering $\mathbf{s k}[0]$ (i.e., the lowest coefficient in the first polynomial of sk). We take a long rectangle to represent a polynomial, and each small rectangle in it represents a coefficient. In Kyber768, the polynomial vector has three dimensions, so sk and $\mathbf{u}^{\prime}$ each have three long rectangles. We omit certain modules, such as Compress operations. The connection between the decrypted $\mathbf{m}^{\prime}$, the selected ciphertext $\mathbf{c}$ and the secret key is as shown in Fig. 6:


Fig. 6. The abstract compute relation for $\boldsymbol{m}^{\prime}$ in line 5 in Algorithm 2. We fill the nonzero coefficients of each polynomial in $\mathbf{m}^{\prime}$, $\mathbf{s k}, \mathbf{u}^{\prime}$, and $\mathbf{v}^{\prime}$ with a different color, with a white rectangle indicating that the coefficient is 0 .

From Fig. 6, we can see that all coefficients in $\mathbf{m}^{\prime}$ except for the lowest coefficient, the remaining are all zeros. This allows the attacker to establish a binary distinguishing identity for $\mathbf{s k}[0]$ by controlling $\mathbf{m}^{\prime}=0 / 1$. By instantiating this binary plaintext checking mechanism through the side channel, $\mathbf{s k}[0]$ can be recovered through multiple queries, and the remaining coefficients of sk can be recovered by exploiting the rotational property of polynomial multiplication in the ring.

Therefore, for the above selected $\mathbf{u}, \mathbf{v}$ (i.e., $\mathbf{u}=k_{u} \cdot x^{0}, \mathbf{v}=k_{v} \cdot x^{0}$ ), the lowest bit of the decrypted message $\mathbf{m}^{\prime}[0]$ can be expressed as:

$$
\mathbf{m}^{\prime}[0]=\left\{\begin{array}{lll}
k_{v}-k_{u} \cdot \mathbf{s k}[0] & \text { if } \quad t=0  \tag{3}\\
k_{v}-k_{u} \cdot-\mathbf{s k}[n-t] & \text { if } \quad 0<t \leq n-1
\end{array}\right.
$$

By iterating through the positions of $t$ from 0 to $n-1$, we can recover the coefficients of the first polynomial in secret key s in the order of $\mathbf{s k}[0],-\mathbf{s k}[n-$ $1],-\mathbf{s k}[n-2], \ldots,-\mathbf{s k}[1]$.

Since the coefficient values of the secret key in Kyber768 are within [ $-2,2$ ], we construct Table 2 to enumerate the mapping between the binary string representation of the decrypted message from a chosen ciphertext and the corresponding secret key value. Where X represents the decrypted $\mathbf{m}^{\prime}=1$, and O represents $\mathbf{m}^{\prime}=0$.

Table 2. Chosen ciphertext pairs

|  | $\left(\boldsymbol{k}_{\boldsymbol{u}}, \boldsymbol{k}_{\boldsymbol{v}}\right)$ |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Coeff. | $(0,0)$ | $(0, q / 2)$ | $(110,657)$ | $(240,2933)$ | $(110,832)$ | $(182,2497)$ | $(416,1248)$ |
| -2 | O | X | X | O | X | O | X |
| -1 | O | X | O | O | X | O | X |
| 0 | O | X | O | O | O | O | X |
| 1 | O | X | O | O | O | O | O |
| 2 | O | X | O | X | O | X | O |

Traces pre-process. As obtained above, Table 2 provides a unique binary label string mapping to each secret key value. Our new profiling approach directly builds templates from this label string to the range of secret key values, instead of mapping the profiled $\mathbf{m}^{\prime}=1$ and $\mathbf{m}^{\prime}=0$ to the binary representation. This expands the original binary message recognition into a 5 -class secret key value recognition problem. In the attack phase, we iterate through the five ciphertexts constructed using Table 2 (last five columns), collecting the power traces over the last four rounds of the hash function during decapsulation for each ciphertext. These are concatenated to form the combined multivariate feature information.

## 4 Experiments

### 4.1 Equipment setup

Our measurement setup is shown in Fig. 7. It consists of the Laptop, the versatile current amplifier, the STM32F429 target board, and the PicoScope 3403D Oscilloscope. We target the optimized unprotected implementation of Kyber768, taken from the public pqm4 library [7], a benchmarking and testing framework for PQC schemes on the 32-bit ARM Cortex-M4 microcontroller. In our initialization, the implementation is compiled with arm-none-eabi-gcc using the optimization flag "-O1". We set the operating clock frequency of the target board to 16 MHz and utilized the power analysis side-channel for our experiments. For traces acquisition, we set the trigger at pin PC6, and the measurement results were collected on the oscilloscope with a sampling rate of $62.5 \mathrm{MSam} / \mathrm{s}$.


Fig. 7. Equipment for trace acquisition and the board used in the experiment.

### 4.2 Target operation

The ensuing problem is how to capture this leakage in the side channel. We assume that the attacker has the ability to completely manipulate the target device and is able to measure the power consumption during the execution of a cryptographic algorithm. Then during the inference phase, the adversary aims at recovering the unknown secret key, processed by the same device, by collecting a new set of power consumption traces. To guarantee a fair and realistic attack comparison, we stress the fact that the training and the attack data sets must be different.

Target Operation. Using the key recovery methods in Sect. 3, we find a chosen ciphertext correspondence that is sufficient to distinguish the values of the polynomial coefficients of the secret key. By means of the binary plaintext checking
oracle described above, the attacker constructs a distinction of the decrypted message $\mathbf{m}^{\prime}$. Exactly through the hash function execution process in the FO transformation, the attacker can amplify the difference of the decrypted message $\mathbf{m}^{\prime}$ from 1 bit message bit to 256 bits.

The KeccakF1600_StatePermute function in $\mathcal{G}$ includes twelve for loops. Therefore, the target option we choose is the last four rounds of the hash operation, as shown in Fig. 8. That is, line 6 in Algorithm 2. The TVLA result of our target operation is as shown in Fig. 9:

(b)

Fig. 8. Original power trace of Kyber768. (a) The whole hash operation $\mathcal{G}$ with twelve for loops in Kyber.KEM.Decaps() (i.e., line 6 in Algorithm 2); (b) The last four rounds of $\mathcal{G}$.

Traces Acquire. We set the STM32F429 microcontroller as a server and our laptop as a client. Every time we selected a random message $\mathbf{m}$ and encapsulated it with the public key into ciphertext $\mathbf{c}$ on the client, then we sent $\mathbf{c}$ to the server through a socket.

During the decapsulation of the profiling stage, we captured power traces and saved O or X (i.e., $\mathbf{m}^{\prime}=0$ or $\mathbf{m}^{\prime}=1$ ) as labels. For each type of template, we collected 9,000 traces, each with a length of 30,000 . Then we combined the


Fig. 9. TVLA results for the last four rounds between $O$ and $X$.
traces in order of the five chosen ciphertexts in Table 2. The templates we get are as shown in Fig. 10, and we only selected two localized positions for zoomedin display (five values of $\mathbf{s k}[i]$ are represented by five lines with different colors respectively):


Fig. 10. Constructed template of $\mathbf{s k}[i]$. (a) Complete template for $\mathbf{s k}[i]$ after trace pre-process; (b) and (c) The expansion of an interval somewhere in the template of $\mathbf{s k}[i]$.

In the attack stage, we only need to poll these five chosen ciphertexts in order and collect the same power traces as in the profiling stage for the same pre-processing.

### 4.3 Model Training

By adjusting the CNN network architecture and hyperparameters, we obtained the CNN model that performs best on our dataset. This model is inherited from [19]. The architecture of which is shown in Table 3. It has seven convolutional layers and four fully-connected layers. In the Function row, conv1d $(F)$ denotes the operation at each layer and $F$ is the filter size. The stride of the filter is two and the padding of it is one. After each convolutional layer, batch normalization and SeLU activation are used, and finally, a $2 \times 2$ size average pooling layer is connected to reduce the dimensionality. The convolutional layers are followed by four fully-connected layers in our network architecture. The first fully-connected layer consists of 1000 neurons. Then followed by two fully-connected layers with 200 neurons each. The final layer has 5 neurons and utilizes softmax activation for classification.

Table 3. NN architecture

|  | Input | Output | Function | Normalization | Activation | Pooling |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Conv1 | $150000 \times 1$ | 4 | conv1d(3) | Yes | SELU | $\operatorname{Avg}(2)$ |
| Conv2 | $75000 \times 4$ | 4 | conv1d(3) | Yes | SELU | $\operatorname{Avg}(2)$ |
| Conv3 | $37500 \times 4$ | 4 | conv1d(3) | Yes | SELU | Avg(2) |
| Conv4 | $18750 \times 4$ | 8 | conv1d(3) | Yes | SELU | Avg(2) |
| Conv5 | $9375 \times 8$ | 8 | conv1d(3) | Yes | SELU | Avg(2) |
| Conv6 | $4687 \times 8$ | 8 | conv1d(3) | Yes | SELU | Avg(2) |
| Conv7 | $2343 \times 8$ | 8 | conv1d(3) | Yes | SELU | Avg(2) |
| Flatten | $1171 \times 8$ | 9368 | flatten |  | - |  |
| FC1 | 9368 | 1000 | dense | - | SELU | - |
| $F C 2$ | 1000 | 200 | dense | - | SELU | - |
| FC3 | 200 | 200 | dense | - | SELU | - |
| $F C 4$ | 200 | 5 | dense | - | Sigmoid | - |

In the following experiments, we employed CUDA 11.6, cuDNN 8.3.0, and Pytorch-gpu 1.13.1 on NVIDIA GeForce GTX 3050 to carry out the NN training. The Adam optimizer is utilized with a learning rate of 0.00005 , the batch size was 128 , and the number of epochs was 50 . We used the cross-entropy loss function during training and validated it after each epoch.

### 4.4 Experimental results and comparison

The loss values of this model trained on our dataset for 50 epochs and the accuracy of the validation set are shown in Fig. 11. After 50 epochs of training, the model's loss stabilizes around 0.9 and the accuracy of the validation set improves to $88 \%$.


Fig. 11. Train loss (a) and validation accuracy (b) of our approach..

As shown in Fig. 12, our approach significantly improves the success probability of recovering secret key values. Compared to Ravi's method [15], the average attack success rate for a secret key value increases by $53.69 \%$. It also outperforms distinguishing message $\mathbf{m}^{\prime}$ using neural networks [19] by $27.45 \%$. Our approach can also tolerate invalid attacks due to errors in single-feature recovery.


Fig. 12. Compare three methods of key recovery success rate (a) and invalid attack rate (b).

## 5 Conclusion

Our Multi-feature-based SCA is a novel attack technique that extracts secret key templates from multivariate features and employs the optimal CNN architecture. All attacks presented in this paper are performed directly on the target device. Our experimental results demonstrate that CNN can significantly improve profiling efficiency as an effective approach. Notably, our approach only uses the traces collected in a single experiment when recovering the secret key. Based on the results, voting across multiple experiments can achieve $100 \%$ attack success rate. Our work reiterates the need for effective countermeasures against side-channel attacks in cryptographic implementations.

Acknowledgements. This work is supported by National Key R\&D Program of China (No. 2022YFB3103800), and the National Natural Science Foundation of China under Grant 62272457 . We thank the anonymous reviewers for their valuable comments and suggestions.

## References

1. Announcing the commercial national security algorithm suite 2.0. Tech. rep. (2022)
2. Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Miller, C., Moody, D., Peralta, R., et al.: Status report on the third round of the nist post-quantum cryptography standardization process. US Department of Commerce, NIST (2022)
3. D'Anvers, J.P., Tiepelt, M., Vercauteren, F., Verbauwhede, I.: Timing attacks on error correcting codes in post-quantum schemes. In: Proceedings of ACM Workshop on Theory of Implementation Security Workshop. pp. 2-9 (2019)
4. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. Journal of cryptology 26, 80-101 (2013)
5. Gidney, C., Ekerå, M.: How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. Quantum 5, 433 (2021)
6. Guo, Q., Johansson, T., Nilsson, A.: A key-recovery timing attack on postquantum primitives using the fujisaki-okamoto transformation and its application on frodokem. In: Annual International Cryptology Conference. pp. 359-386. Springer (2020)
7. Kannwischer, M.J., Rijneveld, J., Schwabe, P., Stoffelen, K.: Pqm4: Post-quantum crypto library for the arm cortex-m4 (2019)
8. Kim, J., Picek, S., Heuser, A., Bhasin, S., Hanjalic, A.: Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 148-179 (2019)
9. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Advances in CryptologyCRYPTO96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18-22, 1996 Proceedings 16. pp. 104-113. Springer (1996)
10. Koeune, F., Standaert, F.X.: A tutorial on physical security and side-channel attacks. International School on Foundations of Security Analysis and Design pp. 78-108 (2004)
11. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: Security, Privacy, and Applied Cryptography Engineering: 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings 6. pp. 3-26. Springer (2016)
12. Moody, D.: Post-quantum cryptography standardization: Announcement and outline of nists call for submissions. In: International Conference on Post-Quantum Cryptography-PQCrypto (2016)
13. Qin, Y., Cheng, C., Zhang, X., Pan, Y., Hu, L., Ding, J.: A systematic approach and analysis of key mismatch attacks on lattice-based nist candidate kems. In: Advances in Cryptology-ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part IV 27. pp. 92-121. Springer (2021)
14. Ravi, P., Roy, S.S.: Side-channel analysis of lattice-based pqc candidates. In: Round 3 Seminars, NIST Post Quantum Cryptography (2021)
15. Ravi, P., Roy, S.S., Chattopadhyay, A., Bhasin, S.: Generic side-channel attacks on cca-secure lattice-based pke and kems. IACR transactions on cryptographic hardware and embedded systems pp. 307-335 (2020)
16. Schwabe, P., Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J., Seiler, G., Stehlé, D.: Crystals-kyber: Algorithm specifications and supporting documentation (version 3.0). NIST Post-Quantum CryptographyRound 3
17. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35 th annual symposium on foundations of computer science. pp. 124-134. Ieee (1994)
18. Sim, B.Y., Kwon, J., Lee, J., Kim, I.J., Lee, T.H., Han, J., Yoon, H., Cho, J., Han, D.G.: Single-trace attacks on message encoding in lattice-based kems. IEEE Access 8, 183175-183191 (2020)
19. Ueno, R., Xagawa, K., Tanaka, Y., Ito, A., Takahashi, J., Homma, N.: Curse of re-encryption: A generic power/em analysis on post-quantum kems. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 296-322 (2022)
20. Wang, R., Ngo, K., Dubrova, E.: A message recovery attack on lwe/lwr-based pke/kems using amplitude-modulated em emanations. In: International Conference on Information Security and Cryptology. pp. 450-471. Springer (2022)
21. Xu, Z., Pemberton, O., Roy, S.S., Oswald, D., Yao, W., Zheng, Z.: Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber. IEEE Transactions on Computers 71(9), 2163-2176 (2021)

# Enhancing Prediction Entropy Estimation of RNG for On-the-Fly Test 

Yuan $\mathrm{Ma}^{1,2}$, Weisong $\mathrm{Gu}^{1,2}$, Tianyu Chen ${ }^{1 \star}$, $\mathrm{Na} \mathrm{Lv}^{1}$, Dongchi Han ${ }^{1,2}$, and Shijie Jia ${ }^{1,2}$<br>${ }^{1}$ SKLOIS, Institute of Information Engineering, CAS, Beijing, China<br>${ }^{2}$ School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China \{mayuan, guweisong, chentianyu, lvna, handongchi, jiashijie\}@iie.ac.cn


#### Abstract

Random number generators (RNGs) play a vital role in cryptographic applications, and ensuring the quality of the generated random numbers is crucial. At the same time, on-the-fly test plays an important role in cryptography because it is used to assess the quality of the sequences generated by entropy sources and to raise an alert when failures are detected. Moreover, environmental noise, changes in physical equipment, and other factors can introduce variations into the sequence, leading to time-varying sequences. This phenomenon is quite common in real-world scenarios, and it needs on-the-fly test. However, in terms of speed and accuracy, current methods based on mathematical formulas or deep learning algorithms for evaluating min-entropy both fail to meet the requirements of on-the-fly test. Therefore, this paper introduces a new estimator specifically designed for on-the-fly min-entropy estimation. To accurately evaluate time-varying data, we employ an appropriate change detection technology. Additionally, we introduce a new calculation method to replace the original global prediction probability calculation approach for accuracy. We evaluate the performance of our estimator using various kinds of simulated datasets, and compare our estimator with other estimators. The proposed estimator effectively meets the requirements of on-the-fly test.


Keywords: On-the-fly test • Entropy estimation • Prediction estimator - Change detection technology . Confidence interval.

## 1 Introduction

In today's cryptographic engineering applications, random numbers have become increasingly important. For instance, in key distribution and mutual authentication schemes, two communicating parties collaborate to exchange information for key distribution and authentication purposes. These random numbers are generated by random number generators that contains entropy sources, and entropy sources are divided into two categories: stationary sources and time-varying sources. Secure random numbers are often used as security primitives for many

[^11]```
Ma, Y. et al.
```

cryptographic applications, so it is necessary to evaluate the quality of random numbers.

Current methods for evaluation are mainly divided into two categories: whitebox test and black-box test. White-box test, also known as theoretical entropy evaluation, requires an understanding of the internal structure and generation principle of the entropy source. It establishes a mathematical model according to appropriate assumptions to calculate the theoretical entropy of the output sequence [8]. However, given the complex and varied structures of many entropy sources, it becomes challenging to model them accurately, thereby limiting the applicability of theoretical entropy evaluation. Black-box test includes statistical test and statistical entropy evaluation: statistical test uses hypothesis-testing methods to conduct tests on the sequence for some properties, determining whether the tested sequence meets the null hypothesis (indicating randomness) or exhibits statistical defects [13]. Nevertheless, it is worth noting that certain specifically constructed pseudo-random sequences may exhibit favorable statistical properties and successfully pass these tests, posing potential security threats. Statistical entropy does not require the knowledge of the internal structure and generation principle of entropy sources. It evaluates the safety of the random numbers from the perspective of "entropy" [15]. In summary, to meet the requirements of generality and security, statistical entropy evaluation has become an indispensable approach.

Statistical entropy evaluation methods can be categorized into two main categories: those based on mathematical and statistical theories, and those based on deep learning. However, some estimators in the former, represented by the NIST SP800-90B standard, have been found to have overestimation and underestimation problems when faced with some typical datasets during entropy evaluation [21]. The latter has a problem of high time consumption. They both don't perform well in time-varying sequence which is common in reality. Thus, in order to detect RNG failures quickly and reliably, we need an on-the-fly test that is suitable for time-varying datasets.

To design a suitable estimator for on-the-fly test, we need solve two issues. Firstly, as mentioned above, we should update the model in a timely manner, especially for time-varying datasets. To address it, we utilize the change detection technique. Secondly, we introduce a new calculation method for global predictability of entropy estimation [15], specifically designed to handle situations involving small samples or extreme probabilities (i.e., probabilities approaching 0 or 1 ), which is different from the SP800-90B Standard, because the raw method is no longer suitable for on-the-fly test.

Our goal is to design an entropy estimator which meets the requirement of speed and accuracy for on-the-fly test. We present several significant contributions in this paper:

1) We propose a modified version of the prediction estimators from SP80090 B , enabling an on-the-fly test for evaluating the quality of entropy sources timely. To support the new framework, we proposed two key technologies: change detection technique and new calculation method for global predictability.
2) By leveraging the characteristics of the prediction estimator model and drawing inspiration from neural network parameter adjustments during training, we design a novel change detection technique suitable for online entropy estimation. Besides, we are the first to address the challenges associated with evaluating min-entropy in scenarios involving small sample datasets and extreme probabilities. We provide a reasonable solution to this issue, which plays a critical role in on-the-fly test.
3) We compare the performance among our estimator and other existing estimators, using different types of simulated datasets with known entropy values. The experimental results show that, our estimator performs well for all different types of tested datasets, outperforming the other ones.

The rest of this paper is organized as follows. In Section 2, we introduce the definition of min-entropy, along with an overview of the 90B standard. In Section 3 , we expound and analyze the existing estimators. Section 4 presents our new framework and provides detailed descriptions, including the change detection technique and so on. In Section 5, we present a series of experiments comparing our estimator with other estimators. Finally, in Section 6, we conclude our paper.

## 2 Preliminaries

### 2.1 Min-Entropy

"Entropy" is the unit representing the size of information in communication, which can quantify the randomness of the output sequence [15]. Min-entropy is a conservative way to ensure the quality of random numbers in the worst case. The definition of min-entropy is as follows: we take the next output from an entropy source as a random variable $X$, which is an independent discrete random variable. If $X$ takes value from the set $A=\left\{x_{1}, x_{2}, \ldots, x_{k}\right\}$ with probability $\operatorname{Pr}\left\{X=x_{i}\right\}=p_{i}$ for $i=1, \ldots, k$, the min-entropy of the output is

$$
\begin{equation*}
\mathrm{H}_{\min }=\min _{1 \leq \mathrm{i} \leq \mathrm{k}}\left[-\log _{2}\left(\mathrm{p}_{\mathrm{i}}\right)\right]=-\log _{2}\left[\max _{1 \leq \mathrm{i} \leq \mathrm{k}}\left(\mathrm{p}_{\mathrm{i}}\right)\right] . \tag{1}
\end{equation*}
$$

If the min-entropy of $X$ is $H$, then the probability of any value that $X$ can take doesn't exceed $2^{-H}$. For a random variable with the possibility of $k$ distinct values, the maximum value that the min-entropy can reach is $\log _{2} k$, achieved when the variable follows a uniform probability distribution, i.e., $p_{1}=p_{2}=\ldots=$ $p_{k}=1 / k$.

### 2.2 NIST SP800-90B Standard

The 90B estimation suite is a widely-used standard for calculating statistical entropy [15]. It calculates global predictability and local predictability with an upper bound of $99 \%$ confidence, and chooses the maximum value between them
Ma, Y. et al.
to estimate min-entropy. The suite comprises ten distinct entropy estimators that will be discussed in Section 3.

Global Predictability: Global predictability is the proportion of all predicted data to be correctly predicted. For a given prediction method, let $p_{\text {global }}^{\prime}=$ $c / n$, where $c$ represents the number of correct predictions and $n$ denotes the number of predictions made. Then, to give a conservative calculation method, 90B calculates $p_{\text {global }}$ according to the following equation [7]:

$$
p_{\text {global }}= \begin{cases}1-0.01^{1 / n}, & p_{\text {global }}^{\prime}=0  \tag{2}\\ \min \left(1, p_{\text {global }}^{\prime}+2.576 \sqrt{\frac{p_{g l o b a l}^{\prime}\left(1-p_{\text {global }}^{\prime}\right)}{n-1}}\right), \text { otherwise }\end{cases}
$$

which is the upper bound of the $99 \%$ confidence interval on $p_{\text {global }}^{\prime}$, and it should meet the condition of De Moivre-Laplace Central Limit Theorem, that is: let $X_{1}, X_{2}, \ldots, X_{n}$ be i.i.d Bernoulli random variables with success probability $p \in(0,1)$ such that $n p \rightarrow \infty$, as $n \rightarrow \infty$. Denote $S_{n}: X_{1}+X_{2}+\ldots+X_{n}$ and

$$
Y_{n}^{*}=\frac{S_{n}-n p}{\sqrt{n p(1-p)}}
$$

Then, $\forall y \in R$, the theorem states that

$$
\begin{equation*}
\lim _{n \rightarrow \infty}\left[P\left(Y_{n}^{*} \leq y\right)\right]=\Phi(y)=\frac{1}{\sqrt{2 \pi}} \int_{-\infty}^{y} e^{-t^{2} / 2} d t \tag{3}
\end{equation*}
$$

Local Predictability: Local predictability is based on the longest run of correct predictions, which is valuable mainly when the source falls into a state of very predictable output for a short time [4]. Let $l$ be the number one larger than the longest run of correct predictions. Then local predictability is calculated as

$$
\begin{equation*}
0.99=\frac{1-p_{\text {local }} x}{(l+1-l x) q} \cdot \frac{1}{x^{n+1}} \tag{4}
\end{equation*}
$$

where $q=1-p_{\text {local }}, n$ represents the number of predictions, and $x$ is the real positive root of the equation $1-x+q p_{l o c a l}^{l} x^{l+1}=0$. Then by iterations and the binary search, we can solve the mentioned equation and calculate the local predictability.

## 3 Related Work

### 3.1 Statistical Entropy Evaluation

Statistical entropy evaluation is comprised of estimators based on statistic methods and deep learning algorithms. For the former, in 2018, the final NIST SP80090B test suite published, which is a typical representative of the statistical entropy estimations, which is based on min-entropy and specifies how to design and test entropy sources.It employs ten different estimators to calculate the minentropy [15]. While it performs well on stationary datasets, it falls short when dealing with time-varying datasets. Before conducting entropy estimation, the 90B standard carries out an initial IID (independent and identically distributed) test. If the dataset meets the IID requirement, the MostCommon Estimator is utilized. Otherwise, the suite employs ten different estimators and selects the minimum value among them. These ten estimators can be divided into two categories: statistic-based and prediction-based. On the one hand, statistic-based estimators treat the test sequence as a whole and employ statistical methods to analyze properties related to entropy sources. On the other hand, predictionbased estimators use a training set comprised of previously observed samples to predict the next sample. By comparing the predicted results with the actual samples, the success rate of prediction is determined, and entropy estimation is performed based on the probability of successful prediction. Prediction-based estimators have a better performance than the other estimators in this standard. A brief introduction of the 10 estimators is as follows.
-Most Common Value Estimator performs entropy estimation based on the frequency of the most commonly occurring sample values in the sequence.
-Collision Estimator performs entropy estimation based on the collision frequency of samples in the sequence.
-Markov Estimator assumes the sequence as a first-order Markov process for entropy estimation.
-Compression Estimator is an entropy estimator based on the Maurer's algorithm.
-T-Tuple Estimator calculates entropy based on the occurrences of some fixed length repeated tuples.
-LRS Estimator calculates entropy based on the occurrences of some longer repeated tuples.
-MultiMCW Prediction Estimator utilizes four sliding windows of different sizes to determine the most frequently occurring value for prediction. A scoreboard is employed to determine the appropriate sliding window to use.
-Lag Prediction Estimator selects a prediction period ranging from 1 to 128 and also employs a scoreboard to select the optimal period.
-MultiMMC Prediction Estimator begins by setting up a dictionary (a two-dimensional array) and a scoreboard (a one-dimensional array). The dictionary is responsible for counting the frequency of prefixes and suffixes, while the scoreboard keeps a record of accurate predictions. After counting, it calculates the min-entropy.
-LZ78Y Prediction Estimator creates a dictionary based on patterns observed in the sequences and uses it for prediction.

Table 1. 90B Estimators.

| Statistic-based | Prediction-based |
| :--- | :--- |
| MostCommon Value Estimator MultiMCW Prediction Estimator |  |
| Collision Estimator | Lag Prediction Estimator |
| Markov Estimator | MultiMMC Prediction Estimator |
| Compression Estimator | LZ78Y Prediction Estimator |
| T-Tuple Estimator |  |
| LRS Estimator |  |

For the latter, Yang et al. [20] were the first to apply neural networks to entropy source evaluation in 2018. In 2020, Lv et al. [12] conducted a comprehensive study on parameter settings for fully-connected neural networks (FNN) and recurrent neural networks (RNN), achieving accurate estimates of M-sequences with up to 20 stages. In 2019, Zhu et al. [21] combined change detection techniques with neural networks, partially resolving the issue of inaccurate prediction for time-varying sequences, and their model is named CDNN. Furthermore, in 2023, Zhang et al. [10] utilized TPA-LSTM to quantify the unpredictability of random numbers, and validated the effectiveness of pruning and quantized deep learning models in the field of random number security analysis. The above methods provide increasingly accurate estimation, but the speed needs to be improved.

In summary, the prediction-based estimators of SP800-90B can provide the same accurate estimation as the deep learning based estimators for stationary datasets and some time-varing datasets, and the former can consume less time.

### 3.2 On-the-fly Test Technologies

In terms of the on-the-fly test applied in cryptography, Santoro et al. [14] conducted the evaluation of the harmonic series on FPGA in the entropy test in 2009. Then, in 2012, Veljković et al. [16] proposed the online implementation for NIST SP800-22 and Yang et al. [18] improved it in 2015.

At the same time, Yang et al. [19] completed hardware implementations of 4 statistic-based estimators of NIST SP800-90B on FPGA after some simplifications, but it is only aimed at the estimators of the first draft 90B and its accuracy needs to be improved. In 2017, Grujić et al. [6] used the three prediction estimator of NIST SP800-90B to implement the on-the-fly test, but the results is not very accurate because there some mistakes in the second draft standard, and besides, the latest draft is also not suitable because the dictionaries updates laggardly. Then, in 2021, Kim et al. [9] proposed an online estimator that updates the min-entropy estimate as a new sample is received, which is based on the idea
of the compression estimate of NIST SP800-90B, and it is implement on software. However, it doesn't perform well in time-varying datasets, even in some stationary datasets. Therefore, new framework should be designed to improve it.

## 4 New Framework of the 90B's Prediction Estimator for On-the-fly Test

### 4.1 Design Goal and Principle

Our design goal and principle is to achieve on-the-fly test effectively, so we need to improve speed while ensuring accuracy. We have tested that the minimum of time consumption of estimators based on deep learning is 30 seconds for processing 1 Mbit of data which can't meet the requirement of on-the-fly test. By contrast, the raw 90B estimators only consume 0.15 seconds. Therefore, we design the new framework according to the 90B estimators.

Besides, we know that on-the-fly test requires as few estimators as possible to reduce the time consuming, and prediction estimators outperform the other ones [7]. Therefore, while ensuring accuracy, we choose the four prediction estimators included in 90B to modify for on-the-fly test. Last but not least, suitable change detection technology and calculation method of global predictability should be designed to improve the accuracy.

### 4.2 Framework of Our Estimator

We can observe that the predictors in SP800-90B all feature scoreboards or dictionaries, which serve as key components in the prediction process. However, the estimation accuracy of these predictors in handling time-varying sequences is compromised. This can be attributed to the fact that, even as the datasets change, the scoreboards and dictionaries retain information from the previous datasets. As a result, there is a lag in the response of the dictionaries and scoreboards to data changes during accumulation, leading to prediction errors when applied to new datasets. Therefore, it is imperative to make improvements in this regard.

We have made the following modifications to the aforementioned estimators for conducting on-the-fly test. The entire process is presented in Figure 1. In it, point is the change position, and $i$ is the serial number of the sample. For each estimator, we perform the simultaneous operations of reading in data and outputting results in a serial manner. In step one, considering that the dictionaries and scoreboards have not yet started accumulating data at startup, which may result in erroneous estimation, we exclude the first 4999 samples from undergoing entropy estimation. During this phase, only the dictionaries and scoreboards are accumulated. Then, in the second step, at the point when there are 5000 samples, we calculate the prediction probability as the initial value for the change detection process based on the accumulated dictionaries. Starting from the 5001st


Fig. 1. The flowchart of the new framework.
sample, we calculate the prediction probability for each subsequently read-in sample. This calculated probability serves as the basis for the change detection technology.

In step three, if the prediction suddenly deviates and exceeds the threshold, we output the calculated min-entropy, clear the dictionaries and scoreboards, and initiate a new round of entropy estimation. Otherwise, as shown in the fourth step, if there is no change appearing, for every $I$ samples input, entropy calculation is performed according to the formula in Section 4.4, and the results are outputted without clearing the dictionaries and scoreboards. Throughout this process, the minimum value among the four estimators is selected as the final output result. Here, $I$ refers to the interval between two outputs.

### 4.3 Change Detection Module

In Figure 1, we employ a sequential approach for the four estimators to carry out the accumulation of dictionaries and scoreboards. We then utilize change detection technology to identify changes in the datasets and promptly clear the dictionaries and scoreboards when such changes are detected. This is followed by initiating a new round of dictionaries accumulation and scoreboards counting.

The current change detection technology can be categorized into three types: error rate-based drift detection, data distribution-based drift detection, and multiple hypothesis test drift detection [11]. The latter two methods require more time and resource consumption as they involve additional feature extraction and comparison processing on the data. Consequently, they are not suitable for our on-the-fly test scenario. Error rate-based drift detection, specifically the widely used Drift Detection Method (DDM) [5], offers a viable approach. Its concept is as follows: when the sample dataset exhibits stable distribution, the error rate of the model gradually decreases with the input of data; when there is a change in the probability distribution, the error rate of the model increases. We can reference the DDM approach for our change detection modules, but some adjustments will be necessary in terms of specific details.

For our estimator, when the probability distribution changes, the error rate of our model has a possibility of both an increase and a decrease; when the sample dataset is stable, the error rate is in an almost stable and unchanging state. Besides, Gama et al. [5] set the confidence level for drift to $99 \%$, and the drift level is reached if $p_{i}+s_{i} \geq p_{\text {min }}+3 \times s_{\text {min }}$, where $p_{i}$ is the probability corresponding to the first i samples and $s_{i}$ is the standard deviation of the first i samples. $p_{\min }$ is the minimum probability of the previous samples and $s_{\text {min }}$ is the corresponding standard deviation. However, the inequality can hold only when the normal approximation of the binomial distribution holds. Therefore, we replace it with another formula for general which mentioned in Section 4.4, that is,

$$
\begin{equation*}
p_{i} \geq \frac{p_{\min }+\frac{z_{\alpha / 2}^{2}}{2\left(i_{\min }-1\right)}+z_{\alpha / 2} \sqrt{\frac{p_{\min }\left(1-p_{\min }\right)}{i_{\min }-1}+\frac{z_{\alpha / 2}^{2}}{4\left(i_{\min }-1\right)^{2}}}}{1+\frac{z_{\alpha / 2}}{i_{\min }-1}}, \tag{5}
\end{equation*}
$$

and for the lower bound of confidence interval, we use the formula

$$
\begin{equation*}
p_{i} \leq \frac{p_{\max }+\frac{z_{\alpha / 2}^{2}}{2\left(i_{\max }-1\right)}-z_{\alpha / 2} \sqrt{\frac{p_{\max }\left(1-p_{\max }\right)}{i_{\max }-1}+\frac{z_{\alpha / 2}^{2}}{4\left(i_{\max }-1\right)^{2}}}}{1+\frac{z_{\alpha / 2}^{2}}{i_{\max }-1}}, \tag{6}
\end{equation*}
$$

where $p_{\max }$ is the maximum probability of the previous samples and $z_{\alpha / 2}$ is 2.576 when the confidence level is $99 \%$. This can apply to all situations.

Then, during the estimation, we use the prediction probabilities of four estimators for simultaneous change detection. As long as a probability that exceeds the confidence interval appears, it is determined that a change has occurred and immediately clear the dictionaries and scoreboards. In theory, our method is similar to the hyperparameter update of deep learning algorithms, but more timely than it.

### 4.4 Optimization of Global Predictability for Small Sample Datasets and Extreme Probability

According to Section 2.2, the calculation method of global predictability confidence interval is divided into two situations. When $p_{\text {global }}^{\prime}$ is zero, it uses ClopperPearson Exact Method [3]. In other cases, use normal distribution to approximate binomial distribution and calculate the confidence interval.

However, the above method only contains situation when $n p>5$ and $n(1-$ $p)>5$, or $p=0$ or $p=1$, where $n$ is the sample size and $p$ is the probability. Therefore, we should consider the case that $n p \leq 5$ or $n(1-p) \leq 5$ to make the perfect.

In our proposed online estimator, the dataset was truncated according to the sample distribution and parameter changes due to the use of change detection technology. Besides, in the process of entropy estimation, the probability may approach to 0 or 1 . The two factors may make us encounter the case mentioned above, i.e., $n p \leq 5$ or $n(1-p) \leq 5$. In this case, the confidence interval calculation of the global prediction probability in 90B standard is no longer valid because it does not meet the condition that the binomial distribution is approximated to normal distribution, that is, the central limit theorem mentioned in Section 2.2. Therefore, we need to use a new method to calculate it.

Poisson approximations can do for the above issue to some extent, but it doesn't provide the method of calculating the confidence interval [4]. Tdistribution can also handle some small sample issues, but still can't solve the above problem completely [2]. Therefore, we use "Plus Four Confidence Intervals" to handle the small sample issue here. This is proposed by Edwin Bidwell Wilson in 1927, which is an asymmetric interval [17]. It can be used for any probability value between 0 and 1 in the case of the small sample securely. It is obtained by solve the equation of $p: p=\hat{p} \pm z_{\alpha / 2} \sqrt{\frac{p(1-p)}{n}} . \hat{p}$ is the correct prediction proportion of the sample, and $n$ is the sample size. $z_{\alpha / 2}$ is the confidence coefficient, and it equals to 2.576 when the confidence interval is $99 \%$. The result is

$$
\begin{equation*}
p=\frac{\hat{p}+\frac{z_{\alpha / 2}^{2}}{2 n} \pm z_{\alpha / 2} \sqrt{\frac{\hat{p}(1-\hat{p})}{n}+\frac{z_{\alpha / 2}^{2}}{4 n^{2}}}}{1+\frac{z_{\alpha / 2}^{2}}{n}} . \tag{7}
\end{equation*}
$$

Then, the upper bound of confidence interval of global prediction under the new method is

$$
\begin{equation*}
p_{\text {global }}=\frac{p_{g l o b a l}^{\prime}+\frac{z_{\alpha / 2}^{2}}{2(n-1)}+z_{\alpha / 2} \sqrt{\frac{p_{g l o b a l}^{\prime}\left(1-p_{g l o b a l}^{\prime}\right)}{n-1}+\frac{z_{\alpha / 2}^{2}}{4(n-1)^{2}}}}{1+\frac{z_{\alpha / 2}^{2}}{n-1}} \tag{8}
\end{equation*}
$$

From the result, according to knowledge of the infinitesimal of higher order of the limit theory, we can see that when $n \rightarrow \infty$, the equation is approximate to

$$
\begin{equation*}
p_{\text {global }}=p_{\text {global }}^{\prime}+z_{\alpha / 2} \sqrt{\frac{p_{\text {global }}^{\prime}\left(1-p_{\text {global }}^{\prime}\right)}{n-1}} \tag{9}
\end{equation*}
$$

This means that the formula is also applicable to the case of large sample datasets. Besides, when $p_{g l o b a l}^{\prime}=0$, the result is greater than 0 , which indicates that it can also handle the situation of endpoint values.

Last but not least, we retain the original local prediction during the process of estimation because it is valid regardless of the sample size and extreme probability. Then, we choose the maximum of the global and local prediction to calculate the min-entropy as the final result.

### 4.5 Setting of Key Parameters

In this section, we discuss the setting of the parameters. We choose an initial accumulation size of 5000 samples for dictionaries and scoreboards due to the fact that the largest sliding window of the MultiMCW prediction estimator is 4095. If the accumulation size is smaller than 4095, the largest sliding window cannot accumulate dictionaries and scoreboards for the initial samples. This setting is a conservative approach, and it is suitable for other prediction estimators in 90B.

When determining the size of the interval $I$ in Figure 1, we take into account it both from theoretical and experimental perspectives. On the one hand, as mentioned earlier, the MultiMCW estimator's largest sliding window has a size of 4095 . Thus, the interval $I$ should be greater than this value, and we also prove it through the experiment. On the other hand, we conducted an experiment to determine the upper bound. We set the intervals as $2^{k}$, and $k$ takes from 1 to 17 , and evaluated sequences that followed IID and non-IID distributions separately. Because the dataset within each segment is stationary after segmentation under change detection technology, we needn't use the time-varying sequence here. For the IID dataset, we select a typical dataset generated by the Oscillatorbased model [1]. For the non-IID dataset, we choose one that followed a Markov


Fig. 2. Accuracy under Different Intervals.


Fig. 3. Throughput Rate under Different Intervals.
model. The accuracy and throughput rate under different intervals are depicted in Figure 2 and Figure 3.

In the results, we use the line chart to depict the accuracy and throughput rate. We see that the accuracy improves as the interval size increases, and when the interval exceeds $2^{12}$, the accuracy starts to fluctuate around $90 \%$. Besides, with larger intervals, the throughput rate grows faster, and when the interval reaches $2^{12}$, the throughput rate gradually becomes stable. However, processing too much data at once may consume a significant amount of memory and lead to latency. Therefore, to ensure accuracy and throughput rate, we set the interval range from $2^{12}$ to $2^{17}$. For the sake of convenience in displaying the results throughout the rest of this paper, we set a fixed interval of 50000 .

## 5 Experiment Results and Analysis

### 5.1 Experiment Setup

Our estimator is implemented in C/C++ language, and we show the results of the other estimators for comparison. In this section, all experiments are con-
ducted on a Windows 11 system with an Intel 11th Gen $\operatorname{Intel}(\mathrm{R})$ Core(TM) i7-1195G7 CPU and 16GB of memory.

During the experiment, we present the results in two ways. Firstly, for the offline estimators such as the estimators in 90B and others based on deep learning algorithms [10, 12, 21], we only compare their final offline estimation results with the endpoint result of our proposed estimator. We then display the error rate in the figure. This is because their methods are exclusively used in offline scenarios, and it would be unreasonable to choose intermediate output results for the final comparison. The error rate is calculated by the following formula:

$$
\begin{equation*}
\text { ErrorRate }=\frac{\left|H_{\text {test }}-H_{\text {correct }}\right|}{H_{\text {correct }}} \times 100 \%, \tag{10}
\end{equation*}
$$

where $H_{\text {correct }}$ is the theoretical min-entropy, and $H_{\text {test }}$ is the results of the estimators.

Secondly, for the online estimators, including our proposed one and the online estimator based on collision entropy proposed by Kim [9], we plot their estimations in the figures. We do not present the values of the 90B estimators implemented on FPGA because they utilize outdated estimators of the old version of 90B standard, which have some mistakes $[6,19]$.

### 5.2 Simulated Datasets for Experiments

The datasets used in our experiment can be divided into two categories: stationary datasets and time-varying datasets. The stationary datasets comprise various distribution families, including discrete uniform distribution, discrete near-uniform distribution, and normal distribution rounded to integers. More details are provided below.

- Discrete Uniform Distribution: The samples are subject to the discrete uniform distribution and are equally-likely. They come from an IID source.
- Discrete Near-uniform Distribution: The samples are subject to the discrete near-uniform distribution with one higher probability than the rest. They come from an IID source.
- Normal Distribution Rounded To Integers: The samples are subject to normal distribution and are rounded to integer values. They come from an IID source.

The time-varying datasets consist of two common situations: mutation (i.e., sudden change) and gradient (i.e., gradual change). To represent mutation, we utilize a dataset that undergoes near-uniform distribution with 9 mutations. For the gradient scenario, we employ a Markov model that exhibits a gradient following a linear function curve. The specific details are outlined below.

- Discrete Near-uniform Distribution with Mutation: The samples are divided into ten parts and each subject to the discrete near-uniform distribu-
tion with different parameter values, i.e., the higher probability. Table 2 shows the changes.
- Markov Model with Gradient: The samples are subject to a firstorder Markov process of $\{0,1\}$, and its transfer matrix is $\left(\begin{array}{cc}1-p & p \\ p & 1-p\end{array}\right)$, where $p$ changes along a linear function curve:

$$
p(i)= \begin{cases}0.1+0.0000004 i & , 0 \leq i<500000  \tag{11}\\ 0.3 & , 500000 \leq i<1000000\end{cases}
$$

where $i$ is the serial number of the sample.

Table 2. Discrete Near-uniform Distribution with Mutation.

| Serial Number of the Sample | Higher probability |
| :--- | :--- |
| $[1,80000]$ | 0.5 |
| $[80001,230000]$ | 0.8 |
| $[230001,330000]$ | 0.6 |
| $[330001,380000]$ | 0.85 |
| $[380001,400000]$ | 0.7 |
| $[400001,600000]$ | 0.9 |
| $[600001,900000]$ | 0.55 |
| $[900001,1200000]$ | 0.75 |
| $[1200001,1350000]$ | 0.95 |
| $[1350001,1500000]$ | 0.65 |

In terms of the dataset size, our proposed online estimator only requires a minimum of $2^{12}$ samples. However, other offline estimators, as per the 90 B standard, necessitate no less than one million samples. To facilitate comparison, we utilize a dataset that follows a discrete near-uniform distribution with 1.5 million samples for the mutation scenario, and other datasets with 1 million samples for the remaining scenarios.

### 5.3 Experimental Results

In this subsection, we present the results and then analyze them.

## 1) Offline estimation results

In figure 4, we use a column chart to represent the comparison of the error rate of ours and other estimators under different sequences. We find that our online estimator can give better results than raw 90 B estimators and other estimators that use deep learning algorithms, especially in time-varying sequences.
2) On-the-fly test results


Fig. 4. Comparison of error rate of estimators.

In the figures, we denote the correct values with red line, and use green "+" dots representing the results of the online estimator based on collision entropy. Ours is shown by blue " x " dots.

Figure 5(c) shows the estimated results of the simulation dataset from the independent normal distribution entropy source. We observed that the results provided by our estimator and the online estimator based on collision entropy are both close to the correct entropy.

Figure 5(a) and Figure 5(b) shows that the online estimator based on collision entropy always provide severely underestimation results on the datasets subject to discrete uniform distribution and discrete near uniform distribution because its algorithm is too simple to mine out the features of the datasets. By contrast, ours provides almost accurate estimations.

For the time-varying datasets, the online estimator based on collision entropy completely deviates from the theoretical min-entropy in Figure 6(a) and Figure 6(b). The estimations of ours can approach the theoretical correct values at most points due to the timely clearing of the dictionaries, although there are some deviations at the inflection points. This is caused by the delay in the change detection technology, and the delay is quite small, which is less than 1000 samples. It proves the effectiveness of our change detection technology.

### 5.4 Performance Evaluation

In this section, we discuss the performance of our proposed estimator. Firstly, in above results, our estimator can give more accurate estimation than other estimators.

Secondly, in terms of time consumption, as is shown in Table 3, the throughput rate of our estimators is stationary under different datasets, which is about $8.85 \mathrm{Mbit} / \mathrm{s}$. Therefore, for on-the-fly test, our estimator is suitable for random


(c) Comparison of min-entropy estimators for normal distribution source.

Fig. 5. Comparison of min-entropy for stationary sequences.
number generators with throughput rates less than or equal to $8.7 \mathrm{Mbit} / \mathrm{s}$ in terms of conservative estimation, whether software or hardware random number generators.

Besides, in the real world, entropy is an issue on low-power devices. Our estimator consumes 300 Mbit of memory for processing 1 Mbit of data, which is the same as the raw 90B standard.

## 6 Conclusion

In this paper, we design a new estimator based on the 90B prediction estimators for on-the-fly test. This design enhances both speed and accuracy. By employing change detection technology in our proposed new framework, we have achieved excellent performance. Additionally, to address situations involving small sample datasets or extreme probability, we utilize the "Plus Four Confidence Intervals" method to calculate the global predictability. Our estimator achieves a throughput rate exceeding 8.7 Mbit/s, meeting the on-the-fly test requirements of many RNGs. It currently stands as the most accurate technology for evaluating min-entropy. Looking ahead, our future plans involve further improving

(a) Comparison of min-entropy estima- (b) Comparison of min-entropy estimators for near uniform mutation sources. tors for Markov model gradient source.

Fig. 6. Comparison of min-entropy for time-varying sequences.
Table 3. Throughput rate under different sequences.

| Data type | Throughput rate (Mbit/s) |
| :--- | :--- |
| Uniform | 8.92 |
| Near-uniform | 8.90 |
| Normal distribution | 8.85 |
| Mutation | 8.76 |
| Gradient | 8.82 |

speed through hardware enhancements and parallel computing, aiming to ensure compatibility with a broader range of entropy sources.

Acknowledgements This work was supported by the National Natural Science Foundation of China under Grant 62272457.

## References

1. Baudet, M., Lubicz, D., Micolod, J., Tassiaux, A.: On the security of oscillatorbased random number generators. Journal of cryptology 24(2), 398-425 (2011)
2. Box, J.F.: Gosset, fisher, and the t distribution. The American Statistician 35(2), 61-66 (1981)
3. Clopper, C.J., Pearson, E.S.: The use of confidence or fiducial limits illustrated in the case of the binomial. Biometrika 26(4), 404-413 (1934)
4. Feller, W.: An introduction to probability theory and its applications, Volume 2, vol. 81. John Wiley \& Sons (1991)
5. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: Advances in Artificial Intelligence-SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings 17. pp. 286-295. Springer (2004)
6. Grujić, M., Rožić, V., Yang, B., Verbauwhede, I.: Lightweight prediction-based tests for on-line min-entropy estimation. IEEE Embedded Systems Letters 9(2), 45-48 (2017)
7. Kelsey, J., McKay, K.A., Sönmez Turan, M.: Predictive models for min-entropy estimation. In: Cryptographic Hardware and Embedded Systems-CHES 2015: 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings 17. pp. 373-392. Springer (2015)
8. Killmann, W., Schindler, W.: A design for a physical rng with robust entropy estimators. In: Cryptographic Hardware and Embedded Systems-CHES 2008: 10th International Workshop, Washington, DC, USA, August 10-13, 2008. Proceedings 10. pp. 146-163. Springer (2008)
9. Kim, Y., Guyot, C., Kim, Y.S.: On the efficient estimation of min-entropy. IEEE Transactions on Information Forensics and Security 16, 3013-3025 (2021)
10. Li, H., Zhang, J., Li, Z., Liu, J., Wang, Y.: Improvement of min-entropy evaluation based on pruning and quantized deep neural network. IEEE Transactions on Information Forensics and Security 18, 1410-1420 (2023)
11. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: A review. IEEE transactions on knowledge and data engineering 31(12), 2346-2363 (2018)
12. Lv, N., Chen, T., Zhu, S., Yang, J., Ma, Y., Jing, J., Lin, J.: High-efficiency minentropy estimation based on neural network for random number generators. Security and Communication Networks 2020, 1-18 (2020)
13. Ruk, A., et al.: A statistical test suite for the validation of random number generators and pseudo-random number generators for cryptographic applications. NIST, http://csrc. nist. gov/rng/rng2. html (2001)
14. Santoro, R., Tisserand, A., Sentieys, O., Roy, S.: Arithmetic operators for on-thefly evaluation of trngs. In: Mathematics for Signal and Information Processing. vol. 7444, pp. 253-264. SPIE (2009)
15. Turan, M.S., Barker, E., Kelsey, J., McKay, K.A., Baish, M.L., Boyle, M., et al.: Recommendation for the entropy sources used for random bit generation. NIST Special Publication 800(90B), 102 (2018)
16. Veljković, F., Rožić, V., Verbauwhede, I.: Low-cost implementations of on-the-fly tests for random number generators. In: 2012 Design, Automation \& Test in Europe Conference \& Exhibition (DATE). pp. 959-964. IEEE (2012)
17. Wilson, E.B.: Probable inference, the law of succession, and statistical inference. Journal of the American Statistical Association 22(158), 209-212 (1927)
18. Yang, B., Rožić, V., Mentens, N., Dehaene, W., Verbauwhede, I.: Embedded hw/sw platform for on-the-fly testing of true random number generators. In: 2015 Design, Automation \& Test in Europe Conference \& Exhibition (DATE). pp. 345-350. IEEE (2015)
19. Yang, B., Rožić, V., Mentens, N., Verbauwhede, I.: On-the-fly tests for non-ideal true random number generators. In: 2015 IEEE International Symposium on Circuits and Systems (ISCAS). pp. 2017-2020. IEEE (2015)
20. Yang, J., Zhu, S., Chen, T., Ma, Y., Lv, N., Lin, J.: Neural network based minentropy estimation for random number generators. In: International Conference on Security and Privacy in Communication Systems. pp. 231-250. Springer (2018)
21. Zhu, S., Ma, Y., Li, X., Yang, J., Lin, J., Jing, J.: On the analysis and improvement of min-entropy estimation on time-varying data. IEEE Transactions on Information Forensics and Security 15, 1696-1708 (2019)

# Leakage-Resilient Attribute-based Encryption with Attribute-hiding 

Yijian Zhang ${ }^{1}$, Yunhao Ling ${ }^{1}$, Jie Chen ${ }^{1,2(\boxtimes)}$, and Luping Wang ${ }^{3,4}$<br>${ }^{1}$ Software Engineering Institute, East China Normal University, Shanghai, China<br>${ }^{2}$ Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, China s080001@e.ntu.edu.sg<br>${ }^{3}$ School of Electronic and Information Engineering, Suzhou University of Science and Technology, Jiangsu, China<br>${ }^{4}$ Jiangsu Key Laboratory for Elevator Intelligent Safety, Jiangsu, China


#### Abstract

In this work, we present two generic frameworks for leakageresilient attribute-based encryption (ABE), which is an improved version of ABE that can be proven secure even when part of the secret key is leaked. Our frameworks rely on the standard assumption ( $k$-Lin) over prime-order groups. The first framework is designed for leakage-resilient ABE with attribute-hiding in the bounded leakage model. Prior to this work, no one had yet derived a generic leakage-resilient ABE framework with attribute-hiding. The second framework provides a generic method to construct leakage-resilient ABE in the continual leakage model. It is compatible with Zhang et al.'s work [DCC 2018] but more generic. Concretely, Zhang et al.'s framework cannot act on some specific ABE schemes while ours manages to do that. Technically, our frameworks are built on the predicate encoding of Chen et al.'s [EUROCRYPT 2015] combined with a method of adding redundancy. At last, several instantiations are derived from our frameworks, which cover the cases of zero inner-product predicate and non-zero inner-product predicate.


Keywords: Leakage-resilient • Attribute-based encryption • Attributehiding Predicate encoding.

## 1 Introduction

Attribute-based encryption (ABE) [18] is a primitive that can provide the confidentiality of data and fine-grained access control simultaneously. In ABE, a ciphertext $\mathrm{ct}_{\mathbf{x}}$ for a message $m$ is associated with an attribute $\mathbf{x} \in \mathcal{X}$, and a secret key $\mathrm{sk}_{\mathbf{y}}$ is associated with a policy $\mathbf{y} \in \mathcal{Y}$. Given a predicate $\mathrm{P}: \mathcal{X} \times \mathcal{Y} \rightarrow\{0,1\}$, $\mathrm{ct}_{\mathbf{x}}$ can be decrypted by sk $\mathrm{k}_{\mathbf{y}}$ if and only if $\mathrm{P}(\mathbf{x}, \mathbf{y})=1$.

The basic security requirement for ABE is payload-hiding. Roughly speaking, an adversary holding the secret key such that $\mathrm{P}(\mathbf{x}, \mathbf{y})=0$ cannot deduce any information about $m$ from the given ciphertext, and besides, this should be guaranteed even the adversary has more than one such secret key. In some scenarios, the attribute $\mathbf{x}$ may contain user privacy. For example, in the cloud
storage [11], the attribute $\mathbf{x}$ contains identity or address, which may be unsuitable to be exposed. Attribute-hiding [13] is an additional security requirement, and it concerns the privacy of attribute $\mathbf{x}$. Informally, attribute-hiding says that no information about attribute $\mathbf{x}$ can be disclosed to the adversary.

Recently, due to the emergence of side-channel attacks [1,9,12] which, through various physical methods, can recover part of the secret key, the leakage-resilient cryptography [8] is hence proposed. It is required that a leakage-resilient scheme should be provably secure in the leakage-resilient model. In this paper, we are interested in two prominent leakage-resilient models, namely, bounded leakage model (BLM) [2] and continual leakage model (CLM) [4]. Both of them assume that an adversary obtains leaked information about the secret key sk via a polynomial-time computable leakage function $f:\{0,1\}^{|s k|} \rightarrow\{0,1\}^{L}$ where $\mid$ sk| is the bit length of sk. In the BLM (resp. CLM), the adversary has access to at most $L<\mid$ sk| bits leakage on the secret key over the whole lifetime (resp. any time period) of the system. It is necessary to update sk periodically in the CLM. Typically, the security of CLM is stronger than BLM [10].

Up to now, various leakage-resilient frameworks have been proposed, while very few of them concentrate on leakage-resilient ABE. There are several generic leakage-resilient frameworks that can convert plain ABE schemes to leakageresilient ones in the BLM/CLM. The first one is introduced by Yu et al. [20]. Their generic leakage-resilient framework is able to convert the ABE schemes based on pair encoding [3] to leakage-resilient ones. However, their generic leakageresilient framework cannot provide attribute-hiding feature. Besides, for several concrete constructions, their security must rely on the non-standard computational assumptions, namely, q-type assumptions. Afterward, Zhang et al. [23] proposed a generic leakage-resilient ABE framework from hash proof system, while it also ignores attribute-hiding feature. Another independent work was proposed by Zhang et al. [22]. Their generic leakage-resilient framework is able to convert most ABE schemes based on predicate encoding [19] to leakageresilient ones. However, their generic leakage-resilient framework cannot guarantee attribute-hiding as well, and besides, cannot act on several specific ABE schemes based on predicate encoding, for example the compact-key ABE for inner-product predicate in [5], to leakage-resilient ones.

In this paper, we will follow the works of Chen et al. [5] and Zhang et al. [22], aimed at presenting two generic leakage-resilient frameworks. The first one can provide the attribute-hiding feature. The second one can convert more ABE schemes to leakage-resilient ones.

### 1.1 Contributions

In this work, we present two generic frameworks for the design of leakage-resilient ABE. Our contributions can be summarized as follows:

## - Leakage-resilient ABE with attribute-hiding in the BLM.

We introduce a new encoding called attribute-hiding-leakage-resilient. Based on the attribute-hiding techniques of CGW15[5] and this new encoding, we
present a generic leakage-resilient ABE construction with attribute-hiding, which is provably secure under the $k$-Lin assumption in the BLM.

- Leakage-resilient ABE in the CLM.

We introduce different redundancy into the secret key and the master key to ensure the security against continual leakage and add a linear map to ensure the generation and update of secret keys. Thus, we present a more generic leakage-resilient ABE in the CLM compared with ZCG+18.

A comparison between our frameworks and previous works is shown in Table 1. Note that, although our second framework in Section 4 has the same properties as $Z C G+18$, it can act on some specific schemes while $Z C G+18$ cannot do that.

Table 1: Comparison between previous works and ours. "Prime" denotes primeorder groups. "SD" means subgroup assumptions over composite-order groups.

| Reference | Leakage model | Attribute-hiding | Prime | Generality | Assumption |
| :---: | :---: | :---: | :---: | :---: | :---: |
| YAX+16[20] | CLM | $\boldsymbol{x}$ | $\boldsymbol{x}$ | $\perp$ | SD, q-type |
| ZZM17[23] | BLM | $\boldsymbol{x}$ | $\boldsymbol{x}$ | $\perp$ | SD |
| ZCG+18[22] | CLM | $\boldsymbol{x}$ | $\checkmark$ | weak | $k$-Lin |
| Ours(Section 3) | BLM | $\checkmark$ | $\checkmark$ | $\perp$ | $k$-Lin |
| Ours(Section 4) | CLM | $\boldsymbol{x}$ | $\checkmark$ | strong | $k$-Lin |

### 1.2 Technical Overview

Let ( $p, G_{1}, G_{2}, G_{T}, g_{1}, g_{2}, e$ ) denote an asymmetric bilinear group of prime-order $p$ with pairing $e: G_{1} \times G_{2} \rightarrow G_{T}$. We use mpk, mk to denote the master public key and the master key in ABE , respectively. Let $L \in \mathbb{N}$ be a leakage parameter.

Leakage-resilient ABE with attribute-hiding in the BLM. Based on the ABE with attribute-hiding in CGW15, we propose a generic leakage-resilient ABE construction that possesses attribute-hiding feature even when the secret key can be leaked to the adversary. An overview of our construction is presented as follows ${ }^{5}$ :

$$
\begin{array}{ll}
\mathrm{mpk}: g_{1}, g_{2}, g_{1}^{\mathbf{w}}, e\left(g_{1}, g_{2}\right)^{\alpha}, & \mathrm{mk}: \alpha, \mathbf{w} \\
\mathrm{sk}_{\mathbf{y}}: \mathbf{z}, g_{2}^{r}, g_{2}^{\mathrm{rkE}(\mathbf{y}, \mathbf{z}, \alpha)+r \cdot \mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{w})}, & \mathrm{ct}_{\mathbf{x}}: g_{1}^{s}, g_{1}^{s \cdot \mathbf{s E}(\mathbf{x}, \mathbf{w})}, m \cdot e\left(g_{1}, g_{2}\right)^{\alpha s} \tag{1}
\end{array}
$$

[^12]where $\mathbf{w} \in \mathcal{W}$ is a set of secret values; $\alpha, r, s \leftarrow \mathbb{Z}_{p} ; \mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y} ; \mathrm{rkE}, \mathrm{rE}, \mathrm{sE}$ are linear encoding algorithms; $\mathbf{z} \in \mathcal{Z}$ and $\mathbf{u}$ are "redundant" information. To achieve attribute-hiding security in the BLM, we require that

- (attribute-hiding.) For all $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ such that $\mathrm{P}(\mathbf{x}, \mathbf{y})=0$ and all $\mathbf{z} \in \mathcal{Z}$, the distributions $\{\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathrm{sE}(\mathbf{x}, \mathbf{w}), \mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{w})\}$ and $\{\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{r}\}$ are statistically indistinguishable where the randomness is taken over $\mathbf{w} \leftarrow \mathcal{W}$ and $\mathbf{r} \leftarrow \mathbb{Z}_{p}^{|\operatorname{sE}(\cdot)|+|\mathrm{rE}(\cdot)|}$.

The above requirement, namely attribute-hiding encoding, ensures the attributehiding feature. It manages to randomize $\mathbf{x}$ in $\mathrm{sE}(\mathbf{x}, \mathbf{w})$ even after the adversary has got $\mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{w})$ on $\mathrm{sk}_{\mathbf{y}}$. However, this property only holds when $\mathrm{P}(\mathbf{x}, \mathbf{y})=0$ and would be broken by the adversary with leak ability, since he can use the leakage function $f$ to acquire the leakage (i.e., $f(\mathbf{z}, \mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{w}))$ ) on $\mathrm{sk}_{\mathbf{y}}$ such that $\mathrm{P}(\mathbf{x}, \mathbf{y})=1$. The "redundant" information in sk $\mathrm{y}_{\mathbf{y}}$ is designed to avoid this problem. Inspired by ZCG+18[22] and LRW11[14], we additionally require that

- (attribute-hiding-leakage-resilient.) For all $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ such that $\mathrm{P}(\mathbf{x}, \mathbf{y})=1$ and $\mathbf{z} \in \mathcal{Z}$, the distributions $\{\mathbf{x}, \mathbf{y}, \mathrm{sE}(\mathbf{x}, \mathbf{w}), f(\mathbf{z}, \mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{w}))\}$ and $\{\mathbf{x}, \mathbf{y}, \mathbf{r}\}$ are identical, where $\mathbf{w} \leftarrow \mathcal{W}$ and $\mathbf{r} \leftarrow \mathbb{Z}_{p}^{|\operatorname{sE}(\cdot)|+|f(\cdot)|}$.

This encoding guarantees that with the leakage of sk $\mathbf{y}_{\mathbf{y}}$ such that $\mathrm{P}(\mathbf{x}, \mathbf{y})=1$, the adversary still cannot reveal the attribute $\mathbf{x}$ under $\mathrm{s} E(\mathbf{x}, \mathbf{w})$ since it seems to be sampled uniformly. Thus, the Equation (1) achieves attribute-hiding in the BLM.

Leakage-resilient ABE in the CLM. For the second leakage-resilient ABE framework, we consider the CLM which is stronger than BLM. Although ZCG +18 has proposed a leakage-resilient ABE framework in the CLM, it is not general enough to act on some specific schemes, e.g., compact-key ABE schemes for zero inner-product and non-zero inner-product in CGW15. For these specific schemes, their master keys contain multiple secret values (e.g., $\alpha$ and $\mathbf{w}$ ), and the adversary can break the security trivially if one of these secret values is leaked. Our solution is to differentiate the redundant information of mk and the redundant information of $s k_{\mathbf{y}}$, which provides more possibilities to avoid the leakage on secret values. Thus, we present a new leakage-resilient ABE generic construction:

$$
\begin{array}{ll}
\mathrm{mpk}: g_{1}, g_{2}, g_{1}^{\mathbf{w}}, g_{2}^{\mathbf{w}}, e\left(g_{1}, g_{2}\right)^{\alpha}, & \mathrm{mk}: \mathbf{v}, g_{2}^{r}, g_{2}^{\mathrm{mkE}(\mathbf{v}, \alpha)+r \cdot \mathrm{mE}(\mathbf{v}, \mathbf{w})}, \\
\mathrm{sk}_{\mathbf{y}}: \mathbf{z}, g_{2}^{r}, g_{2}^{r \mathrm{rk}(\mathbf{y}, \mathbf{z}, \alpha)+r \cdot \cdot \mathrm{EE}(\mathbf{y}, \mathbf{z}, \mathbf{w})}, & \mathrm{ct}_{\mathbf{x}}: g_{1}^{s}, g_{1}^{s \cdot \mathbf{s E}(\mathbf{x}, \mathbf{w})}, m \cdot e\left(g_{1}, g_{2}\right)^{\alpha s} \tag{2}
\end{array}
$$

where $m k E, m E$ are encoding algorithms; $\mathbf{v} \in \mathcal{V}$ and $\mathbf{z} \in \mathcal{Z}$ serve as redundant information for $m k$ and $s k_{\mathbf{y}}$, respectively. Note that this construction is similar to the Equation (1), while it considers CLM (rather than BLM) and allows the leakage on sky and mk. Here, we require that

1) ( $\alpha$-privacy.) For all $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ such that $\mathrm{P}(\mathbf{x}, \mathbf{y})=0$, the distributions $\{\mathbf{x}, \mathbf{y}, \mathbf{z}, \alpha, \mathrm{sE}(\mathbf{x}, \mathbf{w}), \mathrm{rkE}(\mathbf{y}, \mathbf{z}, \alpha)+\mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{w})\}$ and $\{\mathbf{x}, \mathbf{y}, \mathbf{z}, \alpha, \mathrm{sE}(\mathbf{x}, \mathbf{w})$, $\mathrm{r} \mathrm{E}(\mathbf{y}, \mathbf{z}, \mathbf{w})\}$ are identical where the randomness is taken over $\mathbf{w} \leftarrow \mathcal{W}$.
2) ( $\alpha$-leakage-resilient.) For all $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ such that $\mathrm{P}(\mathbf{x}, \mathbf{y})=1$ and all $\alpha \in \mathbb{Z}_{p}, \mathbf{z} \in \mathcal{Z}$, the distributions $\{\mathbf{x}, \mathbf{y}, \alpha, \mathrm{sE}(\mathbf{x}, \mathbf{w}), f(\mathbf{z}, \mathrm{rkE}(\mathbf{y}, \mathbf{z}, \alpha)+$ $\mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{w}))\}$ and $\{\mathbf{x}, \mathbf{y}, \alpha, \mathrm{sE}(\mathbf{x}, \mathbf{w}), f(\mathbf{z}, \mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{w}))\}$ are identical where $\mathbf{w} \leftarrow$ $\mathcal{W}$ and $f$ is a leakage function.
In addition, the distributions $\{\mathbf{x}, \alpha, \mathrm{sE}(\mathbf{x}, \mathbf{w}), f(\mathbf{z}, \operatorname{mkE}(\mathbf{v}, \alpha)+\mathrm{mE}(\mathbf{v}, \mathbf{w}))\}$ and $\{\mathbf{x}, \alpha, \mathrm{sE}(\mathbf{x}, \mathbf{w}), f(\mathbf{v}, \mathrm{mE}(\mathbf{v}, \mathbf{w}))\}$ are identical.
3) (re-randomizable.) There exists a update algorithm for $s \mathrm{k}_{\mathrm{y}}$ and mk .
4) (delegable.) There exists an algorithm that takes as input mk and y and outputs a fresh secret key sky.
$\alpha$-privacy and $\alpha$-leakage-resilient are aimed at resisting continual leakage on $s \mathrm{k}_{\mathrm{y}}$ and mk . Since the total leakage bound of the adversary is unlimited in the CLM, re-randomizable and delegable are proposed to ensure the periodical update for $\mathrm{sk}_{\mathbf{y}}$ and mk . As a specific case, we let $\mathbf{w}:=\left(w_{1}, \ldots, w_{n}, \mathbf{u}\right) \in \mathbb{Z}_{p}^{n+L}, \mathbf{v}:=$ $\left(\mathbf{v}_{0}, \mathbf{v}_{1}, \ldots, \mathbf{v}_{n}\right) \in\left(\mathbb{Z}_{p}^{L}\right)^{n}$,

$$
\operatorname{mkE}(\mathbf{v}, \alpha) \stackrel{\text { def }}{=}(\alpha, 0, \ldots, 0), \mathrm{mE}(\mathbf{v}, \mathbf{w}) \stackrel{\text { def }}{=}\left(\mathbf{v}_{0}^{\top} \mathbf{u}, w_{1}+\mathbf{v}_{1}^{\top} \mathbf{u}, \ldots, w_{n}+\mathbf{v}_{n}^{\top} \mathbf{u}, \mathbf{u}\right)
$$

In the above equality, it is best for the adversary to get the leakage on $(\alpha+$ $\left.\mathbf{v}_{0}^{\top} \mathbf{u}_{0}, \mathbf{v}_{0}, \mathbf{u}\right)$ or $\left(w_{i}+\mathbf{v}_{i}^{\top} \mathbf{u}_{i}, \mathbf{v}_{i}, \mathbf{u}\right)$ if the adversary tries to leak $\alpha$ or $w_{i}$. Note that for any $i \neq j, \mathbf{v}_{i}^{\top} \mathbf{u}$ is statistically independent from $\mathbf{v}_{j}^{\top} \mathbf{u}$ due to the randomness of $\mathbf{v}$. Then based on the subspace lemma in LRW11, $\alpha$ or $w_{i}$ is hidden as long as the adversary gets a limited amount of leakage on mk during a time period. Thus, the randomness of $\mathbf{w}$ is preserved, then $\alpha$-privacy and $\alpha$-leakage-resilient are satisfied. Besides, re-randomizable holds since we have published $g_{2}^{\mathbf{w}}$ in mpk. As for delegable, we additionally require a linear map $S: \mathcal{Y} \times \mathcal{V} \rightarrow \mathcal{Z}$, which enables the redundant information $\mathbf{z}$ in $\mathrm{sk}_{\mathbf{y}}$ to be computed from $\mathbf{v}$ and $\mathbf{y}$. Thus, $\mathrm{sk}_{\mathbf{y}}$ can be generated from mk and $\mathbf{y}$ correctly. At last, we apply our second framework (in Section 4) to compact-key ABE schemes for zero inner-product and non-zero inner-product in CGW15, and hence obtain several leakage-resilient instantiations in Section 5.

### 1.3 Related Work

Other leakage-resilient models. Dziembowski et al. [6] defined the bounded retrieval model (BRM), placing rigorous performance requirements on the leakageresilient scheme. Dodis et al. [7] proposed the auxiliary input leakage model (ALM). It only requires that the leakage function $f$ is hard to invert. Besides, Yuen at al. [21] defined the continual auxiliary leakage model (CAL) that captures the benefits of both CLM and ALM.
Leakage-resilient ABE. Lewko et al. [14] proposed the first identity-based encryption (IBE) and ABE which are proved in the CLM. Zhang and Mu [24] constructed a leakage-resilient anonymous inner-product encryption (IPE) scheme over composite-order groups in the BLM. Nishimaki and Yamakawa [17] proposed several constructions of leakage-resilient public-key encryption and leakage-resilient IBE in the BRM, which reach nearly optimal leakage rates under
standard assumptions in the standard model. To deal with potential side-channel attacks in the distributed environment, Li et al. $[16,15]$ designed a key-policy ABE in the CAL and a hierarchical ABE in the CLM.

Organization. We recall the related definition and security models in §2. The first leakage-resilient ABE framework is presented in §3. The Second leakageresilient ABE framework is shown in $\S 4$. We present some instantiations in $\S 5$.

## 2 Preliminaries

Notations. For $n \in \mathbb{N},[n]$ denote the set $\{1,2, \ldots, n\}$. We use $s \leftarrow \mathcal{S}$ to denote that $s$ is picked randomly from set $\mathcal{S}$. By PPT, we denote a probabilistic polynomial-time algorithm. We use $\stackrel{\mathcal{c}}{\approx}$ and $\stackrel{\sim}{\approx}$ to denote two distributions being computationally and statistically indistinguishable, respectively.

### 2.1 The Definition of ABE

Given attribute universe $\mathcal{X}$, predicate universe $\mathcal{Y}$ and predicate $\mathrm{P}: \mathcal{X} \times \mathcal{Y} \rightarrow$ $\{0,1\}$, an ABE scheme consists of four algorithms (Setup, KeyGen, Enc, Dec):

- Setup $\left(1^{\lambda}\right) \rightarrow(m p k, m k)$. Take as input a security parameter $\lambda$. Then return the public parameters mpk and the master key mk.
- KeyGen $(m k, y) \rightarrow s k_{\mathbf{y}}$. Take as input $m k, \mathbf{y} \in \mathcal{Y}$, and return a secret key sky.
- Enc $(\mathrm{mpk}, \mathbf{x}, m) \rightarrow \mathrm{ct}_{\mathbf{x}}$. Take as input mpk , an attribute $\mathbf{x} \in \mathcal{X}$, and a message $m$. Return a ciphertext $\mathrm{ct}_{\mathrm{x}}$.
- $\operatorname{Dec}\left(m p k, \mathrm{sk}_{\mathbf{y}}, \mathrm{ct}_{\mathbf{x}}\right) \rightarrow m$ or $\perp$. Take as input $\mathrm{sk}_{\mathbf{y}}$ and $\mathrm{ct}_{\mathbf{x}}$. If $\mathrm{P}(\mathbf{x}, \mathbf{y})=1$, return message $m$; otherwise, return $\perp$.

Correctness. For all $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ such that $\mathrm{P}(\mathbf{x}, \mathbf{y})=1$ and all $m \in \mathcal{M}$, it holds that $\operatorname{Pr}\left[\operatorname{Dec}\left(m p k, s k_{\mathbf{y}}, \operatorname{Enc}(m p k, \mathbf{x}, m)\right)=m\right]=1$ where $(m p k, m k) \leftarrow$ $\operatorname{Setup}\left(1^{\lambda}, 1^{n}\right)$, sk $\mathbf{y}_{\mathbf{y}} \leftarrow \operatorname{KeyGen}(\mathrm{mk}, \mathbf{y})$.
Additional algorithm. If we take the presence of continual leakage into account, an extra algorithm should be provided:

- Update(mpk, sk $\mathbf{k}_{\mathbf{y}}$ ) : Take as input a secret key $\mathrm{sk}_{\mathbf{y}}$, and outputs a re-randomized key sk'

It is equivalent to generating a fresh secret key $\mathrm{sk}_{\mathbf{y}}^{\prime} \leftarrow \operatorname{KeyGen}(\mathrm{mk}, \mathbf{y})$. We stress that $m \mathrm{k}$ can be seen as a secret key $\mathrm{sk}_{\mathbf{y}}$ (where $\mathbf{y}$ is an empty string $\epsilon$ ) and algorithm Update also acts on mk.

### 2.2 Security Models

Here, we would define two leakage-resilient models, both of which are parameterized by security parameter $\lambda$ and leakage bounds $L_{\mathrm{mk}}=L_{\mathrm{mk}}(\lambda), L_{\mathrm{sk}}=L_{\mathrm{sk}}(\lambda)$.

Definition 1. We say that an ABE scheme is ( $L_{\mathrm{mk}}, L_{\mathrm{sk}}$ )-bounded-leakage secure and attribute-hiding if for all PPT adversaries $\mathcal{A}$, the advantage function
$\operatorname{Adv}_{\mathcal{A}}^{\mathrm{BLR-AH}}(\lambda):=\left|\operatorname{Pr}\left[b^{\prime}=b \left\lvert\, \begin{array}{l}(\mathrm{mpk}, \mathrm{mk}) \leftarrow \operatorname{Setup}\left(1^{\lambda}\right) \\ \left(\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, m^{(0)}, m^{(1)}\right) \leftarrow \mathcal{A}^{\mathrm{O}_{1}, \mathrm{O}_{2}, \mathrm{O}_{3}(\mathrm{mpk})} \\ b \leftarrow\{0,1\} ; \mathrm{ct}^{*} \leftarrow \operatorname{Enc}\left(\mathrm{mpk}, \mathbf{x}^{(b)}, m^{(b)}\right) \\ b^{\prime} \leftarrow \mathcal{A}^{\mathrm{O}_{1}, \mathrm{O}_{2}, \mathrm{O}_{3}\left(\mathrm{mpk}, \mathrm{ct}^{*}\right)}\end{array}\right.\right]-\frac{1}{2}\right|$. is negligible.

In the above definition, $\mathcal{A}$ has access to oracles $\mathrm{O}_{1}, \mathrm{O}_{2}, \mathrm{O}_{3}$. These oracles maintain sets $\mathcal{H}$ and $\mathcal{R}$ which store some tuples.

- $\mathrm{O}_{1}(h, \mathbf{y}): h$ is a handle to a tuple of $\mathcal{H}$ that must refer to a master key and $\mathbf{y}$ must be a vector in $\mathcal{Y}$. After receiving the input, this oracle finds the tuple $t$ with handle $h$ in $\mathcal{H}$ and answers $\mathcal{A}$ as follows:

1) If the vector part of $t$ is $\epsilon$, then let $t:=(h, \epsilon, \mathrm{mk}, l)$. It runs KeyGen algorithm to obtain a key $\mathbf{s k}_{\mathbf{y}}$ and adds the tuple $\left(H+1, \mathbf{y}, \mathbf{s k}_{\mathbf{y}}, 0\right)$ to $\mathcal{H}$. Then it updates $H \leftarrow H+1$;
2) Otherwise, it returns $\perp$ to $\mathcal{A}$.

- $\mathrm{O}_{2}(h, f): f$ is a polynomial-time computable function of constant output size. After receiving the input, it finds the tuple $t$ with handle $h$ in $\mathcal{H}$ and answers $\mathcal{A}$ as follows:

1) If $t$ is of the form $(h, \epsilon, \mathrm{mk}, l)$, it checks whether $l+|f(\mathrm{mk})| \leq L_{\mathrm{mk}}$. If $l+|f(\mathrm{mk})| \leq L_{\mathrm{mk}}$ holds, the challenger returns $f(\mathrm{mk})$ to $\mathcal{A}$ and updates $l \leftarrow l+|f(\mathrm{mk})|$. Otherwise, it returns $\perp$ to $\mathcal{A}$;
2) Else, $t$ is of the form $\left(h, \mathbf{y}, \mathrm{sk}_{\mathbf{y}}, l\right)$ and then it checks whether $l+\left|f\left(\mathrm{sk}_{\mathbf{y}}\right)\right| \leq$ $L_{\mathbf{s k}}$. If $l+\left|f\left(\mathrm{sk}_{\mathbf{y}}\right)\right| \leq L_{\mathrm{sk}}$ holds, the challenger returns $f\left(\mathrm{sk}_{\mathbf{y}}\right)$ to $\mathcal{A}$ and updates $l \leftarrow l+\left|f\left(\mathbf{s k}_{\mathbf{y}}\right)\right|$. Otherwise, it returns $\perp$.
$-\mathrm{O}_{3}(h)$ : It finds the tuple with handle $h$ in $\mathcal{H}$. If the vector part of the tuple is $\epsilon$, then it returns $\perp$ to $\mathcal{A}$. Otherwise, the tuple is of the form $\left(h, \mathbf{y}, \mathrm{sk}_{\mathbf{y}}, l\right)$. It returns $\mathrm{sk}_{\mathrm{y}}$ and then add y to $\mathcal{R}$.

Note that after $\mathcal{A}$ receives the challenge ciphertext ct*, only queries on sky such that $\mathrm{P}\left(\mathbf{x}^{(0)}, \mathbf{y}\right)=0$ and $\mathrm{P}\left(\mathbf{x}^{(1)}, \mathbf{y}\right)=0$ are allowed when $\mathcal{A}$ access to $\mathrm{O}_{2}, \mathrm{O}_{3}$.

Definition 2. We say that an ABE scheme is ( $L_{\mathrm{mk}}, L_{\mathrm{sk}}$ )-continual-leakage secure if for all PPT adversaries $\mathcal{A}$, the advantage function

$$
\operatorname{Adv}_{\mathcal{A}}^{\mathrm{CLR-PH}}(\lambda):=\left|\operatorname{Pr}\left[b^{\prime}=b \left\lvert\, \begin{array}{l}
(\mathrm{mpk}, \mathrm{mk}) \leftarrow \operatorname{Setup}\left(1^{\lambda}\right) \\
\left(\mathbf{x}, m^{(0)}, m^{(1)}\right) \leftarrow \mathcal{A}^{\mathrm{O}_{1}^{\prime}, \mathrm{O}_{2}^{\prime}, \mathrm{O}_{3}^{\prime}}(\mathrm{mpk}) \\
b \leftarrow\{0,1\} ; \mathrm{ct}^{*} \leftarrow \operatorname{Enc}\left(\mathrm{mpk}, \mathbf{x}, m^{(b)}\right) \\
b^{\prime} \leftarrow \mathcal{A}^{\mathrm{O}_{1}^{\prime}, \mathrm{O}_{2}^{\prime}, \mathrm{O}_{3}^{\prime}\left(\mathrm{mpk}, \mathrm{ct}^{*}\right)}
\end{array}\right.\right]-\frac{1}{2}\right| .
$$

is negligible.
Here, $\mathcal{A}$ has access to oracles $\mathrm{O}_{1}^{\prime}, \mathrm{O}_{2}^{\prime}, \mathrm{O}_{3}^{\prime}$. These oracles maintain sets $\mathcal{H}^{\prime}$ and $\mathcal{R}^{\prime}$.

- $\mathrm{O}_{1}^{\prime}(h, \mathbf{y})$ : This oracle is similar to $\mathrm{O}_{1}$ except that the input $\mathbf{y}$ can also be an empty string $\epsilon$. If $\mathcal{A}$ makes a query for $\mathbf{y}=\epsilon$, it will run Update algorithm to get a fresh master key $\mathrm{mk}^{\prime}$ and add the tuple $\left(H+1, \epsilon, \mathrm{mk}^{\prime}, 0\right)$ to the set $\mathcal{H}$.
$-\mathrm{O}_{2}^{\prime}(h, f)$ : This oracle is the same as $\mathrm{O}_{2}$.
$-\mathrm{O}_{3}^{\prime}(h)$ : This oracle is the same as $\mathrm{O}_{3}$.
Note that after $\mathcal{A}$ receives the challenge ciphertext $\mathrm{ct}^{*}$, only queries on $\mathrm{sk}_{\mathbf{y}}$ such that $\mathrm{P}(\mathbf{x}, \mathbf{y})=0$ are allowed when $\mathcal{A}$ access to $\mathrm{O}_{2}^{\prime}, \mathrm{O}_{3}^{\prime}$.


### 2.3 Assumption

Let $\mathcal{G}$ be a probabilistic polynomial-time algorithm that takes as input a security parameter $1^{\lambda}$ and outputs a group description $\mathbb{G}:=\left(p, G_{1}, G_{2}, G_{T}, g_{1}, g_{2}, e\right)$, where $p$ is a $\Theta(\lambda)$-bit prime and $G_{1}, G_{2}, G_{T}$ are cyclic groups of order $p . g_{1}$ and $g_{2}$ are generators of $G_{1}$ and $G_{2}$ respectively and $e: G_{1} \times G_{2} \rightarrow G_{T}$ is a computationally efficient and non-degenerate bilinear map. We let $g_{T}=e\left(g_{1}, g_{2}\right)$ be the generator of $G_{T}$.

For $s \in\{1,2, T\}$ and $a \in \mathbb{Z}_{p}$, we define $[a]_{s}=g_{s}^{a}$ as the implicit representation of $a$ in $G_{s}$. Similarly, for a matrix $\mathbf{A}$ over $\mathbb{Z}_{p}$, we define $[\mathbf{A}]_{s}=g_{s}^{\mathbf{A}}$, where exponentiations are carried out component-wise. Given $[\mathbf{A}]_{1}$ and $[\mathbf{B}]_{2}$, we define $e\left([\mathbf{A}]_{1},[\mathbf{B}]_{2}\right):=\left[\mathbf{A}^{\top} \mathbf{B}\right]_{T}$. Now we review the definition of $k$-Lin assumption.

Definition 3 ( $k$-Lin Assumption). Let $s \in\{1,2, T\}$. We say that the $k$-Lin assumption holds with respect to $\mathcal{G}$ on $G_{s}$ if for all PPT adversaries $\mathcal{A}$, the following advantage function is negligible in $\lambda$.

$$
\operatorname{Adv}_{\mathcal{A}}^{k-\operatorname{Lin}}(\lambda):=\left|\operatorname{Pr}\left[\mathcal{A}\left(\mathbb{G},[\mathbf{A}]_{s},[\mathbf{A t}]_{s}\right)=1\right]-\operatorname{Pr}\left[\mathcal{A}\left(\mathbb{G},[\mathbf{A}]_{s},[\mathbf{u}]_{s}\right)=1\right]\right|
$$

where $\mathbb{G} \leftarrow \mathcal{G}\left(1^{\lambda}\right), \mathbf{t} \leftarrow \mathbb{Z}_{p}^{k}, \mathbf{u} \leftarrow \mathbb{Z}_{p}^{k+1},\left(a_{1}, \ldots, a_{k}\right) \leftarrow \mathbb{Z}_{p}^{k}$, then

$$
\mathbf{A}:=\left(\begin{array}{cccc}
a_{1} & &  \tag{3}\\
& \ddots & \\
& & a_{k} \\
1 & \cdots & 1
\end{array}\right) \in \mathbb{Z}_{p}^{(k+1) \times k}
$$

Note that we can trivially set $\left(\mathbf{a}^{\perp}\right)^{\top}:=\left(a_{1}^{-1}, \ldots, a_{k}^{-1},-1\right)$ such that $\mathbf{A}^{\top} \mathbf{a}^{\perp}=\mathbf{0}$.

## 3 Leakage-resilient ABE with Attribute-hiding in the BLM

In this section, we will present the first leakage-resilient ABE framework along with the predicate encoding, generic construction and corresponding security analysis.

### 3.1 Leakage-resilient Predicate Encoding

A $\mathbb{Z}_{p}$-linear leakage-resilient predicate encoding with attribute-hiding for predicate $\mathrm{P}: \mathcal{X} \times \mathcal{Y} \rightarrow\{0,1\}$, which contains a set of deterministic algorithms ( $r k E, r E, s E, s D, r D$ ), satisfies the following properties:

- (linearity.) For all $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}, \operatorname{rkE}(\mathbf{y}, \mathbf{z}, \cdot), \mathrm{rE}(\mathbf{y}, \mathbf{z}, \cdot), \mathrm{sE}(\mathbf{x}, \cdot)$, $\mathrm{sD}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \cdot), \mathrm{rD}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \cdot)$ are $\mathbb{Z}_{p^{\prime}}$-linear functions. A $\mathbb{Z}_{p}$-linear function $F$ can be encoded as a matrix $\mathbf{T}=\left(t_{i, j}\right) \in \mathbb{Z}_{p}^{n \times m}$ such that $F:\left(w_{1}, \ldots, w_{n}\right) \longmapsto$ $\left(\sum_{i=1}^{n} t_{i, 1} w_{i}, \ldots, \sum_{i=1}^{n} t_{i, m} w_{i}\right)$.
- (restricted $\alpha$-reconstruction.) For all $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ such that $\mathrm{P}(\mathbf{x}, \mathbf{y})=$ 1 , all $\mathbf{w} \in \mathcal{W}, \mathbf{z} \in \mathcal{Z}$, it holds that $\mathrm{sD}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathrm{sE}(\mathbf{x}, \mathbf{w}))=\mathrm{rD}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{w}))$ and $\operatorname{rD}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \operatorname{rkE}(\mathbf{y}, \mathbf{z}, \alpha))=\alpha$.
- ( $x$-oblivious $\alpha$-reconstruction.) $\mathrm{sD}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \cdot), \mathrm{rD}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \cdot)$ are independent of $\mathbf{x}$. It is a basic requirement for achieving attribute-hiding.
- (attribute-hiding.) For all $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ such that $\mathrm{P}(\mathbf{x}, \mathbf{y})=0$ and all $\mathbf{z} \in \mathcal{Z}$, the distributions $\{\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathrm{sE}(\mathbf{x}, \mathbf{w}), \mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{w})\}$ and $\{\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{r}\}$ are identical, where $\mathbf{w} \leftarrow \mathcal{W}$ and $\mathbf{r} \leftarrow \mathbb{Z}_{p}^{|\operatorname{sE}(\cdot)|+|\mathrm{rE}(\cdot)|}$.
- (attribute-hiding-leakage-resilient.) In order to achieve leakage-resilience on $\mathbf{s k}_{\mathbf{y}}$, we require that for all $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ such that $\mathrm{P}(\mathbf{x}, \mathbf{y})=1$ and $\mathbf{z} \in \mathcal{Z}$, the distributions $\{\mathbf{x}, \mathbf{y}, \mathrm{sE}(\mathbf{x}, \mathbf{w}), f(\mathbf{z}, \mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{w}))\}$ and $\{\mathbf{x}, \mathbf{y}, \mathbf{r}\}$ are identical, where $\mathbf{w} \leftarrow \mathcal{W}$ and $\mathbf{r} \leftarrow \mathbb{Z}_{p}^{|\operatorname{sE}(\cdot)|+|f(\cdot)|}$.


### 3.2 Generic Construction

An overview of our generic construction has been present in Section (1). As mentioned in Section 1.2, a general approach [5] to transform schemes over composite-order groups into ones over prime-order groups can be applied to Equation (1). Concretely, we replace $g_{1}, g_{2}$ with $[\mathbf{A}]_{1},[\mathbf{B}]_{2}$, where $\left(\mathbf{A}, \mathbf{a}^{\perp}\right),\left(\mathbf{B}, \mathbf{b}^{\perp}\right)$ $\leftarrow \mathcal{D}_{k+1, k}$ and other variables are transformed as follows:

$$
\begin{aligned}
\alpha \mapsto & \mathbf{k} \in \mathbb{Z}_{p}^{k+1}, u, w_{i} \mapsto \mathbf{U}, \mathbf{W}_{i} \in \mathbb{Z}_{p}^{(k+1) \times(k+1)}, s \mapsto \mathbf{s} \in \mathbb{Z}_{p}^{k}, r \mapsto \mathbf{r} \in \mathbb{Z}_{p}^{k} \\
& g_{1}^{s} \mapsto[\mathbf{A s}]_{1}, g_{1}^{w_{i} s} \mapsto\left[\mathbf{W}_{i}^{\top} \mathbf{A s}\right]_{1}, g_{2}^{r} \mapsto[\mathbf{B r}]_{2}, g_{2}^{w_{i} r} \mapsto\left[\mathbf{W}_{i} \mathbf{B r}\right]_{2}
\end{aligned}
$$

The above transformation is also suitable to our second framework in Section 4.
Now, we provide the details of our generic construction. Given a $\mathbb{Z}_{p}$-linear leakage-resilient predicate encoding with attribute-hiding for predicate $\mathrm{P}: \mathcal{X} \times$ $\mathcal{Y} \rightarrow\{0,1\}$,

- Setup $\left(1^{\lambda}\right):$ Let $N \in \mathbb{N}$ be the parameter of the $\mathbb{Z}_{p}$-linear leakage-resilient predicate encoding with attribute-hiding for predicate P and $N$ is related to $1^{\lambda}$. Run $\mathbb{G} \leftarrow \mathcal{G}\left(1^{\lambda}\right)$, sample $\left(\mathbf{A}, \mathbf{a}^{\perp}\right),\left(\mathbf{B}, \mathbf{b}^{\perp}\right)$ as in Equation (3), pick $\mathbf{k} \leftarrow \mathbb{Z}_{p}^{k+1}, \mathbf{W}_{1}, \ldots, \mathbf{W}_{N} \leftarrow \mathbb{Z}_{p}^{(k+1) \times(k+1)}$. Then pick $\mathbf{r} \leftarrow \mathbb{Z}_{p}^{k}, \mathbf{v} \leftarrow \mathcal{V}$, output

$$
\mathrm{mpk}:=\left(\mathbb{G} ;[\mathbf{A}]_{1},\left[\mathbf{W}_{1}^{\top} \mathbf{A}\right]_{1}, \ldots,\left[\mathbf{W}_{N}^{\top} \mathbf{A}\right]_{1},\left[\mathbf{A}^{\top} \mathbf{k}\right]_{T}\right), \mathrm{mk}:=\left(\mathbf{B}, \mathbf{k}, \mathbf{W}_{1}, \ldots, \mathbf{W}_{N}\right)
$$

- KeyGen(mk, y): Pick $\mathbf{r} \leftarrow \mathbb{Z}_{p}^{k}, \mathbf{z} \leftarrow \mathcal{Z}$ and output $\mathrm{sk}_{\mathbf{y}}:=\left(\mathbf{z}, K_{0}, \mathbf{K}\right)$, where

$$
K_{0}:=[\mathbf{B r}]_{2}, \mathbf{K}:=\operatorname{rkE}\left(\mathbf{y}, \mathbf{z},[\mathbf{k}]_{2}\right) \cdot \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left[\mathbf{W}_{1} \mathbf{B r}\right]_{2}, \ldots,\left[\mathbf{W}_{N} \mathbf{B r}\right]_{2}\right)
$$

- Enc(mpk, $\mathbf{x}, m):$ Pick $\mathbf{s} \leftarrow \mathbb{Z}_{p}^{k}$ and output $\mathrm{ct}_{\mathbf{x}}:=\left(C_{0}, \mathbf{C}, C_{T}\right)$, where

$$
C_{0}:=[\mathbf{A s}]_{1}, \mathbf{C}:=\mathrm{sE}\left(\mathbf{x},\left[\mathbf{W}_{1}^{\top} \mathbf{A s}\right]_{1}, \ldots,\left[\mathbf{W}_{N}^{\top} \mathbf{A s}\right]_{1}\right), C_{T}=\left[\mathbf{k}^{\top} \mathbf{A s}\right]_{T} \cdot m
$$

- $\operatorname{Dec}\left(\mathrm{mpk}, \mathrm{sk}_{\mathbf{y}}, \mathrm{ct}_{\mathbf{x}}\right)$ : output $m^{\prime}=C_{T} \cdot e\left(C_{0}, \mathrm{rD}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{K})\right)^{-1} \cdot e\left(\mathrm{sD}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{C}), K_{0}\right)$.

Correctness. For any $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ such that $\mathrm{P}(\mathbf{x}, \mathbf{y})=1$, we have

$$
\begin{aligned}
& C_{T} \cdot e\left(C_{0}, \mathrm{rD}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{K})\right)^{-1} \\
= & m \cdot\left[\mathbf{k}^{\top} \mathbf{A s}\right]_{T} \cdot e\left([\mathbf{A s}]_{1}, \mathrm{rD}\left(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathrm{rkE}\left(\mathbf{y}, \mathbf{z},[\mathbf{k}]_{2}\right) \cdot \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left[\mathbf{W}_{1} \mathbf{B r}\right]_{2}, \ldots,\left[\mathbf{W}_{N} \mathbf{B r}\right]_{2}\right)\right)\right)^{-1} \\
= & m \cdot\left[\mathbf{k}^{\top} \mathbf{A s}\right]_{T} \cdot e\left([\mathbf{A s}]_{1}, \mathrm{rD}\left(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathrm{rkE}\left(\mathbf{y}, \mathbf{z},[\mathbf{k}]_{2}\right)\right)^{-1}\right. \\
& \cdot e\left([\mathbf{A s}]_{1}, \mathrm{rD}\left(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left[\mathbf{W}_{1} \mathbf{B r}\right]_{2}, \ldots,\left[\mathbf{W}_{N} \mathbf{B r}\right]_{2}\right)\right)\right)^{-1} \\
= & m \cdot\left[\mathbf{k}^{\top} \mathbf{A s}\right]_{T} \cdot e\left([\mathbf{A s}]_{1},[\mathbf{k}]_{2}\right)^{-1} \cdot e\left([\mathbf{A s}]_{1}, \mathrm{rD}\left(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left[\mathbf{W}_{1} \mathbf{B r}\right]_{2}, \ldots,\left[\mathbf{W}_{N} \mathbf{B r}\right]_{2}\right)\right)\right)^{-1} \\
= & m \cdot e\left([\mathbf{A s}]_{1}, \mathrm{rD}\left(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left[\mathbf{W}_{1} \mathbf{B r}\right]_{2}, \ldots,\left[\mathbf{W}_{N} \mathbf{B r}\right]_{2}\right)\right)\right)^{-1} \\
= & m \cdot \mathrm{rD}\left(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathrm{rE}\left(\mathbf{y}, \mathbf{z}, e\left([\mathbf{A s}]_{1},\left[\mathbf{W}_{1} \mathbf{B r}\right]_{2}\right), \ldots, e\left([\mathbf{A s}]_{1},\left[\mathbf{W}_{N} \mathbf{B r}\right]_{2}\right)\right)\right)^{-1} \\
= & m \cdot \mathrm{rD}\left(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathrm{rE}\left(\mathbf{y}, \mathbf{z}, e\left(\left[\mathbf{W}_{1}^{\top} \mathbf{A s}\right]_{1},[\mathbf{B r}]_{2}\right), \ldots, e\left(\left[\mathbf{W}_{N}^{\top} \mathbf{A s}\right]_{1},[\mathbf{B r}]_{2}\right)\right)\right)^{-1} \\
= & m \cdot \mathrm{sD}\left(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathrm{sE}\left(\mathbf{x}, e\left(\left[\mathbf{W}_{1}^{\top} \mathbf{A s}\right]_{1},[\mathbf{B r}]_{2}\right), \ldots, e\left(\left[\mathbf{W}_{N}^{\top} \mathbf{A s}\right]_{1},[\mathbf{B r}]_{2}\right)\right)\right)^{-1} \\
= & m \cdot e\left(\mathrm{sD}\left(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{s E}\left(\mathbf{x},\left[\mathbf{W}_{1}^{\top} \mathbf{A s}\right]_{1}, \ldots,\left[\mathbf{W}_{N}^{\top} \mathbf{A s}\right]_{1}\right)\right),[\mathbf{B r}]_{2}\right)^{-1} \\
= & m \cdot e\left(\mathbf{s D}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{C}), \mathbf{K}_{0}\right)^{-1}
\end{aligned}
$$

In the above equality, we exploit linearity (for lines $3,6,9$ ) and restricted $\alpha$ reconstruction (for lines 4, 8) mentioned in Section 3.1. Thus, $C_{T} \cdot e\left(C_{0}, \mathrm{rD}(\mathbf{x}, \mathbf{y}, \mathbf{z}\right.$, $\mathbf{K}))^{-1} \cdot e\left(\mathbf{s D}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{C}), K_{0}\right)=m$ and the correctness follows readily.

### 3.3 Security

We start by giving some lemmas of $[5,14]$ which will be used throughout the security proof of our framework.
Lemma 1 ([14]). Let an integer $m \geq 3$ and let $p$ be a prime. Let $\delta \leftarrow \mathbb{Z}_{p}^{m}, \tau \leftarrow$ $\mathbb{Z}_{p}^{m}$, and let $\tau^{\prime}$ be chosen uniformly from the set of vectors in $\mathbb{Z}_{p}^{m}$ which are orthogonal to $\delta$ under the dot product modulo $p$. Let $f: \mathbb{Z}_{p}^{m} \rightarrow \mathbf{W}$ be some function. Then there exists any positive constant $c$, such that $\operatorname{dist}\left(\left(\delta, f\left(\tau^{\prime}\right)\right),(\delta, f(\tau))\right) \leq$ $p^{-c}$, as long as $|\mathbf{W}| \leq 4 \cdot\left(1-\frac{1}{p}\right) \cdot p^{m-2 c-2}$.
Suppose that A and B have the same form as Equation (3), then we set

$$
\begin{align*}
\mathrm{PP} & :=\left(\mathbb{G} ; \begin{array}{l}
{[\mathbf{A}]_{1},\left[\mathbf{W}_{1}^{\top} \mathbf{A}\right]_{1}, \ldots,\left[\mathbf{W}_{N}^{\top} \mathbf{A}\right]_{1},} \\
{[\mathbf{B}]_{2},\left[\mathbf{W}_{1} \mathbf{B}\right]_{2}, \ldots,\left[\mathbf{W}_{N} \mathbf{B}\right]_{2}}
\end{array}\right),  \tag{4}\\
\mathrm{PP}^{-} & :=\left(\mathbb{G} ;[\mathbf{A}]_{1},\left[\mathbf{W}_{1}^{\top} \mathbf{A}\right]_{1}, \ldots,\left[\mathbf{W}_{N}^{\top} \mathbf{A}\right]_{1},[\mathbf{B}]_{2}\right)
\end{align*}
$$

where $\mathbf{W}_{1}, \ldots, \mathbf{W}_{N} \leftarrow \mathbb{Z}_{p}^{(k+1) \times(k+1)}$.

Lemma 2 (Parameter-Hiding[5]). The following distributions are statistically indistinguishable:

$$
\begin{aligned}
& \left\{\mathrm{PP},\left[\mathbf{a}^{\perp}\right]_{2}, \begin{array}{l}
{\left[\mathbf{b}^{\perp} \hat{s}\right]_{1},\left[\mathbf{W}_{1}^{\top} \mathbf{b}^{\perp} \hat{s}\right]_{1}, \ldots,\left[\mathbf{W}_{N}^{\top} \mathbf{b}^{\perp} \hat{\hat{s}}\right]_{1}} \\
{\left[\mathbf{a}^{\perp} \hat{r}\right]_{2},\left[\mathbf{W}_{1} \mathbf{a}^{\perp} \hat{r}\right]_{2}, \ldots,\left[\mathbf{W}_{N} \mathbf{a}^{\perp} \hat{r}\right]_{2}}
\end{array}\right\} \text { and } \\
& \begin{cases}\mathrm{PP},\left[\mathbf{a}^{\perp}\right]_{2}, & \left.\begin{array}{l}
\left.\mathbf{b}^{\perp} \hat{s}\right]_{1},\left[\left(\mathbf{W}_{1}^{\top} \mathbf{b}^{\perp}+u_{1} \mathbf{b}^{\perp}\right) \hat{s}\right]_{1}, \ldots,\left[\left(\mathbf{W}_{N}^{\top} \mathbf{b}^{\perp}+u_{N} \mathbf{b}^{\perp}\right) \hat{s}\right]_{1} \\
{\left[\mathbf{a}^{\perp} \hat{r}\right]_{2},\left[\left(\mathbf{W}_{1} \mathbf{a}^{\perp}+u_{1} \mathbf{a}^{\perp}\right) \hat{r}\right]_{2}, \ldots,\left[\left(\mathbf{W}_{N} \mathbf{a}^{\perp}+u_{N} \mathbf{a}^{\perp}\right) \hat{r}\right]_{2}}
\end{array}\right\}\end{cases}
\end{aligned}
$$

where $\hat{s}, \hat{r} \leftarrow \mathbb{Z}_{p}^{*}, \mathbf{u}:=\left(u_{1}, \ldots, u_{N}\right) \leftarrow \mathbb{Z}_{p}^{N}$.
Lemma 3 ( $\mathbb{H}$-hiding[5]). The following distributions are statistically indistinguishable:

$$
\begin{aligned}
& \left\{\mathrm{PP}^{-},\left[\mathbf{a}^{\perp}\right]_{2},[\mathbf{B r}]_{2},\left[\mathbf{W}_{1} \mathbf{B r}+\hat{v}_{1} \mathbf{a}^{\perp}\right]_{2}, \ldots,\left[\mathbf{W}_{N} \mathbf{B r}+\hat{v}_{N} \mathbf{a}^{\perp}\right]_{2}\right\} \text { and } \\
& \left\{\mathrm{PP}^{-},\left[\mathbf{a}^{\perp}\right]_{2},[\mathbf{B r}]_{2},\left[\hat{\mathbf{u}}_{1}\right]_{2}, \ldots,\left[\hat{\mathbf{u}}_{N}\right]_{2}\right\}
\end{aligned}
$$

where $\mathbf{r} \leftarrow \mathbb{Z}_{p}^{k}, \hat{\mathbf{v}}:=\left(\hat{v}_{1}, \ldots, \hat{v}_{N}\right) \leftarrow \mathbb{Z}_{p}^{N}$ and for $i=1, \ldots, N, \hat{\mathbf{u}}_{i} \leftarrow \mathbb{Z}_{p}^{k+1}$ subject to the constraint $\mathbf{A}^{\top} \hat{\mathbf{u}}_{i}=\left(\mathbf{W}_{i}^{\top} \mathbf{A}\right)^{\top} \mathbf{B r}$.

Lemma 4 ( $\mathbb{G}$-uniformity[5]). The following distributions are statistically indistinguishable:

$$
\begin{aligned}
& \left\{\mathrm{PP}^{-},\left[\mathbf{a}^{\perp}\right]_{1},\left[\mathbf{A s}+\mathbf{b}^{\perp} \hat{s}\right]_{2},\left[\mathbf{W}_{1}^{\top}\left(\mathbf{A s}+\mathbf{b}^{\perp} \hat{s}\right)\right]_{1}, \ldots,\left[\mathbf{W}_{N}\left(\mathbf{A s}+\mathbf{b}^{\perp} \hat{s}\right)\right]_{1}\right\} \text { and } \\
& \left\{\mathrm{PP}^{-},\left[\mathbf{a}^{\perp}\right]_{2},\left[\mathbf{A s}+\mathbf{b}^{\perp} \hat{s}\right]_{1},\left[\hat{\mathbf{w}}_{1}\right]_{1}, \ldots,\left[\hat{\mathbf{w}}_{N}\right]_{1}\right\}
\end{aligned}
$$

where $\mathbf{s} \leftarrow \mathbb{Z}_{p}^{k}, \hat{s} \leftarrow \mathbb{Z}_{p}^{*} ; \hat{\mathbf{w}}_{1}, \ldots, \hat{\mathbf{w}}_{N} \leftarrow \mathbb{Z}_{p}^{k+1}$.
Theorem 1. If $k$-Lin assumption holds, the construction described in Section 3.2 is $\left(0, L_{\mathrm{sk}}\right)$-bounded-leakage secure and attribute-hiding. More precisely, for all PPT adversaries $\mathcal{A}$ subject to the restrictions: (1) $\mathcal{A}$ queries $\mathrm{O}_{2}$ and $\mathrm{O}_{3}$ at most $q$ times; (2) The leakage on mk is not allowed and the leakage amount of sk are at most $L_{\mathrm{sk}}$ bits. There exists an algorithm $\mathcal{B}$ such that $\operatorname{Adv}_{\mathcal{A}}^{\operatorname{BLR}-A H}(\lambda) \leq$ $(2 q+1) \operatorname{Adv}_{\mathcal{B}}^{k-\operatorname{Lin}}(\lambda)+\operatorname{negl}(\lambda)$.

Proof. Our proof sketch for the game sequence is shown in Table 2. In Table 2, we use a box to highlight the difference between two adjacent games and the cell marked by "-" means that the corresponding part of $\mathrm{sk}_{\mathbf{y}}$ or ct* is the same as the last game. For the transition from $\mathrm{Game}_{2, i, 1}$ to $\mathrm{Game}_{2, i, 2}$, we employ ParameterHiding lemma, attribute-hiding encoding and attribute-hiding-leakage-resilient encoding mentioned in Section 3.1. In Game ${ }_{3}$ and $\mathrm{Game}_{4}, m^{\prime}$ denotes a random message and $\mathbf{x}^{\prime}$ denotes a random attribute. $\mathrm{Game}_{0}$ is the same as Game ${ }_{\text {BLM-AH }}$. In Game ${ }_{4}$, the advantage of $\mathcal{A}$ is 0 .

Table 2: Our proof sketch for the game sequence.

| game | $i$-th queried secret key $\mathrm{sk}_{\mathrm{y}}$ |  |  | $\mathrm{ct}^{*}$ |  |  | justification |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | $K_{0}$ | rkE( $\mathbf{y}, \mathbf{z}, \cdot) \mid$ | $\mathrm{rE}(\mathbf{y}, \mathbf{z}, \cdot)$ | $C_{0}$ | sE( $\cdot, \cdot$ ) | $C_{T}$ |  |
| Game $_{0}$ | $[\mathrm{Br}]_{2}$ | $[\mathrm{k}]_{2}$ | $\left[\mathbf{W}_{k} \mathbf{B r}\right]_{2}$ | [As $]_{1}$ | $\mathbf{x}^{(b)},\left[\mathbf{W}_{j}^{\top} \mathbf{A s}\right]_{1}$ | $e\left([\mathbf{A s}]_{1},[\mathbf{k}]_{2}\right) \cdot m$ | real game |
| Game ${ }_{1}$ | - | - | - | ${\mathbf{A s}+\mathbf{b}^{\perp} \hat{s}}^{1}$ | $\mathbf{x}^{(b)},\left[\mathbf{W}_{j}^{\top}\left(\mathbf{A s}+\mathbf{b}^{\perp} \hat{s}\right)\right]_{1}$ | $\left.e\left(\underline{\mathbf{A s}+\mathbf{b}^{\perp}}\right]_{1},[\mathbf{k}]_{2}\right) \cdot m$ | $k$-Lin |
| $\underline{\text { Game }_{2, i, 1}}$ | $\underline{\mathrm{Br}+\mathbf{a}^{\perp} \hat{r}} \mathrm{l}_{2}$ | - | $\left.\mathbf{W}_{k}\left(\mathbf{B r}+\mathbf{a}^{\perp} \hat{r}\right)\right]_{2}$ | - | - | - | $k$-Lin |
| Game $_{2, i, 2}$ | - | $\underline{\hat{\mathbf{k}}}{ }_{2}$ | $\left[\mathbf{W}_{k}\left(\mathbf{B r}+\mathbf{a}^{\perp} \hat{r}\right)+\hat{v}_{k} \mathbf{a}^{\perp}\right]_{2}$ | - | - | $-$ | attribute-hiding, Parameter-Hiding, attribute-hiding-leakage-resilient |
| $\underline{\text { Game }_{2, i, 3}}$ | $[\mathrm{Br}]_{2}$ | - | $\left.\mathbf{W}_{k} \mathbf{B r}+\hat{v}_{k}^{j} \mathbf{a}^{\perp}\right]_{2}$ | - | - | - | $k$-Lin |
| Game $_{3}$ | - | - | - | - | - | $e\left(\left[\mathbf{A s}+\mathbf{b}^{ \pm} \hat{s_{1}},[\mathbf{k}]_{2}\right) \cdot m^{\prime}\right.$ | statistically identical |
| Game $_{4}$ | - | - | - | - | $\underline{\mathbf{x}^{\prime}},\left[\mathbf{W}_{j}^{\top}\left(\mathbf{A s}+\mathbf{b}^{ \pm} \hat{s}\right)\right]_{1}$ | - | $\qquad$ <br> I-hiding, $\mathbb{G}$-uniformity, attribute-hiding, attribute-hiding-leakage-resilient |

We denote the advantage of $\mathcal{A}$ in $\operatorname{Game}_{i}$ by $\operatorname{Adv}_{i}(\lambda)$. Then we will show Theorem 1 by proving the indistinguishability among these games with the following lemmas.

Lemma 5 ( $\mathrm{Game}_{0} \stackrel{c}{\approx} \mathrm{Game}_{1}$ ). For all PPT adversary $\mathcal{A}$, there exists an algorithm $\mathcal{B}_{1}$ such that $\left|\operatorname{Adv}_{0}(\lambda)-\operatorname{Adv}_{1}(\lambda)\right| \leq \operatorname{Adv}_{\mathcal{B}_{1}}^{k-\operatorname{Lin}}(\lambda)+2 / p$.

Proof. The proof is a simpler case of the proof of Lemma 6, we omit it here.
Lemma 6 ( $\mathrm{Game}_{2, i-1,3} \stackrel{c}{\approx}$ Game $\left._{2, i, 1}\right)$. For all PPT adversary $\mathcal{A}$ and $i=$ $1, \ldots, q$, there exists an algorithm $\mathcal{B}_{2}$ such that $\left|\operatorname{Adv}_{2, i-1,3}(\lambda)-\operatorname{Adv}_{2, i, 1}(\lambda)\right| \leq$ $\operatorname{Adv}_{\mathcal{B}_{2}}^{k-\operatorname{Lin}}(\lambda)+2 / p$.

Proof. $\mathcal{B}_{2}$ samples $\left(\mathbf{A}, \mathbf{a}^{\perp}\right) \leftarrow \mathcal{D}_{k+1, k}$ along with $\mathbf{W}_{1}, \ldots, \mathbf{W}_{N} \leftarrow \mathbb{Z}_{p}^{(k+1) \times(k+1)}$. We know that $\left\{\mathbf{B r}+\mathbf{a}^{\perp} \hat{r}: \mathbf{r} \leftarrow \mathbb{Z}_{p}^{k}, \hat{r} \leftarrow \mathbb{Z}_{p}\right\}$ is statistically close to the uniform distribution. Then $\mathcal{B}_{2}$ gets as input $\left(\mathbb{G},[\mathbf{B}]_{2},[\mathbf{t}]_{2}\right)=\left(\mathbb{G},[\mathbf{B}]_{2},\left[\mathbf{B r}+\mathbf{a}^{\perp} \hat{r}\right]_{2}\right)$ where either $\hat{r}=0$ or $\hat{r} \leftarrow \mathbb{Z}_{p}^{*}$ and proceeds as follows:
Setup. Pick $\mathbf{k} \leftarrow \mathbb{Z}_{p}^{k+1}, \alpha \leftarrow \mathbb{Z}_{p}$ and set $\hat{\mathbf{k}}:=\mathbf{k}+\alpha \mathbf{a}^{\perp}$. With $\mathbb{G}, \mathbf{A}, \mathbf{W}_{1}, \ldots, \mathbf{W}_{n}$, $\mathcal{B}_{2}$ can simulate mpk $:=\left(\mathbb{G} ;[\mathbf{A}]_{1},\left[\mathbf{W}_{1}^{\top} \mathbf{A}\right]_{1}, \ldots,\left[\mathbf{W}_{n}^{\top} \mathbf{A}\right]_{1},\left[\mathbf{A}^{\top} \mathbf{k}\right]_{T}\right)$.
Key Queries. When $\mathcal{A}$ makes the $j$ 'th Leak or Reveal key query,

- When $j<i$, since $\mathbf{a}^{\perp}, \hat{\mathbf{k}}, \mathbf{W}_{1}, \ldots, \mathbf{W}_{n}$ and $[\mathbf{B}]_{2}$ has been known, semifunctional $\mathrm{sk}_{\mathbf{y}}$ can be generated properly;
- When $j=i, \mathcal{B}_{2}$ generates

$$
\mathrm{sk}_{\mathbf{y}}:=\left(\mathbf{z},[\mathbf{t}]_{2}, \mathrm{rkE}\left(\mathbf{y}, \mathbf{z},[\mathbf{k}]_{2}\right) \cdot \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left[\mathbf{W}_{1} \mathbf{t}\right]_{2}, \ldots,\left[\mathbf{W}_{N} \mathbf{t}\right]_{2}\right)\right)
$$

- When $j>i$, it is not hard to know that normal $\mathrm{sk}_{\mathbf{y}}$ can also be generated properly;
Challenge. Since $\mathbf{b}^{\perp}$ is unknown, As $+\mathbf{b}^{\perp} \hat{s}$ is statistically close to the uniform distribution. Thus, $\mathcal{B}_{2}$ would sample $\tilde{\mathbf{s}} \leftarrow \mathbb{Z}_{p}^{k+1}$ to replace $\mathbf{A s}+\mathbf{b}^{\perp} \hat{s}$. After receiving challenge messages $\left(m^{(0)}, m^{(1)}\right)$ and challenge vectors $\left(\mathbf{x}^{(0)}, \mathbf{x}^{(1)}\right), \mathcal{B}_{2}$ chooses a random bit $b \in\{0,1\}$ and returns

$$
\mathrm{ct}^{*}:=\left([\tilde{\mathbf{s}}]_{1}, \mathrm{sE}\left(\mathbf{x}^{(b)},\left[\mathbf{W}_{1}^{\top} \tilde{\mathbf{s}}\right]_{1}, \ldots,\left[\mathbf{W}_{N}^{\top} \tilde{\mathbf{s}}\right]_{1}\right), e\left([\tilde{\mathbf{s}}]_{1},[\mathbf{k}]_{2}\right) \cdot m^{(b)}\right)
$$

Observe that if $\mathbf{t}=\mathbf{B r}, \mathcal{B}_{2}$ has properly simulated Game $_{2, i-1,3}$ and if $\mathbf{t}=$ $\mathrm{Br}+\mathbf{a}^{\perp} \hat{r}, \mathcal{B}_{2}$ has properly simulated $\mathrm{Game}_{2, i, 1}$. Since $\hat{s}, \hat{r} \leftarrow \mathbb{Z}_{p}^{*}$ yields a $2 / p$ negligible difference in the advantage, Lemma 6 hence holds.

Lemma $7\left(\operatorname{Game}_{2, i, 1} \stackrel{s}{\approx} \operatorname{Game}_{2, i, 2}\right)$. For $i=1, \ldots, q$, it holds that $\mid \operatorname{Adv}_{2, i, 1}(\lambda)-$ $\operatorname{Adv}_{2, i, 2}(\lambda) \mid \approx 0$ as long as the leakage amount of sk are at most $L_{\text {sk }}$ bits.
Proof. Given PP as in Equation (4), we state that Game ${ }_{2, i, 1}$ and Game ${ }_{2, i, 2}$ are statistically indistinguishable if the following distributions $\left\{\mathrm{PP},[\mathbf{k}]_{2},\left[\alpha \mathbf{a}^{\perp}\right]_{2}, \mathrm{ct}^{*}, \mathrm{sk}_{\mathbf{y}}\right\}$ and $\left\{\mathrm{PP},[\mathbf{k}]_{2},\left[\alpha \mathbf{a}^{\perp}\right]_{2}, \mathrm{ct}^{*}, \mathrm{sk}_{\mathbf{y}}^{\prime}\right\}$ are identical where

$$
\begin{aligned}
\mathrm{ct}^{*}= & \left([\mathbf{A s}]_{1}, \mathrm{sE}\left(\mathbf{x}^{(b)},\left\{\left[\mathbf{W}_{k}^{\top} \mathbf{A s}\right]_{1}\right\}_{k \in[N]}\right),\left[\mathbf{k}^{\top} \mathbf{A} \mathbf{s}\right]_{T} \cdot m^{(b)}\right) . \\
& \left(\left[\mathbf{b}^{\perp} \hat{s}\right]_{1}, \mathbf{s E}\left(\mathbf{x}^{(b)},\left\{\left[\mathbf{W}_{k}^{\top} \mathbf{b}^{\perp} \hat{s}\right]_{1}\right\}_{k \in[N]}\right),\left[\mathbf{k}^{\top} \mathbf{b}^{\perp} \hat{s}\right]_{T}\right)
\end{aligned}
$$

and $\mathrm{sk}_{\mathbf{y}}, \mathrm{sk}_{\mathbf{y}}^{\prime}$ are the $i^{\prime}$ th queried key in $\mathrm{Game}_{2, i, 1}$ and $\mathrm{Game}_{2, i, 2}$, respectively. Now we consider the following cases:
(1) If $\mathbf{y} \in \mathcal{Y}$ such that $\left\langle\mathbf{x}^{(0)}, \mathbf{y}\right\rangle=0$ and $\left\langle\mathbf{x}^{(1)}, \mathbf{y}\right\rangle=0$, we have

$$
\begin{aligned}
\mathrm{sk}_{\mathbf{y}}= & \left(\mathbf{1},[\mathrm{Br}]_{2}, \mathrm{rkE}\left(\mathbf{y}, \mathbf{z},[\mathbf{k}]_{2}\right) \cdot \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left\{\left[\mathbf{W}_{k} \mathbf{B r}\right]_{2}\right\}_{k \in[N]}\right)\right) . \\
& \left(\mathbf{z},\left[\mathbf{a}^{\perp} \hat{r}\right]_{2}, \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left\{\left[\mathbf{W}_{k} \mathbf{a}^{\perp} \hat{r}\right]_{2}\right\}_{k \in[N]}\right)\right) \\
\mathrm{sk}_{\mathbf{y}}^{\prime}= & \left(\mathbf{1},[\mathbf{B r}]_{2}, \mathrm{rkE}\left(\mathbf{y}, \mathbf{z},[\mathbf{k}]_{2}\right) \cdot \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left\{\left[\mathbf{W}_{k} \mathbf{B r}\right]_{2}\right\}_{k \in[N]}\right)\right) . \\
& \left(\mathbf{z},\left[\mathbf{a}^{\perp} \hat{r}\right]_{2}, \mathrm{rkE}\left(\mathbf{y}, \mathbf{z},\left[\alpha \mathbf{a}^{\perp}\right]_{2}\right) \cdot \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left\{\left[\mathbf{W}_{k} \mathbf{a}^{\perp} \hat{r}+\hat{v}_{k} \mathbf{a}^{\perp}\right]_{2}\right\}_{k \in[N]}\right)\right)
\end{aligned}
$$

where $\hat{\mathbf{v}}:=\left(\hat{v}_{1}, \ldots, \hat{v}_{N}\right) \leftarrow \mathbb{Z}_{p}^{N}$ and the length of vector $\mathbf{1}:=(1, \ldots, 1)$ is equal to the length of $\mathbf{z}$. We observe that it suffices to show that

$$
\begin{aligned}
& \left\{\begin{array}{l}
\text { aux : PP, }[\mathbf{k}]_{2},[\mathbf{B}]_{2},\left[\alpha \mathbf{a}^{\perp}\right]_{2} \\
\mathrm{ct}_{\mathbf{x}}:\left[\mathbf{b}^{\perp} \hat{s}\right]_{1}, \mathrm{sE}\left(\mathbf{x}^{(b)},\left\{\left[\mathbf{W}_{k}^{\top} \mathbf{b}^{\perp} \hat{s}\right]_{1}\right\}_{k \in[N]}\right),\left[\mathbf{k}^{\top} \mathbf{b}^{\perp} \hat{s}\right]_{T} \\
\text { sky: } \mathbf{z},\left[\mathbf{a}^{\perp} \hat{r}\right]_{2}, \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left\{\left[\mathbf{W}_{k} \mathbf{a}^{\perp} \hat{r}\right]_{2}\right\}_{k \in[N]}\right)
\end{array}\right\} \text { and } \\
& \left\{\begin{array}{l}
\text { aux : PP, }[\mathbf{k}]_{2},[\mathbf{B}]_{2},\left[\alpha \mathbf{a}^{\perp}\right]_{2} \\
\mathrm{ct}_{\mathbf{x}}:\left[\mathbf{b}^{\perp} \hat{s}\right]_{1}, \mathrm{sE}\left(\mathbf{x}^{(b)},\left\{\left[\mathbf{W}_{k}^{\top} \mathbf{b}^{\perp} \hat{s}\right]_{1}\right\}_{k \in[N]}\right),\left[\mathbf{k}^{\top} \mathbf{b}^{\perp} \hat{s}\right]_{T} \\
\mathrm{sk} \mathbf{y}: \mathbf{z},\left[\mathbf{a}^{\perp} \hat{r}\right]_{2}, \mathrm{rkE}\left(\mathbf{y}, \mathbf{z},\left[\alpha \mathbf{a}^{\perp}\right]_{2}\right) \cdot \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left\{\left[\mathbf{W}_{k} \mathbf{a}^{\perp} \hat{r}+\hat{v}_{k} \mathbf{a}^{\perp}\right]_{2}\right\}_{k \in[N]}\right)
\end{array}\right\}
\end{aligned}
$$

are indistinguishable. By parameter-hiding in Lemma 2, it suffices to show that:

$$
\begin{aligned}
& \left\{\begin{array}{l}
\operatorname{aux}: \mathrm{PP},[\mathbf{k}]_{2},[\mathbf{B}]_{2},\left[\alpha \mathbf{a}^{\perp}\right]_{2} \\
\mathrm{ct}_{\mathbf{x}}:\left[\mathbf{b}^{\perp} \hat{s}\right]_{1}, \mathrm{sE}\left(\mathbf{x}^{(b)},\left\{\left[\left(\mathbf{W}_{k}^{\top} \mathbf{b}^{\perp}+u_{k} \mathbf{b}^{\perp}\right) \hat{s}\right]_{1}\right\}_{k \in[N]}\right),\left[\mathbf{k}^{\top} \mathbf{b}^{\perp} \hat{s}\right]_{T} \\
\mathrm{sk} \mathbf{y}_{\mathbf{y}}: \mathbf{z},\left[\mathbf{a}^{\perp} \hat{r}\right]_{2}, \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left\{\left[\left(\mathbf{W}_{k} \mathbf{a}^{\perp}+u_{k} \mathbf{a}^{\perp}\right) \hat{r}\right]_{2}\right\}_{k \in[N]}\right)
\end{array}\right\} \text { and } \\
& \left\{\begin{array}{l}
\text { aux }: \operatorname{PP},[\mathbf{k}]_{2},[\mathbf{B}]_{2},\left[\alpha \mathbf{a}^{\perp}\right]_{2} \\
\mathrm{ct}_{\mathbf{x}}:\left[\mathbf{b}^{\perp} \hat{s}\right]_{1}, \mathrm{sE}\left(\mathbf{x}^{(b)},\left\{\left[\left(\mathbf{W}_{k}^{\top} \mathbf{b}^{\perp}+u_{k} \mathbf{b}^{\perp}\right) \hat{s}\right]_{1}\right\}_{k \in[N]}\right),\left[\mathbf{k}^{\top} \mathbf{b}^{\perp} \hat{s}\right]_{T} \\
\mathrm{sk} \mathbf{k}_{\mathbf{y}}: \mathbf{z},\left[\mathbf{a}^{\perp} \hat{r}\right]_{2}, \mathrm{rkE}\left(\mathbf{y}, \mathbf{z},\left[\alpha \mathbf{a}^{\perp}\right]_{2}\right) \cdot \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left\{\left[\left(\mathbf{W}_{k} \mathbf{a}^{\perp}+u_{k} \mathbf{a}^{\perp}\right) \hat{r}+\hat{v}_{k} \mathbf{a}^{\perp}\right]_{2}\right\}_{k \in[N]}\right]
\end{array}\right\}
\end{aligned}
$$

are indistinguishable. Let $\hat{g}_{0}=\left[\mathbf{b}^{\perp} \hat{s}\right]_{1}, \hat{h}_{0}=\left[\mathbf{a}^{\perp} \hat{r}\right]_{2}$ and set $\left[\mathbf{a}^{\perp}\right]=\left(\hat{h}_{0}\right)^{\beta}$, we note that

$$
\begin{aligned}
& \mathrm{sE}\left(\mathbf{x}^{(b)},\left\{\left[\left(\mathbf{W}_{k}^{\top} \mathbf{b}^{\perp}+u_{k} \mathbf{b}^{\perp}\right) \hat{s}\right]_{1}\right\}_{k \in[N]}\right)=\mathrm{sE}\left(\mathbf{x}^{(b)},\left\{\left[\mathbf{W}_{k}^{\top} \mathbf{b}^{\perp} \hat{s}\right]_{1}\right\}_{k \in[N]}\right) \cdot \hat{g}_{0}^{\mathrm{sE}\left(\mathbf{x}^{(b)}, \mathbf{u}\right)}, \\
& \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left\{\left[\left(\mathbf{W}_{k} \mathbf{a}^{\perp}+u_{k} \mathbf{a}^{\perp}\right) \hat{r}\right]_{2}\right\}_{k \in[N]}\right)=\mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left\{\left[\mathbf{W}_{k} \mathbf{a}^{\perp} \hat{r}\right]_{2}\right\}_{k \in[N]}\right) \cdot \hat{h}_{0}^{\mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{u})} \\
& \mathrm{rkE}\left(\mathbf{y}, \mathbf{z},\left[\alpha \mathbf{a}^{\perp}\right]_{2}\right) \cdot \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left\{\left[\left(\mathbf{W}_{k} \mathbf{a}^{\perp}+u_{k} \mathbf{a}^{\perp}\right) \hat{r}+\hat{v}_{k} \mathbf{a}^{\perp}\right]_{2}\right\}_{k \in[N]}\right) \\
= & \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left\{\left[\mathbf{W}_{k} \mathbf{a}^{\perp} \hat{r}\right]_{2}\right\}_{k \in[N]}\right) \cdot \hat{h}_{0}^{\mathrm{rkE}(\mathbf{y}, \mathbf{z}, \beta \alpha)+\mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{u})+\mathrm{rE}(\mathbf{y}, \mathbf{z}, \beta \hat{\mathbf{v}})} .
\end{aligned}
$$

Since $\mathcal{A}$ can only make Leak query on $\mathrm{sk}_{\mathbf{y}}$, according to attribute-hiding-leakage-resilient encoding, it holds that $\{\mathbf{x}, \mathbf{y}, \mathrm{sE}(\mathbf{x}, \mathbf{u}), f(\mathbf{z}, \mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{u}))\}$ and $\{\mathbf{x}, \mathbf{y}, \mathbf{r}\}$ are indistinguishable. In other words, the adversary $\mathcal{A}$ cannot get any useful information to distinguish between $\mathrm{sk}_{\mathbf{y}}$ and $\mathrm{sk}_{\mathbf{y}}^{\prime}$.
(2) If $\mathbf{y} \in \mathcal{Y}$ such that $<\mathbf{x}^{(0)}, \mathbf{y}>\neq 0$ and $<\mathbf{x}^{(1)}, \mathbf{y}>\neq 0$, the proof is also analogous to the proof of last case. Except that we should use attribute-hiding encoding, which claims that $\{\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{s E}(\mathbf{x}, \mathbf{u}), \mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{u})\}$ and $\{\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{r}\}$ are indistinguishable.

Finally, Lemma 7 holds.
Lemma $8\left(\right.$ Game $_{2, i, 2} \stackrel{c}{\approx}$ Game $\left._{2, i, 3}\right)$. For all PPT adversary $\mathcal{A}$ and $i=1, \ldots, q$, there exists an algorithm $\mathcal{B}_{3}$ such that $\left|\operatorname{Adv}_{2, i, 2}(\lambda)-\operatorname{Adv}_{2, i, 3}(\lambda)\right| \leq \operatorname{Adv}_{\mathcal{B}_{3}}^{k-\operatorname{Lin}}(\lambda)+$ $2 / p$

Proof. The proof is completely analogous to Lemma 6.
Lemma $9\left(\right.$ Game $\left._{2, q, 3} \stackrel{s}{\approx} \operatorname{Game}_{3}\right)$. For $i=1, \ldots, q$, it holds that $\mid \operatorname{Adv}_{2, q, 3}(\lambda)-$ $\operatorname{Adv}_{3}(\lambda) \mid \approx 0$

Proof. First, pick $\hat{\mathbf{k}} \leftarrow \mathbb{Z}_{p}^{k+1}, \alpha \leftarrow \mathbb{Z}_{p}$ and set $\mathbf{k}:=\hat{\mathbf{k}}-\alpha \mathbf{a}^{\perp}$. Given just $\left(\mathrm{PP},\left[\mathbf{a}^{\perp}\right]_{2},[\hat{\mathbf{k}}]_{2}\right)$, we can simulate the setup phase and answer key queries as follows:
Setup. Since $e\left([\mathbf{A}]_{1},[\hat{\mathbf{k}}]_{2}\right):=\left[\mathbf{A}^{\top} \mathbf{k}-\alpha \mathbf{A}^{\top} \mathbf{a}^{\perp}\right]_{T}=\left[\mathbf{A}^{\top} \mathbf{k}\right]_{T}$, then we can simulate $\mathrm{mpk}:=\left(\mathbb{G} ;[\mathbf{A}]_{1},\left[\mathbf{W}_{1}^{\top} \mathbf{A}\right]_{1}, \ldots,\left[\mathbf{W}_{N}^{\top} \mathbf{A}\right]_{1},\left[\mathbf{A}^{\top} \mathbf{k}\right]_{T}\right)$.
Key Queries. For the $j$ 'th key query for $\mathbf{y}$, we can generate a semi-functional secret key properly:

$$
\mathrm{sk}_{\mathbf{y}}:=\left(\mathbf{z},[\mathbf{B r}]_{2}, \operatorname{rkE}\left(\mathbf{y}, \mathbf{z},[\hat{\mathbf{k}}]_{2}\right) \cdot \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left\{\left[\mathbf{W}_{k} \mathbf{B r}+\hat{v}_{k}^{j} \mathbf{a}^{\perp}\right]_{2}\right\}_{k \in[N]}\right)\right)
$$

Challenge. Now, observe that the challenge ciphertext in $\mathrm{Game}_{2, q, 3}$ is given by:

$$
\begin{aligned}
\mathrm{ct}^{*}:= & \left(C_{0}=\left[\mathbf{A} \mathbf{s}+\mathbf{b}^{\perp} \hat{s}\right]_{1}, \mathbf{C}:=\mathrm{sE}\left(\mathbf{x}^{(b)},\left\{\left[\mathbf{W}_{k}^{\top}\left(\mathbf{A} \mathbf{s}+\mathbf{b}^{\perp} \hat{s}\right\}_{k \in[N]}\right)\right]_{1}\right)\right. \\
& \left.C^{\prime}=e\left(\left[\mathbf{A} \mathbf{s}+\mathbf{b}^{\perp} \hat{s}\right]_{1},[\mathbf{k}]_{2}\right) \cdot m^{(b)}\right)
\end{aligned}
$$



Recall that (mpk, $\left.[\mathbf{B}]_{2}, \hat{\mathbf{k}}\right)$ and $\left(C_{0}, \mathbf{C}\right)$ are statistically independent of $\alpha \leftarrow$ $\mathbb{Z}_{p}$, then we can say that $e\left(\left[\mathbf{b}^{\perp} \hat{s}\right]_{1},\left[\mathbf{a}^{\perp}\right]_{2}\right)^{-\alpha}$ is uniformly distributed over $\mathbb{G}_{T}$. This implies ct* is identically distributed to semi-functional encryption of a random message in $G_{T}$, as in $\mathrm{Game}_{3}$. Thus, Lemma 9 holds.

Lemma $10\left(\mathrm{Game}_{3} \stackrel{s}{\approx} \mathrm{Game}_{4}\right)$. For $i=1, \ldots, q$, it holds that $\mid \operatorname{Adv}_{3}(\lambda)-$ $\operatorname{Adv}_{4}(\lambda) \mid \approx 0$

Proof. Pick $\hat{\mathbf{k}} \leftarrow \mathbb{Z}_{p}^{k+1}, \alpha \leftarrow \mathbb{Z}_{p}$ and set $\mathbf{k}:=\hat{\mathbf{k}}-\alpha \mathbf{a}^{\perp}$. Given just $\left(\mathrm{PP}^{-},\left[\mathbf{a}^{\perp}\right]_{2},[\hat{\mathbf{k}}]_{2}\right)$, we note that $\left[\mathbf{W}_{i} \mathbf{B}\right]_{2}$ will not be simulated to ensure $\mathbb{G}$-uniformity holds. But we can still simulate the setup phase and answer key queries as follows:
Setup. We can simulate mpk $:=\left(\mathbb{G} ;[\mathbf{A}]_{1},\left[\mathbf{W}_{1}^{\top} \mathbf{A}\right]_{1}, \ldots,\left[\mathbf{W}_{N}^{\top} \mathbf{A}\right]_{1},\left[\mathbf{A}^{\top} \mathbf{k}\right]_{T}\right)$.
Key Queries. For the $j$ 'th key query for $\mathbf{y}$, by $\mathbb{H}$-hiding in Lemma 3, we can simulate a semi-functional secret key:

$$
\mathrm{sk}_{\mathbf{y}}:=\left(\mathbf{z},[\mathbf{B r}]_{2}, \mathrm{rkE}\left(\mathbf{y}, \mathbf{z},[\hat{\mathbf{k}}]_{2}\right) \cdot \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left[\hat{\mathbf{u}}_{1}^{j}\right]_{2}, \ldots,\left[\hat{\mathbf{u}}_{N}^{j}\right]_{2}\right)\right)
$$

where for $i=1, \ldots, N, \hat{\mathbf{u}}_{i}^{j} \leftarrow \mathbb{Z}_{p}^{k+1}$ subject to the constraint $\mathbf{A}^{\top} \hat{\mathbf{u}}_{i}^{j}=\left(\mathbf{W}_{i}^{\top} \mathbf{A}\right)^{\top} \mathbf{B r}$.
Challenge. Now, observe that the challenge ciphertext in $\mathrm{Game}_{2, q, 3}$ is given by:

$$
C_{0}=\left[\mathbf{A s}+\mathbf{b}^{\perp} \hat{s}\right]_{1}, \mathbf{C}:=\mathrm{sE}\left(\mathbf{x}^{(b)},\left\{\left[\mathbf{W}_{k}^{\top}\left(\mathbf{A} \mathbf{s}+\mathbf{b}^{\perp} \hat{s}\right)\right]_{1}\right\}_{k \in[N]}\right), C^{\prime}=e\left(\left[\mathbf{A} \mathbf{s}+\mathbf{b}^{\perp} \hat{s}\right]_{1},[\hat{\mathbf{k}}]_{2}\right) \cdot m^{\prime}
$$

where $C^{\prime}$ is is uniformly distributed over $G_{T}$. By $\mathbb{G}$-uniformity in Lemma 4 , then

$$
\begin{aligned}
&\left\{\left[\mathbf{A s}+\mathbf{b}^{\perp} \hat{s}\right]_{1},\left[\mathbf{W}_{1}^{\top}\left(\mathbf{A s}+\mathbf{b}^{\perp} \hat{s}\right)\right]_{1}, \ldots,\left[\mathbf{W}_{N}^{\top}\left(\mathbf{A} \mathbf{s}+\mathbf{b}^{\perp} \hat{s}\right)\right]_{1}\right\} \\
& \stackrel{s}{\approx}\left\{\left[\mathbf{A s}+\mathbf{b}^{\perp} \hat{s}\right]_{1},\left[\hat{\mathbf{w}}_{1}\right]_{1}, \ldots,\left[\hat{\mathbf{w}}_{N}\right]_{1}\right\}
\end{aligned}
$$

where $\hat{\mathbf{w}}_{1}, \ldots, \hat{\mathbf{w}}_{N} \leftarrow \mathbb{Z}_{p}^{k+1}$. Note that $\mathcal{A}$ has no idea any information about $\mathbf{W}_{i} \mathbf{B}$ from $\mathrm{sk}_{\mathbf{y}}$ and mpk and hence $\mathbb{G}$-uniformity holds. So we can rewrite $\mathbf{C}:=$ $\mathrm{sE}\left(\mathbf{x}^{(b)},\left[\hat{\mathbf{w}}_{1}\right]_{1}, \ldots,\left[\hat{\mathbf{w}}_{N}\right]_{1}\right)$. From attribute-hiding and attribute-hiding-leakage-resilient encoding, we can say that $\mathbf{C}$ is uniformly distributed over $G_{1}^{\mathrm{sE}(\cdot)}$. Thus, Lemma 10 holds.

Finally, we complete the proof of Theorem 1 by showing the above lemmas which imply the indistinguishability between $\mathrm{Game}_{0}$ and $\mathrm{Game}_{4}$.

## 4 Leakage-resilient ABE in the CLM

In this section, we present our second leakage-resilient ABE framework, which is compatible with ZCG+18 but more versatile. Note that an overview of this generic construction has been present in Equation (2).

### 4.1 Leakage-resilient Predicate Encoding

We define a $\mathbb{Z}_{p}$-linear leakage-resilient predicate encoding for predicate $\mathrm{P}: \mathcal{X} \times$ $\mathcal{Y} \rightarrow\{0,1\}$. It consists of a set of deterministic algorithms (mkE, mE, rkE, rE, sE, $\mathrm{sD}, \mathrm{rD})$ and satisfies the following properties:

- (linearity.) For all $(\mathbf{x}, \mathbf{y}, \mathbf{v}, \mathbf{z}) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{V} \times \mathcal{Z}, \operatorname{mkE}(\mathbf{v}, \cdot), \mathrm{mE}(\mathbf{v}, \cdot), \mathrm{rkE}(\mathbf{y}, \mathbf{z}, \cdot)$, $\mathrm{rE}(\mathbf{y}, \mathbf{z}, \cdot \cdot), \mathrm{sE}(\mathbf{x}, \cdot), \mathrm{sD}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \cdot), \mathrm{rD}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \cdot)$ are $\mathbb{Z}_{p}$-linear.
- (restricted $\alpha$-reconstruction.) This property is the same as restricted $\alpha$-reconstruction in Section 3.1.
- ( $\alpha$-privacy.) For all $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ such that $\mathrm{P}(\mathbf{x}, \mathbf{y})=0$, the distributions $\{\mathbf{x}, \mathbf{y}, \mathbf{z}, \alpha, \mathrm{sE}(\mathbf{x}, \mathbf{w}), \mathrm{rkE}(\mathbf{y}, \mathbf{z}, \alpha)+\mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{w})\}$ and $\{\mathbf{x}, \mathbf{y}, \mathbf{z}, \alpha, \mathrm{sE}(\mathbf{x}, \mathbf{w}), \mathrm{rE}(\mathbf{y}, \mathbf{z}$, $\mathbf{w})\}$ are identical, where the randomness is taken over $\mathbf{w} \leftarrow \mathcal{W}$.
- ( $\alpha$-leakage-resilient.) For all $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ such that $\mathrm{P}(\mathbf{x}, \mathbf{y})=1$ and all $\alpha \in \mathbb{Z}_{p}, \mathbf{z} \in \mathcal{Z}, \mathbf{v} \in \mathcal{V}$, the distributions $\{\mathbf{x}, \mathbf{y}, \alpha, \mathrm{sE}(\mathbf{x}, \mathbf{w}), f(\mathbf{z}, \mathrm{rkE}(\mathbf{y}, \mathbf{z}, \alpha)+$ $\mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{w}))\}$ and $\{\mathbf{x}, \mathbf{y}, \alpha, \mathrm{sE}(\mathbf{x}, \mathbf{w}), f(\mathbf{z}, \mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{w}))\}$ are identical, where $\mathbf{w} \leftarrow$ $\mathcal{W}$. In addition, the distributions $\{\mathbf{x}, \alpha, \mathrm{sE}(\mathbf{x}, \mathbf{w}), f(\mathbf{v}, \operatorname{mkE}(\mathbf{v}, \alpha)+\mathrm{mE}(\mathbf{v}, \mathbf{w}))\}$ and $\{\mathbf{x}, \alpha, \mathrm{sE}(\mathbf{x}, \mathbf{w}), f(\mathbf{v}, \mathrm{mE}(\mathbf{v}, \mathbf{w}))\}$ are also identical.
- (delegable.) There exits a linear algorithm dE such that for all $\alpha \in \mathbb{Z}_{p}, \mathbf{v} \in$ $\mathcal{V}, \mathbf{z} \in \mathcal{Z}, \mathbf{w} \in \mathcal{W}, \mathbf{y} \in \mathcal{Y}$, it holds that $\operatorname{dE}(\mathbf{y}, \operatorname{mkE}(\mathbf{v}, \alpha)+\mathrm{mE}(\mathbf{v}, \mathbf{w}))=$ $\mathrm{rkE}(\mathbf{y}, \mathbf{z}, \alpha)+\mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{w})$. Note that the algorithm dE implies a linear map $S: \mathcal{Y} \times \mathcal{V} \rightarrow \mathcal{Z}$.
- (re-randomizable.) For all $\alpha \in \mathbb{Z}_{p}, \mathbf{v}, \mathbf{v}^{\prime} \in \mathcal{V}, \mathbf{w} \in \mathcal{W}$, there exists a linear algorithm $m R$ such that $m R\left(\mathbf{v}, \mathbf{v}^{\prime}, \operatorname{mkE}(\mathbf{v}, \alpha)+\mathrm{mE}(\mathbf{v}, \mathbf{w})\right)=m \mathrm{kE}\left(\mathbf{v}^{\prime}, \alpha\right)+$ $\mathrm{mE}\left(\mathbf{v}^{\prime}, \mathbf{w}\right)$. Similarly, for all $\alpha \in \mathbb{Z}_{p}, \mathbf{z}, \mathbf{z}^{\prime} \in \mathcal{Z}, \mathbf{w} \in \mathcal{W}, \mathbf{y} \in \mathcal{Y}$, there exists a linear algorithm $k R$ such that $\operatorname{kR}\left(\mathbf{z}, \mathbf{z}^{\prime}, \mathrm{rkE}(\mathbf{y}, \mathbf{z}, \alpha)+\mathrm{rE}(\mathbf{y}, \mathbf{z}, \mathbf{w})\right)=$ rkE $\left(\mathbf{y}, \mathbf{z}^{\prime}, \alpha\right)+\mathrm{rE}\left(\mathbf{y}, \mathbf{z}^{\prime}, \mathbf{w}\right)$


### 4.2 Generic Construction

Given a $\mathbb{Z}_{p}$-linear leakage-resilient predicate encoding for predicate $\mathrm{P}: \mathcal{X} \times \mathcal{Y} \rightarrow$ $\{0,1\}$,

- Setup $\left(1^{\lambda}\right)$ : This algorithm is similar to the setup algorithm in Section 3.2. Run $\mathbb{G} \leftarrow \mathcal{G}\left(1^{\lambda}\right)$, sample $\left(\mathbf{A}, \mathbf{a}^{\perp}\right),\left(\mathbf{B}, \mathbf{b}^{\perp}\right)$ as in Equation (3), pick $\mathbf{k} \leftarrow$ $\mathbb{Z}_{p}^{k+1}, \mathbf{W}_{1}, \ldots, \mathbf{W}_{N} \leftarrow \mathbb{Z}_{p}^{(k+1) \times(k+1)}, \mathbf{r} \leftarrow \mathbb{Z}_{p}^{k}, \mathbf{v} \leftarrow \mathcal{V}$, output

$$
\begin{aligned}
\mathrm{mpk} & :=\left(\mathbb{G} ; \begin{array}{l}
{[\mathbf{A}]_{1},\left[\mathbf{W}_{1}^{\top} \mathbf{A}\right]_{1}, \ldots,\left[\mathbf{W}_{N}^{\top} \mathbf{A}\right]_{1},\left[\mathbf{A}^{\top} \mathbf{k}\right]_{T},} \\
{[\mathbf{B}]_{2},\left[\mathbf{W}_{1} \mathbf{B}\right]_{2}, \ldots,\left[\mathbf{W}_{N} \mathbf{B}\right]_{2}}
\end{array}\right) \\
\mathrm{mk} & :=\left(\mathbf{v},[\mathbf{B r}]_{2}, \operatorname{mkE}\left(\mathbf{v},[\mathbf{k}]_{2}\right) \cdot \mathrm{mE}\left(\mathbf{v},\left[\mathbf{W}_{1} \mathbf{B r}\right]_{2}, \ldots,\left[\mathbf{W}_{N} \mathbf{B r}\right]_{2}\right)\right)
\end{aligned}
$$

where we set $K_{0}=[\mathbf{B r}]_{2}, \mathbf{K}=\operatorname{mkE}\left(\mathbf{v},[\mathbf{k}]_{2}\right) \cdot \mathrm{mE}\left(\mathbf{v},\left[\mathbf{W}_{1} \mathbf{B r}\right]_{2}, \ldots,\left[\mathbf{W}_{N} \mathbf{B r}\right]_{2}\right)$.

- Update $\left(\mathrm{mpk}, \mathrm{sk}_{\mathbf{y}}\right)$ : If $\mathbf{y}=\epsilon$, then $\mathrm{sk}_{\mathbf{y}}$ is a master key and we rewrite it as $\mathrm{mk}:=\left(\mathbf{v},[\mathbf{B r}]_{2}, \mathbf{K}\right)$. Pick $\tilde{\mathbf{r}} \leftarrow \mathbb{Z}_{p}^{k}, \mathbf{v}^{\prime} \leftarrow \mathcal{V}$, we set $\mathbf{r}^{\prime}=\mathbf{r}+\tilde{\mathbf{r}}$ and output

$$
\begin{aligned}
\mathrm{mk}^{\prime}:= & \left(\mathbf{v}^{\prime},\left[\mathbf{B r}^{\prime}\right]_{2}, \mathrm{mR}\left(\mathbf{v}, \mathbf{v}^{\prime}, \mathbf{K}\right) \cdot \mathrm{mE}\left(\mathbf{v}^{\prime},\left[\mathbf{W}_{1} \mathbf{B} \tilde{\mathbf{r}}\right]_{2}, \ldots,\left[\mathbf{W}_{N} \mathbf{B} \tilde{\mathbf{r}}\right]_{2}\right)\right) \\
& \Downarrow \\
\mathrm{mk} & =\left(\mathbf{v}^{\prime},\left[\mathbf{B r}^{\prime}\right]_{2}, \operatorname{mkE}\left(\mathbf{v}^{\prime},[\mathbf{k}]_{2}\right) \cdot \mathrm{mE}\left(\mathbf{v}^{\prime},\left[\mathbf{W}_{1} \mathbf{B r}^{\prime}\right]_{2}, \ldots,\left[\mathbf{W}_{N} \mathbf{B r}\right]_{2}\right)\right)
\end{aligned}
$$

Thus, we can generate a new master key mk' with the same distribution as $m k$. If $\mathbf{y} \in \mathcal{Y}$, sk $\mathbf{y}_{\mathbf{y}}$ is a user secret key. Similarly, we can generate a new secret key $\mathrm{sk}_{\mathrm{y}}^{\prime}$ using the algorithm kR .

- KeyGen(mk, y): Parse mk:=( $\left.v,[\mathbf{B r}]_{2}, \mathbf{K}\right)$. we compute $\mathbf{z} \leftarrow S(\mathbf{y}, \mathbf{v})$ and

$$
\mathrm{dE}(\mathbf{y}, \mathbf{K})=\operatorname{rkE}\left(\mathbf{y}, \mathbf{z},[\mathbf{k}]_{2}\right) \cdot \mathrm{rE}\left(\mathbf{y}, \mathbf{z},\left[\mathbf{W}_{1} \mathbf{B r}\right]_{2}, \ldots,\left[\mathbf{W}_{N} \mathbf{B r}\right]_{2}\right)
$$

Then pick $\tilde{\mathbf{r}} \leftarrow \mathbb{Z}_{p}^{k}, \mathbf{z}^{\prime} \leftarrow \mathcal{Z}$ and set $\mathbf{r}^{\prime}=\mathbf{r}+\tilde{\mathbf{r}}$. Output

$$
\begin{aligned}
\mathrm{sk}_{\mathbf{y}}:=\left(\mathbf{z}^{\prime},\left[\mathbf{B r}^{\prime}\right]_{2}, \mathrm{kR}\left(\mathbf{z}, \mathbf{z}^{\prime}, \mathrm{dE}(\mathbf{y}, \mathbf{K})\right) \cdot \mathrm{rE}\left(\mathbf{y}, \mathbf{z}^{\prime},\left[\mathbf{W}_{1} \mathbf{B} \tilde{\mathbf{r}}\right]_{2}, \ldots,\left[\mathbf{W}_{N} \mathbf{B} \tilde{\mathbf{r}}\right]_{2}\right)\right) \\
\Downarrow \\
\mathrm{sk}_{\mathbf{y}}:=\left(\mathbf{z}^{\prime},\left[\mathbf{B r}^{\prime}\right]_{2}, \operatorname{rkE}\left(\mathbf{y}, \mathbf{z}^{\prime},[\mathbf{k}]_{2}\right) \cdot \mathrm{rE}\left(\mathbf{y}, \mathbf{z}^{\prime},\left[\mathbf{W}_{1} \mathbf{B r}^{\prime}\right]_{2}, \ldots,\left[\mathbf{W}_{N} \mathbf{B r}^{\prime}\right]_{2}\right)\right)
\end{aligned}
$$

Similar to mk, here we also set $K_{0}=\left[\mathbf{B r}^{\prime}\right]_{2}$ and

$$
\mathbf{K}=\operatorname{rkE}\left(\mathbf{y}, \mathbf{z}^{\prime},[\mathbf{k}]_{2}\right) \cdot \mathrm{rE}\left(\mathbf{y}, \mathbf{z}^{\prime},\left[\mathbf{W}_{1} \mathbf{B r}^{\prime}\right]_{2}, \ldots,\left[\mathbf{W}_{N} \mathbf{B r}\right]_{2}\right)
$$

- Enc $(\mathrm{mpk}, \mathbf{x}, m):$ Pick $\mathrm{s} \leftarrow \mathbb{Z}_{p}^{k}$ and output $\mathrm{ct}_{\mathbf{x}}:=\left(C_{0}, \mathbf{C}, C_{T}\right)$, where

$$
C_{0}:=[\mathbf{A s}]_{1}, \mathbf{C}:=\mathrm{sE}\left(\mathbf{x},\left[\mathbf{W}_{1}^{\top} \mathbf{A} \mathbf{s}\right]_{1}, \ldots,\left[\mathbf{W}_{N}^{\top} \mathbf{A s}\right]_{1}\right), C_{T}=\left[\mathbf{k}^{\top} \mathbf{A} \mathbf{s}\right]_{T} \cdot m
$$

- $\operatorname{Dec}\left(\mathrm{mpk}, \mathrm{sk}_{\mathbf{y}}, \mathrm{ct}_{\mathbf{x}}\right):$ Parse $\mathrm{sk}_{\mathbf{y}}:=\left(\mathbf{z}, K_{0}, \mathbf{K}\right), \mathrm{ct}_{\mathbf{x}}:=\left(C_{0}, \mathbf{C}, C_{T}\right)$ and output $m^{\prime}=C_{T} \cdot e\left(C_{0}, \mathrm{rD}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{K})\right)^{-1} \cdot e\left(\mathbf{s D}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{C}), K_{0}\right)$.

Correctness. Since linearity and restricted $\alpha$-reconstruction (for $\operatorname{rkE}(\mathbf{y}, \mathbf{z}, \cdot)$, $\mathrm{rE}(\mathbf{y}, \mathbf{z}, \cdot), \mathrm{sE}(\mathbf{x}, \cdot), \mathrm{sD}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \cdot), \mathrm{rD}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \cdot))$ are similar to ones in Section 3.1, the correctness also follows Section 3.2.

### 4.3 Security

Theorem 2. If $k$-Lin assumption holds, the scheme described in Section 4.2 is $\left(L_{\mathrm{mk}}, L_{\mathrm{sk}}\right)$-continual-leakage secure. More precisely, for all PPT adversaries $\mathcal{A}$ subject to the restrictions: (1) $\mathcal{A}$ makes at most $q \mathrm{O}_{2}^{\prime}$ and $\mathrm{O}_{3}^{\prime}$ queries; (2) The leakage amount of mk and sk are at most $L_{\mathrm{mk}}, L_{\mathrm{sk}}$ bits, respectively. There exists an algorithm $\mathcal{B}$ such that $\operatorname{Adv}_{\mathcal{A}}^{\mathrm{CLR}-\mathrm{PH}}(\lambda) \leq(2 q+1) \operatorname{Adv}_{\mathcal{B}}^{k-\operatorname{Lin}}(\lambda)+\operatorname{negl}(\lambda)$.

Proof. The proof sketch of Theorem 2 is similar to the proof of our first framework. It still designs a sequence of games which are the same as Table 2 except that $\mathrm{Game}_{4}$ is canceled and there is no need to add $\hat{v}_{k} \mathbf{a}^{\perp}$ in Game ${ }_{2, i, 2}, \mathrm{Game}_{2, i, 3}$ and $\mathrm{Game}_{3}$. Besides, we replace attribute-hiding and attribute-hiding-leakageresilient with $\alpha$-privacy and $\alpha$-privacy-leakage-resilient. We omit details due to the page limitation.

## 5 Instantiations

In this section, we apply our frameworks to the compact-key ABE schemes for zero inner-product and non-zero inner-product in CGW15 and hence obtain several leakage-resilient instantiations.

### 5.1 Instantiation for the First Framework

Zero Inner-product Predicate. Let $\mathcal{X}=\mathcal{Y}:=\mathbb{Z}_{p}^{n}, \mathcal{Z}:=\mathbb{Z}_{p}^{L}, \mathcal{W}:=\mathbb{Z}_{p} \times \mathbb{Z}_{p}^{n} \times$ $\mathbb{Z}_{p}^{L}$, where $n$ is the dimension of vector space. Let $L_{\mathrm{sk}}=(L-2 c-1) \log p$ where $c$ is a fixed positive constant. Pick $(u, \mathbf{w}, \mathbf{u}) \leftarrow \mathcal{W}, \mathbf{z} \leftarrow \mathcal{Z}$, then we have

$$
\begin{array}{ll}
\bullet & \operatorname{rkE}(\mathbf{y}, \mathbf{z}, \alpha):=(\alpha, \mathbf{0}) \in \mathbb{Z}_{p}^{L+1}, \\
\text { • } \operatorname{sE}\left(\mathbf{r}(\mathbf{y}, \mathbf{z},(u, \mathbf{w}, \mathbf{u})):=\left(\mathbf{y}^{\top} \mathbf{w}+\mathbf{z}^{\top} \mathbf{u}, \mathbf{u}\right),\right. \\
& \operatorname{rD}\left(\mathbf{x}, \mathbf{y}, \mathbf{z},\left(d^{\prime}, \mathbf{d}\right):=u \mathbf{x}+\mathbf{w} \in \mathbb{Z}_{p}^{n},\right. \\
\bullet & \bullet \mathrm{sD}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{c}):=\mathbf{c}^{\top} \mathbf{y}
\end{array}
$$

### 5.2 Instantiations for the Second Framework

Zero Inner-product Predicate. Let $\mathcal{X}=\mathcal{Y}:=\mathbb{Z}_{p}^{n}, \mathcal{V}:=\mathbb{Z}_{p}^{(n+1) \times L}, \mathcal{Z}:=$ $\mathbb{Z}_{p}^{L}, \mathcal{W}:=\mathbb{Z}_{p} \times \mathbb{Z}_{p}^{n} \times \mathbb{Z}_{p}^{L}$, where $n$ is the dimension of vector space. Let $L_{\mathrm{mk}}=L_{\mathrm{sk}}=$ $(L-2 c-1) \log p$ where $c$ is a fixed positive constant. Pick $(u, \mathbf{w}, \mathbf{u}) \leftarrow \mathcal{W}, \mathbf{v} \leftarrow$ $\mathcal{V}, \mathbf{z} \leftarrow \mathcal{Z}$. We denote the $i$ 's row vector by $\mathbf{v}_{i-1}^{\top} \in \mathbb{Z}_{p}^{1 \times L}$ for $i=1,2, \ldots, n+1$ and the last n rows by $\overrightarrow{\mathbf{v}} \in \mathbb{Z}_{p}^{n \times L}$, respectively. Define

$$
\begin{array}{ll}
\bullet m \mathrm{mE}(\mathbf{v}, \alpha):=(\alpha, \mathbf{0}) \in \mathbb{Z}_{p}^{n+L+1}, & \bullet \mathrm{mE}(\mathbf{v},(u, \mathbf{w}, \mathbf{u})):=\left(\mathbf{v}_{0}^{\top} \mathbf{u}, \mathbf{w}+\overrightarrow{\mathbf{v}} \mathbf{u}, \mathbf{u}\right), \\
\text { - } \mathrm{rkE}(\mathbf{y}, \mathbf{z}, \alpha):=(\alpha, \mathbf{0}) \in \mathbb{Z}_{p}^{L+1}, & \bullet \mathrm{rE}(\mathbf{y}, \mathbf{z},(u, \mathbf{w}, \mathbf{u})):=\left(\mathbf{y}^{\top} \mathbf{w}+\mathbf{z}^{\top} \mathbf{u}, \mathbf{u}\right), \\
\text { - } \mathrm{sE}(\mathbf{x},(u, \mathbf{w}, \mathbf{u})):=u \mathbf{x}+\mathbf{w} \in \mathbb{Z}_{p}^{n}, & \bullet \mathrm{sD}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{c}):=\mathbf{c}^{\top} \mathbf{y}, \\
\text { - } \mathrm{rD}\left(\mathbf{x}, \mathbf{y}, \mathbf{z},\left(d^{\prime}, \mathbf{d}\right):=d^{\prime}-\mathbf{z}^{\top} \mathbf{d}\right. &
\end{array}
$$

Non-zore Inner-product Predicate. Let $\mathcal{X}=\mathcal{Y}:=\mathbb{Z}_{p}^{n}, \mathcal{V}:=\mathbb{Z}_{p}^{n \times L}, \mathcal{Z}:=$ $\mathbb{Z}_{p}^{L}, \mathcal{W}:=\mathbb{Z}_{p} \times \mathbb{Z}_{p}^{n} \times \mathbb{Z}_{p}^{L}$. Pick $(u, \mathbf{w}, \mathbf{u}) \leftarrow \mathcal{W}, \mathbf{v} \leftarrow \mathcal{V}, \mathbf{z} \leftarrow \mathcal{Z}$. Define

- $\mathrm{mkE}(\mathbf{v}, \alpha):=(\alpha, \mathbf{0}) \in \mathbb{Z}_{p}^{n+L+1}, \quad$ mE $(\mathbf{v},(u, \mathbf{w}, \mathbf{u})):=(u, \mathbf{w}+\mathbf{v u}, \mathbf{u})$,
- $\operatorname{rkE}(\mathbf{y}, \mathbf{z}, \alpha):=(\alpha, \mathbf{0}) \in \mathbb{Z}_{p}^{L+2}$,
- $\mathrm{rE}(\mathbf{y}, \mathbf{z},(u, \mathbf{w}, \mathbf{u})):=\left(u, \mathbf{y}^{\top} \mathbf{w}+\mathbf{z}^{\top} \mathbf{u}, \mathbf{u}\right)$,
- $\mathrm{sD}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{c}):=\mathbf{c}^{\top} \mathbf{y} \cdot\left(\mathbf{x}^{\top} \mathbf{y}\right)^{-1}, \quad \bullet \mathrm{rD}\left(\mathbf{x}, \mathbf{y}, \mathbf{z},\left(d^{\prime}, d, \mathbf{d}\right)\right):=d^{\prime}+d \cdot\left(\mathbf{x}^{\top} \mathbf{y}\right)^{-1}-\mathbf{z}^{\top} \mathbf{d}$,
- $\mathrm{sE}(\mathbf{x},(u, \mathbf{w}, \mathbf{u})):=u \mathbf{x}+\mathbf{w} \in \mathbb{Z}_{p}^{n}$


## Acknowledgments

This work was supported by National Natural Science Foundation of China (61972156, 62372180), NSFC-ISF Joint Scientific Research Program (61961146004), Innovation Program of ShanghaiMunicipal Education Commission (2021-01-07-00-08-E00101) and the "Digital Silk Road" Shanghai International Joint Lab of Trustworthy Intelligent Software (22510750100), University natural science research project in Jiangsu Province (22KJB520035), Open project of "Jiangsu Key Laboratory for Elevator Intelligent Safety" (JSKLESS202104) and Special teaching project of Jiangsu Computer Society (JSCS2022049).

## References

1. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The em side - channel (s). In: International workshop on cryptographic hardware and embedded systems. pp. 29-45. Springer (2002)
2. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Theory of cryptography conference. pp. 474-495. Springer (2009)
3. Attrapadung, N.: Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In: Advances in Cryptology-EUROCRYPT 2014: 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings 33. pp. 557-577. Springer (2014)
4. Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In: 2010 IEEE 51st Annual Symposium on Foundations of Computer Science. pp. 501-510. IEEE (2010)
5. Chen, J., Gay, R., Wee, H.: Improved dual system abe in prime-order groups via predicate encodings. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 595-624. Springer (2015)
6. Crescenzo, G.D., Lipton, R., Walfish, S.: Perfectly secure password protocols in the bounded retrieval model. In: Theory of Cryptography Conference. pp. 225244. Springer (2006)
7. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 621-630 (2009)
8. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: 2008 49th Annual IEEE Symposium on Foundations of Computer Science. pp. 293-302. IEEE (2008)
9. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: cold-boot attacks on encryption keys. Communications of the ACM 52(5), 91-98 (2009)
10. Kalai, Y.T., Reyzin, L.: A survey of leakage-resilient cryptography. IACR Cryptol. ePrint Arch. 2019, 302 (2019)
11. Kim, I., Hwang, S.O., Park, J.H., Park, C.: An efficient predicate encryption with constant pairing computations and minimum costs. IEEE Transactions on Computers 65(10), 2947-2958 (2016)
12. Kocher, P., Horn, J., Fogh, A., Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T., et al.: Spectre attacks: Exploiting speculative execution. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 1-19. IEEE (2019)
13. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 62-91. Springer (2010)
14. Lewko, A., Rouselakis, Y., Waters, B.: Achieving leakage resilience through dual system encryption. In: Theory of Cryptography Conference. pp. 70-88. Springer (2011)
15. Li, J., Yu, Q., Zhang, Y.: Hierarchical attribute based encryption with continuous leakage-resilience. Information Sciences 484, 113-134 (2019)

20 Zhang et al.
16. Li, J., Yu, Q., Zhang, Y., Shen, J.: Key-policy attribute-based encryption against continual auxiliary input leakage. Information Sciences 470, 175-188 (2019)
17. Nishimaki, R., Yamakawa, T.: Leakage-resilient identity-based encryption in bounded retrieval model with nearly optimal leakage-ratio. In: IACR International Workshop on Public Key Cryptography. pp. 466-495. Springer (2019)
18. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Advances in Cryptology-EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24. pp. 457-473. Springer (2005)
19. Wee, H.: Dual system encryption via predicate encodings. In: Theory of Cryptography Conference. pp. 616-637. Springer (2014)
20. Yu, Z., Au, M.H., Xu, Q., Yang, R., Han, J.: Leakage-resilient functional encryption via pair encodings. In: Australasian Conference on Information Security and Privacy. pp. 443-460. Springer (2016)
21. Yuen, T.H., Chow, S.S., Zhang, Y., Yiu, S.M.: Identity-based encryption resilient to continual auxiliary leakage. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 117-134. Springer (2012)
22. Zhang, J., Chen, J., Gong, J., Ge, A., Ma, C.: Leakage-resilient attribute based encryption in prime-order groups via predicate encodings. Designs, Codes and Cryptography 86(6), 1339-1366 (2018)
23. Zhang, L., Zhang, J., Mu, Y.: Novel leakage-resilient attribute-based encryption from hash proof system. The Computer Journal 60(4), 541-554 (2017)
24. Zhang, M., Mu, Y.: Token-leakage tolerant and vector obfuscated ipe and application in privacy-preserving two-party point/polynomial evaluations. The Computer Journal 59(4), 493-507 (2016)

# Constant-Deposit Multiparty Lotteries on Bitcoin for Arbitrary Number of Players and Winners 

Shun Uchizono ${ }^{1}$, Takeshi Nakai ${ }^{2}$, Yohei Watanabe ${ }^{1,3}$, and Mitsugu Iwamoto ${ }^{1}$<br>${ }^{1}$ The University of Electro-Communications, Tokyo, Japan<br>\{uchizono, watanabe, mitsugu\}@uec.ac.jp<br>${ }^{2}$ Toyohashi University of Technology, Aichi, Japan<br>nakai@cs.tut.ac.jp<br>${ }^{3}$ National Institute of Advanced Industrial Science and Technology, Tokyo, Japan.


#### Abstract

Secure lottery is a cryptographic protocol that allows multiple players to determine a winner from them uniformly at random, without any trusted third party. Bitcoin enables us to construct a secure lottery to guarantee further that the winner receives reward money from the other losers. Many existing works for Bitcoin-based lottery use deposits to ensure that honest players never be disadvantaged in the presence of adversaries. Bartoletti and Zunino (FC 2017) proposed a Bitcoin-based lottery protocol with a constant deposit, i.e., the deposit amount is independent of the number of players. However, their scheme is limited to work only when the number of participants is a power of two. We tackle this problem and propose a lottery protocol applicable to an arbitrary number of players based on their work. Furthermore, we generalize the number of winners; namely, we propose a secure $(k, n)$ lottery protocol. To the best of our knowledge, this is the first work to address Bitcoin-based ( $k, n$ )-lottery protocol. Notably, our protocols maintain the constant deposit property.


Keywords: Secure lottery • Bitcoin • Fairness • Elimination tournament.

## 1 Introduction

### 1.1 Backgrounds

Consider a bet in which each of the $n$ players gambles one dollar. The champion is randomly chosen from them and he/she receives the sum of the bets, $n$ dollars, as a reward. Secure lottery is a cryptographic protocol that allows us to play such games fairly $[12-14,18]$. That is, it ensures that no honest player is disadvantaged in the presence of adversarial players who do not follow procedures.

One of the crucial issues in constructing a secure lottery is how to deal with the abort attack, which terminates in the middle of a protocol to avoid losing. To counter the attack, we must enforce an adversary to tell the lottery result to all honest parties. Such a property is typically defined as fairness, which ensures
that at the end of a protocol, either all parties learn the output or none of them learn it. Unfortunately, it is known that fairness cannot be achieved without any additional assumption such as the honest majority or trusted third parties [11].

Another fundamental challenge is how to force losers to pay winners. Since a typical cryptographic protocol treats no monetary entity, we cannot require a protocol to guarantee such a property. To enforce the payoff, we need to introduce a setup for handling monetary operations, e.g., a trusted bank [19, 22], e-cash $[6,9,17]$, or decentralized cryptocurrency.

Secure lottery based on cryptocurrency. Using cryptocurrency, e.g., Bitcoin [23] and Ethereum [24], we can construct a secure lottery protocol that forces losers to pay winners without relying upon any trusted third party even in the dishonest majority. Informally, in cryptocurrency-based protocols, parties cooperate to create some transactions at the beginning of the protocol and deposit or bet money. One of the transactions is corresponding to $n$ dollars, the prize money. If a protocol guarantees that only the winner can learn the witness to redeem it, it implies that only the winner can receive the prize.

There is a line of work on achieving a variant of fairness using monetary penalties. The monetary penalty enforces adversaries to follow procedures to avoid losing money, and it allows us to achieve fairness. In secure multi-party computation, many works adopt such a definition, e.g., $[5,7,8,15,16,21]$.

Similarly, it is known that monetary penalties enable us to construct a secure lottery protocol even in the dishonest majority. Back and Bentov [2] showed a secure lottery based on Bitcoin in the two-party setting. Their protocol can enforce a payment from the loser to the winner. Moreover, it guarantees that an aborting party loses money and then another party obtains money as compensation. Afterward, Andrychowicz, Dziembowski, Malinowski, and Mazurek [1] and Bentov and Kumaresan [2], respectively, proposed Bitcoins-based secure lottery protocols that can be applied to an arbitrary number of parties.

In many works of Bitcoin-based secure lotteries, parties must input deposit to achieve fairness in addition to the bet. Indeed, the existing protocol made of Marcin [1] requires parties to input $O\left(n^{2}\right)$ deposits, where $n$ is the number of parties. The deposit is guaranteed to be returned to every honest party at the end of the protocol. On the other hand, for adversarial parties, the deposit is not returned to them but is instead distributed to honest parties as compensation. Even though the protocol promises to refund deposits to honest parties, it is undesirable to require money other than bets. That is to say, too expensive deposits make it difficult for parties to participate in the protocol. Based on the backgrounds, Bartoletti and Zunino [4] proposed a secure lottery protocol with a constant deposit. Independently, Miller and Bentov [20] proposed a secure lottery without any deposit money. However, as pointed out by Bartoletti and Zunino [4], their scheme has an issue of depending on a Bitcoin specific opcode, MULTIINPUT. To be a generic scheme, it should not rely on a custom scripting language supported by a particular blockchain.

In this paper, we focus on Bartoletti-Zunino work [4]. Informally they realize a constant-deposit protocol based on a single-eliminate tournament, i.e., they use multiple matches between two players to determine one winner of the lottery. However, their protocol assumes that the tournament has a complete binary tree structure. In other words, it has an issue to be applicable only if the number of participants can be expressed in $2^{L}$, where $L$ is a positive integer that refers to the tree depth.

### 1.2 Our Contribution

This paper presents two contributions. The first one is to solve the issue of the restriction of the number of participants in the Bartolotti and Zunino scheme. That is, we propose $(1, n)$-lottery protocol for an arbitrary positive integer $n$. Our construction idea is we bias the winning percentage for each match to ensure that all participants are equal even if the tournament is not a complete binary tree.

The second contribution is to generalize the number of winners, namely, we propose a ( $k, n$ )-lottery protocol for arbitrary $k$ and $n$. To realize the protocol, we first construct ( $k, k+1$ )-lottery protocol. Our $(k, n)$-lottery protocol is derived from a composition of $(k, k+1)$-lottery protocols. More precisely, in our protocol parties first run $(n-1, n)$-lottery protocol and determine one loser. Afterward, $n-1$ winners run ( $n-2, n-1$ )-lottery protocol and further determine one loser. Players repeat such a process until deciding $n-k$ losers. To the best of our knowledge, this is the first work to realize $(k, n)$-lottery protocol based on Bitcoin with a constant deposit.

### 1.3 Basic Notations

For any positive integer $i$, let $[i]:=\{0,1, \ldots, i-1\}$. We denote by $\eta$ a security parameter. We suppose that all players are probabilistic polynomial-time algorithms (PPTA) in a security parameter $\eta$.

We construct lottery protocols based on a tournament structure represented as a binary tree, as in [4]. Hereafter, we call a champion to distinguish it from the winners of matches in the tournament. In a binary tree, its leaf nodes refer to players, and the other nodes represent a match (or the winner) of two child nodes. Each node at level $l$ in the tree is identified as a $(l+1)$-bit string. For a node $\pi$, we denote its child nodes as $\pi_{\text {left }}=\pi \| 0$ and $\pi_{\text {right }}=\pi \| 1$. Namely, $\pi$ is the prefix of its child nodes. We write $\pi \sqsubset \pi^{\prime}$ if $\pi$ is a prefix of $\pi^{\prime}$. We note that, since we handle an arbitrary number of players, the tournaments may not be the complete binary tree. Hence, the binary tree in our protocol is represented by $\Pi \subseteq\left\{\{0,1\}^{l} \mid 1 \leq l \leq L\right\}$, where the tree has $L$ levels. We denote by $P$ the set of players. Note that, since the players correspond to leaf nodes, it holds $P \subset \Pi$. For a bit string $\pi,|\pi|$ means the bit length of $\pi$. We denote by $\pi_{r}$ the root node of a binary tree.
Organization. As a preparation for the introduction of our protocol, we first describe a bitcoin overview in Section 2. Section 3 presents several notations and
useful lemmas regarding tournament structures. In Section 4, we define secure lottery protocol. We show our constructions for $(1, n)$-lottery and $(k, n)$-lottery in Sections 5 and 6, respectively. In these sections, we prove security of the protocols according to the security definitions, shown in Section 4.

## 2 A Brief Introduction to Bitcoin

In a blockchain protocol, parties maintain a global ledger that holds ordered sets of records, i.e., blocks. To append a new block to the blockchain, parties must race and win to solve a cryptographic puzzle, as known as the mining process. The puzzle hardness is parameterized so that the intervals between the growth of blocks are approximately constant at a particular time (about 10 minutes in Bitcoin). Since each block contains a cryptographic hash function of the previous block, the state of each block is preserved by subsequent blocks. Furthermore, when the blockchain diverges into multiple states, proper parties accept the longest chain. Hence, if an adversary tries to rewrite data contained in a block, it needs to reconstruct the subsequent blocks in addition to the block. The adversary must further make the rewritten chain the longest to get other parties to accept it. However, it is infeasible unless the adversary possesses more than half the computing power of the entire network. That is, a blockchain realizes a tamper-resistant public bulletin board based on the assumption about the computing power of adversaries $[3,10]$.

Bitcoin is a decentralized cryptocurrency based on a blockchain. The Bitcoin ledger manages transactions on its blocks. Roughly speaking, a transaction $\mathrm{T}_{1}$ refers to a sender, the amount transferred coins, and the recipient, i.e., it expresses information about "a sender $S$ sends $Q$ coins to a recipient $R$." The party $R$ can send $Q$ coins to the other party by making a new transaction $\mathrm{Tx}_{2}$ that refers to $T \mathrm{x}_{1}$. Then, $\mathrm{T} \mathrm{x}_{1}$ becomes a spent transaction and thereafter $R$ cannot re-use it. We can check the balance of a party by referring to all unspent transactions corresponding to the party on the blockchain.

Precisely, a transaction form has inputs and outputs. An input specifies a transaction to be used for this remittance. In the above example, the input of $\mathrm{T}_{2}$ is $\mathrm{T}_{1}$ ('s output). An output (script) specifies the recipient by describing a condition to use the transaction. Typically, the output script contains a signature verification with a public key of the recipient. When a party uses a transaction, he/she needs to write a witness on the transaction as an input script that satisfies the output script of the input transaction. See Fig. 1 that shows the transaction flow in the simplest case. Transaction $T x_{2}$ redeems the previous transaction $T x_{1}$ to use $\$ v$. Then, witness $w_{1}$ written in the input script of $\mathrm{Tx}_{2}$ must satisfy the condition $\phi_{1}$, which is the output script of $\mathrm{T}_{1}$. Similarly, to use $\$ v$ with reference to $\mathrm{Tx}_{2}$, it is necessary to create a transaction that holds $w$ in its input script such that $\phi_{2}(w)=1$. Hereafter, in the graphical description, an arrow connects the corresponding input and output.

In Bitcoin, by specifying some transactions as inputs, a party can create a transaction to transfer the sum of the coins. Similarly, a single transaction

| $\mathrm{Tx}_{1}$ | $\mathrm{Tx}_{2}$ |
| :--- | :--- | :--- |
| in: $\mathrm{Tx}_{0}$ <br> inscript $: w_{0}$ |  |
| outscript $(x): \phi_{1}(x)$ <br> value: $\$ v$ | in: $\mathrm{Tx} x_{1}$ <br> inscript $: w_{1}$ s.t. $\phi_{1}\left(w_{1}\right)=1$ |
| outscript $(x): \phi_{2}(x)$ <br> value: $\$ v$ |  |

Fig. 1. Graphical description of a transaction flow.
can specify multiple recipients by holding multiple output scripts. Formally, we denote a $m$-input and $l$-output transaction in Bitcoin by

$$
\text { (in }[m] \text {, inscript }[m] \text {, value }[l] \text {, outscript }[l] \text {, lockTime), }
$$

where $\operatorname{in}[i]$ is an identifier of the input transaction (i.e., the previous one), inscript $[i]$ is the corresponding input script (i.e., a witness), value $[i]$ is the number of coins, outscript $[i]$ refers to the corresponding output script, and lockTime specifies the earliest time when the transaction appears on the ledger. Namely, the miners do not approve the transaction until the time specified by lockTime. Note that the sum of the input coins must match the sum of the output coins.

A transaction excluding the input script (in $[m]$, value $[l]$, outscript $[l]$, lockTime) is called the simplified form. Typically, as described above, the output script contains a signature verification algorithm to specify the recipient. The input script of the next transaction states a signature in its simplified form in order to prove the creator is the specified recipient.

## 3 Tournaments with Uniform Winning Probability

### 3.1 Tournaments with a Single Champion

First, we discuss the case where the champion is only one. In cases of tournaments based on complete binary trees, it is obvious that every party has an equivalent chance to be champion by equating win probabilities of all matches by $1 / 2$. On the other hand, if it is not a complete binary tree, i.e., the number of matches differs from player to player, then it is necessary to bias the winning probabilities to make the tournament equal for all players. We here present several useful lemmas to make fair tournaments even in such cases. (We show the proofs in Appendix A.)

Let us consider a tournament that may not be a complete binary tree and consider the biased probabilities of each match to make it fair. Suppose a match $\pi$ of which child nodes are $\pi_{\text {left }}$ and $\pi_{\text {right }}$. We consider two subtrees such that its root nodes are $\pi_{\text {left }}$ and $\pi_{\text {right }}$, and let $v_{\text {left }}^{\pi}$ and $v_{\text {right }}^{\pi}$ be the number of leaf nodes in these subtrees, respectively. (Note that, if the entire tournament form
is the complete binary tree, then $v_{\text {left }}^{\pi}=v_{\text {right }}^{\pi}$ always holds.) Based on the above notations, for a node $\pi$, we define $\operatorname{Biased} \operatorname{Pr}(\pi):=v_{\text {left }}^{\pi} /\left(v_{\text {left }}^{\pi}+v_{\text {right }}^{\pi}\right)$. We can construct a fair tournament based on any binary tree using this function from the following lemma.

Lemma 1 For any tournament consisting of a binary tree, if the winning probabilities of each match $\pi$ is set with $(\operatorname{Biased} \operatorname{Pr}(\pi), 1-\operatorname{Biased} \operatorname{Pr}(\pi))$, then the tournament is equal for every player.

### 3.2 Tournaments with Multiple Champions

Next, we discuss the case of multiple champions. In this case, we must pay attention to the joint winning probabilities of each set of players not only to the winning probability of individual players. For instance, in order to choose two champions, consider the case of dividing the players half into two groups and running a single champion tournament in each group. In this case, although each player has the same probability of being champion, the problem arises that players in the same group can never win simultaneously. To tackle this issue and deal with an arbitrary number of winners, we construct $(k, k+1)$-lottery protocol. Thus, we first discuss a single eliminate tournament that determines $k$ champions from $k+1$ players, $(k, k+1)$-tournament. Afterward, we show that tournaments applicable to an arbitrary number of champions can be constructed by combining multiple ( $k, k+1$ )-tournaments.

To construct a $(k, k+1)$-tournament, we adopt the single-elimination tournament proceeding as follows: First two players $p_{1}$ and $p_{2}$ play a match $\pi_{b}$. The winning player is determined to be a champion, and the loser $l_{1} \in\left\{p_{1}, p_{2}\right\}$ plays the next match $\pi_{2}$ with $p_{2}$. In a similar way, for $i=1 \ldots k-1$, player $p_{i+1}$ and the previous match loser $l_{i-1}$ plays a match $\pi_{i}$. The loser of $(k-1)$-th match $\pi_{k-1}$ becomes the only loser of the tournament.

Lemma 2 If the winning probabilities of match $\pi_{i}$ between $l_{i-1}$ and $p_{i+1}$ is set with $(i /(i+1), 1-i /(i+1))$ for $i=1 \ldots k-1$, then the tournament is equal for every player. Moreover, for any subset $S \subset P$ such that $|S|=k$, the probability of winning the parties in $S$ simultaneously is also equivalent.

We can construct a $(k, n)$-tournaments for arbitrary $k$ and $n$ by running ( $n-j, n-j+1$ )-tournament for $j=1 \ldots n-k$. Concretely, the winners of ( $n-j^{\prime}, n-j^{\prime}+1$ )-tournament continue the next $\left(n-j^{\prime}-1, n-j^{\prime}\right)$-tournament to further determine one loser, and the players continue such process until the remaining winners are $k$ players.

Lemma 3 For any positive integers $k$ and $n$ with $k<n$, if $a(k, n)$-tournament is composed of sequential executions of $(n-j, n-j+1)$-tournament for $j=$ $1 \ldots n-k$, then the probability of being the winner of a tournament is equivalent for every player. Moreover, for any subset $S \subset P$ such that $|S|=k$, the probability of winning the parties in $S$ simultaneously is also equivalent.

## 4 Definition of Secure Lottery Protocol

Suppose a game in which each of $n$ player bets $\$ \alpha$. A secure $(k, n)$-lottery protocol is a cryptographic protocol to $k$ champions who obtain $\$(n \alpha / k)$ from them fairly. This section presents the security model of this protocol.

Hereafter, we say that player $p$ can freely redeem transaction $\mathrm{Tx}_{\mathrm{x}}$ if $p$ holds a witness that satisfies the output script of Tx . Let wealth of player $p$ at round $t$ mean the total amount of coins in transactions such that $p$ can freely redeem at round $t$. Note that we ignore coins not involved in the protocol. Also, payoff of player $p$ refers to the difference between wealth at the beginning of the protocol and at the end.

Before presenting formal descriptions, we discuss an intuitive understanding of security requirements. First, we focus on the case of $k=1$, i.e., the champion is only one. As a premise, if all players behave honestly, it is necessary to determine the champion uniformly at random. Of course, it is ideal to achieve this property even in the presence of an adversary. However, such a requirement is somewhat too strong to achieve. For instance, an adversary may abort early after the start of a protocol. In this case, since the protocol terminates without determining the champion, it does not fulfill the condition of determining the champion uniformly at random. Thus, in the case where corrupted players exist, we relax the requirement. More concretely, a secure protocol ensures that the expected value of honest parties' payoffs is never negative for the arbitrary strategy of the adversary.

In the case of $k \geq 2$, the requirements are almost similar to the above, however, there is one additional condition that comes from having multiple champions. We require that, if all players are honest, for any set of players $W \subset P$ such that $|W|=k$, the probability that $W$ becomes champions is the same. In other words, it ensures that not only the tournament is fair for individual players, but also is equal for each set of players. It is also necessary that, if an adversary violates this property, its expected payoff becomes negative. This requirement means that adversaries cannot prevent a certain set of players from becoming champions simultaneously without loss.

To capture the above requirements formally, we introduce several notations. Let $\sigma_{\mathcal{A}}$ denote a strategy set of a PPTA adversary $\mathcal{A}$, and let $s t_{0}$ denote the ledger state at the beginning of the protocol. We denote by $\Omega\left(p, s t_{0}, t, \sigma_{\mathcal{A}}\right)$ a random variable of wealth of player $p$ at round $t$. In the case where there is no corrupted party, we describe $\sigma_{\mathcal{A}}=\perp$. Let $\beta$ and $\epsilon$ denote the round number at the beginning and at the end of the protocol, respectively. We define a random variable with respect to payoff as follows.

$$
\begin{equation*}
\Phi\left(p, s t_{0}, \sigma_{\mathcal{A}}\right)=\Omega\left(p, s t_{0}, \epsilon, \sigma_{\mathcal{A}}\right)-\Omega\left(p, s t_{0}, \beta, \sigma_{\mathcal{A}}\right) \tag{1}
\end{equation*}
$$

We denote by $E\left(\Phi\left(p, s t_{0}, \sigma_{\mathcal{A}}\right)\right)$ the expected value of the payoff.
Definition 1 We say a (1,n)-lottery protocol $\Pi$ is secure if $\Pi$ fulfills the followings except a negligible probability in $\eta$ :

- If all players are honest, $E\left(\Phi\left(p, s t_{0}, \perp\right)\right)=0$ and $\Omega\left(p, s t_{0}, \epsilon, \perp\right) \in\{-\alpha, \alpha(n-$ 1) $\}$ for all $p \in P$.
- For all PPTA adversaries $\mathcal{A}$, i.e., if there exist corrupted players, $E\left(\Phi\left(p, s t_{0}, \sigma_{\mathcal{A}}\right)\right) \geq$ 0 holds for all $p \in H$.

Definition 2 We say a $(k, n)$-lottery protocol $\Pi$ is secure if $\Pi$ fulfills the followings except a negligible probability in $\eta$ :

- If all players are honest, $E\left(\Phi\left(p, s t_{0}, \perp\right)\right)=0$ and $\Omega\left(p, s t_{0}, \epsilon, \perp\right) \in\{-\alpha,(\alpha / k)(n-$ $k)\}$ for all $p \in P$. Furthermore, $\operatorname{Pr}\left(\sum_{s \in S} \Omega\left(s, s t_{0}, \beta, \perp\right)=k(n-k)\right)=\binom{n}{k}^{-1}$ for all $S=\left\{s_{1}, \ldots, s_{k}\right\} \subset P$.
- For any PPTA adversary $\mathcal{A}, E\left(\Phi\left(p, s t_{0}, \sigma_{\mathcal{A}}\right)\right) \geq 0$ holds for all $p \in H$.
- For any PPTA adversary $\mathcal{A}$, if there exists $S \subseteq H$ such that $|S| \leq k$ and $\operatorname{Pr}\left(\sum_{s \in S} \Phi\left(s, s t_{0}, \sigma_{\mathcal{A}}\right)=|S|(n-k)\right) \neq\binom{ n-|S|}{k-|S|}^{-1}$, the protocol guarantees that $\sum_{p \in C} E\left(\Phi\left(p, s t_{0}, \sigma_{\mathcal{A}}\right)\right)<0$.

To achieve a secure protocol, we require players to input deposit in addition to the bets. The deposits play a roll of compensation for honest players when an adversary behaves maliciously. We say that a protocol is constant-deposit if the deposit amount of every player is a constant value independent from the number of players.

## 5 (1, $n$ )-Lottery Protocol with Constant Deposits

This section presents a $(1, n)$-lottery protocol for an arbitrary positive integer $n$. We suppose that a bet amount of each party is $\alpha=1$. Our protocol is based on single-elimination tournaments with binary tree structure. The tournament consists of $n-1$ two-player matches: the winners of the matches at level $l \in[L]$ play at the next level $l-1$, where $L$ is the tree depth. The winner of the match at level 0 obtains $\$ n$ as a reward. Bartoletti and Zunino's protocol set the winning probability to $1 / 2$ in each match. To construct a protocol for an arbitrary number of players, it is necessary to modify it so that all players are fair to win even if the tournament is not the complete binary tree. The main idea of our protocol is to bias the probability of winning in each match.

### 5.1 Building Block: Biased Coin-Tossing Protocol

We denote with $\tau_{\text {Ledger }}$ the sufficient time to write a transaction on the ledger and confirm it. (It is about 60 minutes in Bitcoin.) We denote by $K_{p}(\mathrm{~T} \times, \pi, \mathcal{P})$ a key pair of player $p$ for transaction Tx , which corresponds to a match $\pi$. $\mathcal{P}$ refers to players' identifiers corresponding to the match. We suppose that the private part of key pairs is kept secret by $p$. (Note that we write signing and verification keys without distinguishing between them.) We define $\mathbf{K}(\mathrm{Tx}, \pi, \mathcal{P}):=$ $\left\{K_{p}(\mathrm{~T} \times, \pi, \mathcal{P}) \mid p \in P\right\}$.

Let (ver, sig) be a signature scheme. Following Bartoletti and Zunino's work, we allow the partial signature that enables to exclude of the input field from
the signature subjects. It allows to generate a signature on a transaction before determining the input field of the transaction. Namely, we use the malleability of input fields. Hereafter, a signature written in the input field of transaction $\mathrm{Tx}=$ (in $[m]$, inscript $[m]$, value $[l]$, outscript $[l]$, lockTime) is for (value $[l]$, outscript $[l]$, lockTime). Below, we omit the inputs of signatures and refer to it as $\operatorname{sig}_{K_{p}\left(T_{x}, \pi, \mathcal{P}\right)}$. Also, $\operatorname{sig}_{\mathbf{K}\left(\mathrm{T}_{x}, \pi, \mathcal{P}\right)}$ means the multi-signature with $\mathbf{K}(\mathrm{Tx}, \pi, \mathcal{P})$.

As described the previous section, we construct a protocol based on a tournament structure. Thus, before presenting our lottery protocol, we show a protocol to realize a match between two parties. Since we deal with tournaments not the complete binary tree, it is necessary to bias some matches to ensure all players to have the same probability of winning the tournament. Hence, we construct a match protocol, called a biased coin tossing protocol, that can parameterize the winning probability.

To handle biased probabilities, we introduce a winner function to determine the winner in a match. Let $a$ and $b$ be players that hold secrets $s_{a}$ and $s_{b}$, respectively. We consider a match such that the winner depends on $s_{a}$ and $s_{b}$, and define the function to determine the winner as follows.

$$
\text { Winner }\left(s_{a}, s_{b}, v_{a}, v_{b}\right)= \begin{cases}a & \text { if } s_{a}+s_{b}\left(\bmod v_{a}+v_{b}\right)<v_{a}  \tag{2}\\ b & \text { otherwise }\end{cases}
$$

where $v_{a}$ and $v_{b}$ are positive integers. Hereafter, we suppose that $s_{a}$ and $s_{b}$ are sampled from $\left[v_{a}+v_{b}\right]$ uniformly at random. ${ }^{4}$ The output $x \in\{a, b\}$ means the winner of the match.

See Protocol 1 and Fig. 2 that shows a protocol of realizing a match $\pi_{i}$ in a tournament. (Suppose $\pi_{a}$ and $\pi_{b}$ be the child nodes of $\pi_{i}$.) A match consists of three types of transactions, Win, Turn1, and Turn2. At the beginning of the protocol, suppose $\operatorname{Win}\left(\pi_{a}, a\right)$ and $\operatorname{Win}\left(\pi_{b}, b\right)$ being on the ledger, which implies that player $a$ and $b$ won the previous matches $\pi_{a}$ and $\pi_{b}$, respectively. Now, they play a match $\pi_{i}$. Turn1 is used to aggregate the coins of $\operatorname{Win}\left(\pi_{a}, a\right)$ for the preparation of the match. Turn2 is a transaction of which input is Turn1. See the output script of Turn1. To redeem Turn1, a player must write $s_{a}$ on the input script of Turn2. Thus, putting Turn2 on the ledger implies to reveal $a$ 's secret $s_{a}^{\pi_{i}} . \operatorname{Win}\left(\pi_{i}, a\right)$ and $\operatorname{Win}\left(\pi_{i}, b\right)$ are transactions of which input is Turn2. See the output script of Turn2. To redeem Turn2, a player must write $s_{a}^{\pi_{i}}$ and $s_{b}^{\pi_{i}}$ on the input script of the next transaction. Thus, to redeem Turn2, player $b$ must reveal his/her secret $s_{b}^{\pi_{i}}$. Furthermore, since $s_{a}^{\pi_{i}}$ and $s_{b}^{\pi_{i}}$ satisfy either $a=\operatorname{Winner}\left(s_{a}^{\pi_{i}}, s_{b}^{\pi_{i}}, v_{a}, v_{b}\right)$ or $b=\operatorname{Winner}\left(s_{a}^{\pi_{i}}, s_{b}^{\pi_{i}}, v_{a}, v_{b}\right)$, players can put only one of $\operatorname{Win}\left(\pi_{i}, a\right)$ and $\operatorname{Win}\left(\pi_{i}, b\right)$ on the ledger. The transaction put on the ledger refers to the winner of this match and is used as the input of Turn1 in the next match.

[^13]

Fig. 2. Graphical description of biased coin-tossing (for match $\pi_{i}$ ).

### 5.2 Our Construction of $(1, n)$-lottery

The biased probability of each match in $(1, n)$-Lottery. First, we present the biased probability of each match. Let us consider a match $\pi$ of which child nodes $\pi_{a}$ and $\pi_{b}$. As in Section 3.1, we consider two subtrees such that its root nodes are $\pi_{a}$ and $\pi_{b}$, and let $v_{a}^{\pi}$ and $v_{b}^{\pi}$ be the number of leaf nodes in these subtrees, respectively. From Lemma 1, we set the winner function in each match $\pi$ of our ( $1, n$ )-lottery protocol as Winner $\left(s_{a}^{\pi}, s_{b}^{\pi}, v_{a}^{\pi}, v_{b}^{\pi}\right)$.

Our protocol is applicable to an arbitrary binary tree. Let $\Pi \subseteq\left\{\{0,1\}^{n} \mid 1 \leq\right.$ $n \leq L\}$ be a binary tree applied to our protocol, and it has $L$ levels. Based on the binary tree and the biased probability, our protocol proceeds as follows.

Precondition: For all $p \in P$, the ledger contains a transaction $\operatorname{Bet}_{p}$ with value $\$(1+d)$, and redeemable with key $K_{p}\left(\operatorname{Bet}_{p}\right)$.

## Initialization phase:

1. For all player $p \in P, p$ generates the following secret keys locally. Each player $p$ generates all the following key pairs.

- For all $\pi$ such that $\pi$ is leaf and every $p \in P$ :
$K_{p}\left(\operatorname{Bet}_{p}\right), K_{p}($ CollectW $), K_{p}($ Init, $a)$
- For all $\pi$ and every $p \in P$ :
$K_{p}(\mathrm{Win}, \pi, a), K_{p}(\mathrm{WinTo}, \pi, a)$
- For all $\pi$ such that $\pi$ is neither leaf nor root and every $a, b \in P$ such that $a, b \sqsubset \pi$ :
$K_{p}($ Turn1To $, \pi, a, b), K_{p}($ Turn1, $\pi, a), K_{p}($ Turn2To $, \pi, a, b), K_{p}($ Turn2, $\pi, a)$,
$K_{p}($ Timeout $1, \pi, a, b), K_{p}($ Timeout2, $\pi, a, b)$


## Protocol 1 Biased Coin-Tossing $\Pi_{a, b}^{\mathrm{W}}\left(s_{a}, s_{b}, v_{a}, v_{b}\right)$

## Setup:

1: The initialization phase was completed, and $\operatorname{Win}(\pi, a)$ and $\operatorname{Win}(\pi, b)$ have been put already on the ledger. Players $a$ and $b$ hold secrets $s_{a}$ and $s_{b}$, respectively. Let $\tau$ be the round of the beginning of the protocol.

## Procedure:

2: One of the players puts $\operatorname{Turn} 1(\pi, a, b)$ on the ledger.
3: $a$ writes $s_{a}$ on the input script of Turn2 $(\pi, a, b)$, and put the transaction on the ledger.
if Turn2 $(\pi, a, b)$ does not appear within $\tau+2 \tau_{\text {Ledger }}$ then
One of the players puts $\operatorname{Timeout} 1(\pi, a, b)$ on the ledger.
6: One of the players puts $\operatorname{Win}(\pi, b)$ on the ledger.
$b$ computes $w=\operatorname{Winner}\left(s_{a}, s_{b}, v_{a}, v_{b}\right)$
if $w=a$ then
$b$ puts $\operatorname{Win}(\pi, a)$ on the ledger.
if $w=b$ then
$b$ puts $\operatorname{Win}(\pi, b)$ on the ledger.
if $\operatorname{Win}(\pi, x \in\{a, b\})$ does not appear within $\tau+4 \tau_{\text {Ledger }}$ then
One of the players puts Timeout2 $(\pi, a, b)$ into the ledger.
14: One of the players puts $\operatorname{Win}(\pi, a)$ on the ledger.
2. For all player $p \in P, p$ generates $\operatorname{secrets} s_{p}^{\pi_{p}}$ for each $\pi_{p}$, such that $\left(\left|\pi_{p}\right|<L\right)$, and broadcasts to the other players his/her public keys and hashes $h_{p}^{\pi_{p}}=H\left(s_{p}^{\pi_{p}}\right)$.
3. If $h_{p}^{\pi_{p}}=h_{p^{\prime}}^{\pi_{p}{ }^{\prime}}$ for some $\left(p, \pi_{p}\right) \neq\left(p^{\prime}, \pi_{p}{ }^{\prime}\right)$, the players abort.
4. Parties agree the time $\tau_{\text {Init }}$ large enough to fall after the initialization phase.
5. Each player signs all transaction templates in Fig. 3 except for Init, and broadcasts the signatures.
6. Each player verifies the signatures received by the others. some signature is not valid or missing, the player aborts the protocol.
7. Each player signs Init, and sends the signature to the first player.
8. The first player puts the (signed) transaction Init on the ledger.
9. If Init does not appear within one $\tau_{\text {Ledger }}$, then each $p$ redeems $\operatorname{Bet}_{p}$ and aborts.
10. The players put the signed transactions $\operatorname{Win}(p, p)$ on the ledger, for all $p \in P$.
Tournament execution phase: For all levels $l=L \ldots 1$, players proceed as
follows: Run $\Pi_{a, b}^{\mathrm{W}}\left(s_{a}^{\pi_{a}}, s_{b}^{\pi_{b}}, v_{a}^{\pi_{a}}, v_{b}^{\pi_{b}}\right)$ for each $\pi$, such that $|\pi|=l-1$, in parallel. Then, $v_{a}^{\pi_{a}}, v_{b}^{\pi_{b}}$ denote the biased probability determined in the manner shown in the above. ${ }^{5}$
Garbage collection phase: If there is some unredeemed $\operatorname{Win}(\pi, p)$ such that $\pi$ is not the root on the ledger, players put CollectOrphanWin $(\pi, p)$ on the

[^14]ledger. (If all players behave honestly, this step is not carried out. It is a countermeasure for the transaction insertion attack, shown in Section 5.3.)
At step 2, players prepare all transactions that may be used in the protocol. Note that they then signs the transactions using signing keys of all players. Thus, after this step, it is not possible for some players to collude and forge transactions, except for input and input script fields. The number of transactions created in this step is $O\left(n^{2}\right)$, which is derived from the number of possible match combinations. See $\operatorname{Win}\left(\pi_{r}, a\right)$ in Fig. 3 that is a transaction for the champion. At the end of the tournament execution phase, only the champion is freely redeemable a $\operatorname{Win}\left(\pi_{r}, a\right)$ and can obtain $\$(n+d)$, which is the reward and deposit for the champion. Furthermore, $\operatorname{Win}\left(\pi_{r}, a\right)$ holds outputs to return deposits for each player.

### 5.3 Transaction Insertion Attack

In our scheme, as in the Bartoletti-Zunino scheme, an adversary can turn an honest player who should be the winner into the loser in a match. The details of the attack are described below.
Settings. Consider a match $\pi$ with honest player $a$ and malicious player $b$, where they are winners of the previous matches $\pi_{0}$ and $\pi_{1}$, respectively. Let player $c$ be the loser of $\pi_{0}$, and let $\pi^{\prime}$ be the parent node of $\pi$. Player $b$ has a freely redeemable transaction $T_{b}$ with $\$\left(v_{a}^{\pi}+v_{b}^{\pi}\right)(1+d)$ in the external to the protocol. Procedures. Suppose when honest player $a$ puts Turn2 $(\pi, a, b)$ on the ledger in the biased coin tossing protocol for $\pi$, player $b$ realizes that he has lost the match. Then, $b$ redeems $T_{b}$ through a transaction $\operatorname{Win}(\pi, b)$ by malleating its input and input script fields. (Note that in our scheme, we assume the input malleability.) Player $b$ can now redeem both his transaction and $\operatorname{Win}\left(\pi^{\prime}, c\right)$ by putting Turn1 $\left(\pi_{1}, b, c\right)$ on the ledger. Player $a$ can redeem the pending Turn2 $(\pi, a, b)$ (after its timeout has expired) using $\operatorname{Timeout} 2(\pi, a, b)$, and then redeem that with $\operatorname{Win}(\pi, a)$. This transaction is now orphan, i.e. it can no longer be used in the next rounds because its $\operatorname{Win}\left(\pi^{\prime}, c\right)$ was already redeemed by $b$. However, the orphan transaction can be redeemed in the garbage collection phase by CollectW $(\pi, a)$. Thus, player $a$ can collect $\$\left(v_{a}^{\pi}+v_{b}^{\pi}\right)(1+d)$ at the garbage collection phase.

As shown above, in order to realize this attack in match $\pi$, an adversary needs to invest additional coins $\$\left(v_{a}^{\pi}+v_{b}^{\pi}\right)(1+d)$ into the protocol. The affected honest player can collect $\$ d$ by the root $\operatorname{Win}\left(\pi_{r}, a\right)$ and $\$\left(v_{a}^{\pi}+v_{b}^{\pi}\right)(1+d)$ by the garbage collection. Informally, this adversarial scenario does not affect the security since the honest player who is applied this attack would rather gain due to the deposit. We present security proof of our protocol in the next subsection.

### 5.4 Security Proof

This section shows security proof of our ( $1, n$ )-lottery protocol. Our proof is based on the fact that the possible attack strategies for adversaries is only the transaction insertion attack or the rejection of revealing their secrets.

| Init |
| :--- |
| $\operatorname{in}[p]: \operatorname{Bet}_{p}$ <br> $\operatorname{inscript}[p]: \operatorname{sig}_{K_{p}\left(\operatorname{Bet}_{p}\right)}$ <br> outscript $[p](\mathrm{T}, \sigma): \operatorname{ver}_{\mathrm{K}(\text { Init, } p)}(\mathrm{T}, \sigma)$ <br> value $[p]: \$(1+d)$ |


| $\mathrm{Win}(\pi, a) \quad\left(\pi \in \Pi^{\prime}, \pi \sqsubset a\right)$ |
| :---: |
| in: Timeout1 $(\pi, b, a)$ <br> inscript: $\operatorname{sig}_{\mathbf{K}(\text { Timeout } 1, \pi, b, a)}$ |
| in: Timeout2 ( $\pi, a, b$ ) <br> inscript : $\operatorname{sig}_{\mathbf{K}_{(\text {Timeout } 2, \pi, a, b)}}$ |
| in: Turn2 $(\pi, a, b)$ <br> inscript : $s_{a}^{\pi}, s_{b}, \operatorname{sig}_{\mathbf{K}_{(\text {Turn } 2, \pi, a)}}$ |
| in: Turn2 $(\pi, b, a)$ <br> inscript : $s_{a}^{\pi}, s_{b}, \operatorname{sig}_{\mathbf{K}_{(\text {Turn2 } 2, \pi, a)}}$ |
| $\begin{gathered} \text { outscript }(\mathrm{T}, \sigma): \operatorname{ver}_{\mathbf{K}(\operatorname{Win}, \pi, a)}(\mathrm{T}, \sigma) \\ \operatorname{Vver}_{\mathbf{K}(\operatorname{WinTO}, \pi, a)}(\mathrm{T}, \sigma) \\ \text { value: } \$(1+d)\left(v_{a}+v_{b}\right) \end{gathered}$ |


| $\operatorname{Win}(a, a)(a \in P)$ (leaf) |
| :--- |
| in: $\operatorname{Init}(a)$ <br> inscript $: \operatorname{sig}_{\mathbf{K}(\text { Init }, a)}$ |
| outscript $(\mathrm{T}, \sigma): \operatorname{ver}_{\mathbf{K}(\mathrm{Win}, a, a)}(\mathrm{T}, \sigma)$ <br> value $[p]: \$(1+d)$ |
| $\operatorname{Win}\left(\pi_{r}, a\right)(a \in P)($ root $)$ |
|  |
| in and inscript are variants as for $\operatorname{Win}(\pi, a)$ |
| outscript $[a](\mathrm{T}, \sigma): \operatorname{ver}_{\mathbf{K}(\text { Collect })}(\mathrm{T}, \sigma)$ <br> value $[a]: \$(n+d)$ |
| outscript $[\forall p \neq a](\mathrm{T}, \sigma):$ ver $_{\mathbf{K}(\text { Collect })}(\mathrm{T}, \sigma)$ <br> value $[p]: \$ d$ |



|  | Turn2 $(\pi, a, b)\left(\pi \in \Pi^{\prime}, \pi \sqsubset a, b\right)$ |
| :---: | :---: |
|  | in: Turn1 <br> inscript: $s_{a}^{\pi}, \operatorname{sig}_{\mathbf{K}(\text { Turn } 1, \pi, a, b)}$ |
|  | $\begin{aligned} & \text { outscript }\left(\mathrm{T}, s_{a}, s_{b}, \sigma\right): \\ & \quad\left(H\left(s_{a}\right)=h_{a}^{\pi} \wedge H\left(s_{b}\right)=h_{b}^{\pi}\right. \\ & \left.\wedge \operatorname{ver}_{\mathbf{K}\left(\operatorname{Turn} 2, \pi, \text { winner }\left(s_{a}, s_{b}, v_{a}, v_{b}\right)\right)}(\mathrm{T}, \sigma)\right) \\ & \operatorname{ver}_{\mathbf{K}(\mathrm{Turn2} 2 \mathrm{O}, \pi, a, b)}(\mathrm{T}, \sigma) \\ & \text { value: } \$(1+d)\left(v_{a}+v_{b}\right) \end{aligned}$ |


| Timeout1 $(\pi, a, b)\left(\pi \in \Pi^{\prime}, \pi \sqsubset a, b\right)$ |
| :--- |
| in: Turn1 $(\pi, a, b)$ <br> inscript $\left.: \perp, \operatorname{sig}_{\mathbf{K}(\operatorname{Turn} 1 \mathrm{TO}}, \pi, a, b\right)$ |
| outscript $(\mathrm{T}, \sigma): \operatorname{ver}_{\mathbf{K}(\operatorname{Timeout} 1, \pi, a, b)}(\mathrm{T}, \sigma)$ <br> value: $\$(1+d)\left(v_{a}+v_{b}\right)$ <br> lockTime: $\tau_{\text {Init }}+(L-\|\pi\|-1) \tau_{\text {Round }}+2 \tau_{\text {Ledger }}$ |


| $\operatorname{Timeout2}(\pi, a, b)\left(\pi \in \Pi^{\prime}, \pi \sqsubset a, b\right)$ |
| :--- |
| in: Turn2 <br> inscript $: ~$ <br>  <br> in,,$~$,$b \operatorname{sig}_{\mathbf{K}(\text { Turn2T0 }, \pi, a, b)}$ |
| outscript $(\mathrm{T}, \sigma): \operatorname{ver}_{\mathbf{K}(\text { Timeout } 2, \pi, a, b)}(\mathrm{T}, \sigma)$ <br> value: $\$(1+d)\left(v_{a}+v_{b}\right)$ <br> lockTime: $\tau_{\text {Init }}+(L-\|\pi\|-1) \tau_{\text {Round }}+4 \tau_{\text {Ledger }}$ |


| CollectOrphanWin $(\pi, a) \quad\left(\pi \in \Pi^{\prime}, \pi \sqsubset a\right)$ |
| :--- |
| in: $\operatorname{Win}(\pi, a)$ <br> Inscript: $\operatorname{sig}_{K(\text { WinTo }, \pi, a)}$ |
| outscript $[a](\mathrm{T}, \sigma): \operatorname{ver}_{K_{p}(\text { Collect })}(\mathrm{T}, \sigma)$ <br> value $[a]: \$\left(v_{a}+d\right)$ <br> lockTime: $\tau_{\text {Init }}+(L-\|\pi\|) \tau_{\text {Round }}+\tau_{\text {Ledger }}$ |
| outscript $[\forall p \neq a](\mathrm{T}, \sigma): \operatorname{ver}_{K_{p}(\text { Collect })}(\mathrm{T}, \sigma)$ <br> value $[p]: \$ d$ <br> lockTime: $\tau_{\text {Init }}+(L-\|\pi\|) \tau_{\text {Round }}+\tau_{\text {Ledger }}$ |

Fig. 3. Transaction templates used in our protocols. Let $\Pi^{\prime}$ be the set of nodes excluding leafs and the root. (Part I) Transaction templates for our ( $1, n$ )-lottery protocol. The dashed line in the inscript field indicates that both inscripts are redeemed at the same time. On the other hand, a solid line indicates that only one of the inscriptions is redeemed.

| $\operatorname{Win}(\pi, a)\left(\pi \in \Pi^{\prime}, \pi ᄃ a\right)$ <br> winner of $(k, k+1)$-lottery |
| :---: |
| in: Timeout1 $(\pi, b, a)$ <br> inscript : $\operatorname{sig}_{K(\text { Timeout1Win }, \pi, b, a)}$ |
| $\begin{aligned} & \text { in: Timeout2Win }(\pi, a, b) \\ & \text { inscript : } \operatorname{sig}_{\mathbf{K}(\text { Timeout } 2 \mathrm{Win}, \pi, a, b)} \end{aligned}$ |
| $\begin{aligned} & \text { in: Turn2 }(\pi, a, b) \\ & \text { inscript }: s_{a}, s_{b}, \operatorname{sig}_{\mathbf{K}(\operatorname{Turn} 2, \pi, a)} \end{aligned}$ |
| $\begin{aligned} & \text { in: Turn2 }(\pi, b, a) \\ & \text { inscript }: s_{a}, s_{b}, \operatorname{sig}_{\mathbf{K}(\operatorname{Turn} 2, \pi, a)} \end{aligned}$ |
| outscript $[a](\mathrm{T}, \sigma):$ ver $_{\mathbf{K}(\text { CollectW) }}(\mathrm{T}, \sigma)$ value $[a]: \$(k+1)$ |
| $\begin{gathered} \operatorname{Win}\left(\pi_{r}, a\right)(a \in P) \\ k \text {-th winner of }(k, k+1) \text {-lottery) } \end{gathered}$ |
| in and inscript are variants as for $\operatorname{Win}(\pi, a)$ |
| $\begin{aligned} & \text { outscript }[a](\mathrm{T}, \sigma): \text { ver }_{\mathbf{K}(\text { CollectW) }}(\mathrm{T}, \sigma) \\ & \text { value }[a]: \$(k+1+d) \end{aligned}$ |
| $\begin{aligned} & \text { outscript }[\forall p \neq a](\mathrm{T}, \sigma): \operatorname{ver}_{\mathrm{K}_{\mathrm{b}}(\text { CollectW) }}(\mathrm{T}, \sigma) \\ & \text { value }[p]: \$ d \end{aligned}$ |


| $\operatorname{Lose}\left(\pi, a=l_{i-1}\right) \quad\left(\pi \in \Pi^{\prime}, \pi \subset a\right)$ |
| :---: |
| in: Timeout1 $(\pi, b, a)$ <br> inscript : $\operatorname{sig}_{\mathbf{K} \text { (Timeout1Lose }, \pi, b, a)}$ |
| in: Timeout2Lose $(\pi, a, b)$ <br> inscript : $\operatorname{sig}_{\boldsymbol{K}(\text { Timeout2Lose }, \pi, a, b)}$ |
| $\begin{aligned} & \text { in: Turn2 }(\pi, a, b) \\ & \text { inscript }: s_{a}, s_{b}, \operatorname{sig} \mathbf{K}_{\mathbf{K}(\operatorname{Turn} 2, \pi, a)} \end{aligned}$ |
| $\begin{aligned} & \text { in: Turn2 }(\pi, b, a) \\ & \text { inscript }: s_{a}, s_{b}, \operatorname{sig} \\ & \mathbf{K}(\operatorname{Turn} 2, \pi, a) \end{aligned}$ |
| $\begin{gathered} \text { outscript }(\mathrm{T}, \sigma): \operatorname{ver}_{\mathbf{K}(\operatorname{Lose}, \pi, a)}(\mathrm{T}, \sigma) \\ \operatorname{Vver}_{\mathbf{K}(\text { Loseto }, \pi, a)}(\mathrm{T}, \sigma) \\ \text { value: } \$(k+1-i+d(1+i)) \end{gathered}$ |



| $\operatorname{Turn2} 2\left(\pi_{r}, a, b\right)\left(\pi_{r} \sqsubset a, b\right)$ |
| :--- |
| in: Turn1 |
| inscript: $s_{a}, \operatorname{sig}_{\mathbf{K}(\operatorname{Turn} 1, \pi, a, b)}$ |
| outscript(T, $\left.s_{a}, s_{b}, \sigma\right):$ <br> $\quad\left(H\left(s_{a}\right)=h_{a}^{\pi} \wedge H\left(s_{b}\right)=h_{b}^{\pi}\right.$ <br> $\left.\wedge \operatorname{ver}_{\mathbf{K}\left(\operatorname{Turn2} 2 \pi, \text { winner }\left(s_{a}, s_{b}, v_{a}, v_{b}\right)\right)}(\mathrm{T}, \sigma)\right)$ <br> $\operatorname{Vver}_{\mathbf{K}(\operatorname{Turn2ToWin}, \pi, a, b)}(\mathrm{T}, \sigma)$ <br> $\operatorname{value:~}(k+1)(d+1)$ |


| Timeout1 $(\pi, a, b)\left(\pi_{r} \neq \pi \in \Pi^{\prime}, \pi \sqsubset a, b\right)$ |
| :---: |
| $\begin{aligned} & \text { in: Turn1 }(\pi, a, b) \\ & \text { inscript }: \perp \text {, } \operatorname{sig}_{\mathbf{K}(\text { Turn1TO }, \pi, a, b)} \end{aligned}$ |
| $\begin{aligned} & \text { outscript }(\mathrm{T}, \sigma) \text { : } \operatorname{ver}_{\mathbf{K}(\text { Timeout1Win }, \pi, a, b)}(\mathrm{T}, \sigma) \\ & \text { value: } \$(k+1) \\ & \text { lockTime: } \tau_{\text {Init }}+(L-\|\pi\|-1) \tau_{\text {Round }}+2 \tau_{\text {Ledger }} \end{aligned}$ |
| $\begin{aligned} & \text { outscript }(\mathrm{T}, \sigma) \text { : } \operatorname{ver}_{\mathbf{K}(\text { Timeout 1Lose }, \pi, a, b)}(\mathrm{T}, \sigma) \\ & \text { value: } \$(k-1-i+d(2+i)) \\ & \text { lockTime: } \tau_{\text {Init }}+(L-\|\pi\|-1) \tau_{\text {Round }}+2 \tau_{\text {Ledger }} \end{aligned}$ |
| Timeout2Win $(\pi, a, b)\left(\pi \in \Pi^{\prime}, \pi \sqsubset a, b\right)$ |
| $\begin{aligned} & \text { in: Turn2 }(\pi, a, b) \\ & \text { inscript } \left.: \perp, \perp, \operatorname{sig}_{K(T u r n 2 T O W i n}, \pi, a, b\right) \end{aligned}$ |
| $\begin{aligned} & \text { outscript }(\mathrm{T}, \sigma) \text { : } \text { ver }_{\mathbf{K}(\text { Timeout } 2 \mathrm{Win}, \pi, a, b)}(\mathrm{T}, \sigma) \\ & \text { value: } \$(k+1) \\ & \text { lockTime: } \tau_{\text {Init }}+(L-\|\pi\|-1) \tau_{\text {Round }}+4 \tau_{\text {Ledger }} \end{aligned}$ |


| CollectOrphanLose $(\pi, a)\left(\pi \in \Pi^{\prime}, \pi \sqsubset a\right)$ |
| :---: |
| in: Lose $(\pi, a)$ Inscript:sig K(LoseTo $, \pi, a)$ |
| $\begin{aligned} & \text { outscript }[a](\mathrm{T}, \sigma): \operatorname{ver}_{K_{p}(\text { CollectL })}(\mathrm{T}, \sigma) \\ & \text { value }[a]: \$(k+1-i+d) \\ & \text { lockTime: } \tau_{\text {Init }}+(L-\|\pi\|) \tau_{\text {Round }}+\tau_{\text {Ledger }} \end{aligned}$ |
| $\begin{aligned} & \text { outscript }[\forall p \neq a](\mathrm{T}, \sigma): \operatorname{ver}_{K_{p}(\text { CollectL })}(\mathrm{T}, \sigma) \\ & \text { value }[p]: \$ d \\ & \text { lockTime: } \tau_{\text {Init }}+(L-\|\pi\|) \tau_{\text {Round }}+\tau_{\text {Ledger }} \end{aligned}$ |

Fig. 4. Transaction templates for our ( $k, k+1$ )-lottery protocol. Let $\pi_{i}$ denote $i$-th match of the protocol. We omit Win and Init descriptions since they are almost the same in Fig. 3. The differences from Fig. 3 are just changes of the values $\$(1+d)$ to $\$(k+d)$.

Theorem 1 Our $(1, n)$-lottery protocol is secure and constant deposit.
Proof (Sketch). We prove that our protocol fulfills the definition 1. In the case of $C=\emptyset$, it is obvious from Lemma 1 . If an adversary deviates from the procedure or aborts at some step in the initialization phase, players terminate the protocol. In this case, all honest players do not lose money since no money transfers occur. Thus, we suppose that the initialization phase completes correctly. Below, we discuss two cases in the tournament execution phase, (i) an adversary rejects to reveal its secrets and (ii) an adversary applies the transaction insertion attack, described in Section 5.3.

In the case of (i), the biased coin-tossing protocol guarantees that the player who did not reveal the secret is treated as a loser. Thus, no honest player is lost nevertheless an adversary refuses to disclose its secret in any matches.

In the case of (ii), let us consider the case where an adversary applies the transaction insertion attack to a player $p$ at match $\pi$. The player $p$ obtains payoff $\$\left(v_{p}^{\pi}+d-1\right)$ by CollectOrphanWin at the end of the protocol, as described Section 5.3. Note that, in this case, the player $p$ does not reveal his/her secret corresponding to match $\pi$. Furthermore, at the beginning of the match, we can express the expected payoff of $p$ as follows

$$
\begin{equation*}
\frac{v_{p}^{\pi}}{v_{p}^{\pi_{r}}} \times \$(n-1)+\left(1-\frac{v_{p}^{\pi}}{v_{p}^{\pi_{r}}}\right) \times \$(-1)=\$\left(v_{p}^{\pi}-1\right) \tag{3}
\end{equation*}
$$

The inequality $v_{p}^{\pi}+d-1>v_{p}^{\pi}-1$ implies that $E\left(\Phi\left(p, s t_{0}, \sigma_{\mathcal{A}}\right)\right)>0$ if $d>0$. Also, this property holds for an arbitrary positive integer $d$, our protocol satisfies constant-deposit. From the above, $(1, n)$-lottery protocol is secure.

## $6(k, n)$-Lottery Protocol with Constant Deposits

This section shows our ( $k, n$ )-lottery protocol for arbitrary $k$ and $n$ and $(k, k+1)$ lottery protocol for arbitrary $k$ and $k+1$. We compose a ( $k, n$ )-lottery protocol from a composition of $(k, k+1)$-lottery protocols as follows:

First $n$ parties run $(n-1, n)$-lottery and determine one loser. Thereafter, the remaining $n-1$ winners run ( $n-2, n-1$ )-lottery and further determine one loser. Parties repeat the similar process until removing $n-k$ players, i.e., resulting in $k$ winners.

### 6.1 Building Block: Modified Biased Coin-Tossing Protocol

We adopt the single-elimination tournament as described in Lemma 2 to construct a $(k, k+1)$-lottery protocol. That is, it is a tournament where the winner of each match becomes the champion of ( $k, k+1$ )-lottery, and the loser moves on to the next match. Protocol 1 is insufficient to implement such a tournament since it does not enable the loser to proceed to the next match. Hence, we here modify the protocol to resolve this problem.

## Protocol 2 Modified Biased Coin-Tossing $\Pi_{a, b}^{\mathrm{WL}}\left(s_{a}, s_{b}, v_{a}, v_{b}\right)$

## Setup:

1: The initialization phase was successfully completed, and $\operatorname{Lose}(\pi, a)$ and $\operatorname{Win}(\pi, b)$ have been put already on the ledger. Players $a$ and $b$ hold secrets $s_{a}^{\pi}$ and $s_{b}^{\pi}$, respectively. Let $\tau$ be the round of the beginning of the protocol.

## Procedure:

2: One of the players puts $\operatorname{Turn1}(\pi, a, b)$ on the ledger.
3: $a$ writes $s_{a}^{\pi}$ on the input script of $\operatorname{Turn2}(\pi, a, b)$, and put the transaction on the ledger.
if Turn2 $(\pi, a, b)$ does not appear within $\tau+2 \tau_{\text {Ledger }}$ then
One of the players puts Timeout $1(\pi, a, b)$ on the ledger.
One of the players puts $\operatorname{Win}(\pi, b)$ and $\operatorname{Lose}(\pi, a)$ on the ledger.
$b$ computes $w=\operatorname{Winner}\left(s_{a}, s_{b}, v_{a}, v_{b}\right)$
if $w=a$ then $b$ puts $\operatorname{Win}(\pi, a)$ and $\operatorname{Lose}(\pi, b)$ on the ledger.
if $w=b$ then $b$ puts $\operatorname{Win}(\pi, b)$ and $\operatorname{Lose}(\pi, a)$ on the ledger.
if $\operatorname{Win}(\pi, x \in\{a, b\})$ does not appear within $\tau+4 \tau_{\text {Ledger }}$ then One of the players puts $\operatorname{Timeout} 2 \mathrm{Win}(\pi, a, b)$ into the ledger. One of the players puts $\operatorname{Win}(\pi, a)$ on the ledger.
if Lose $(\pi, x \in\{a, b\})$ does not appear within $\tau+4 \tau_{\text {Ledger }}$ then One of the players puts Timeout2Lose $(\pi, a, b)$ into the ledger.
17: One of the players puts Lose $(\pi, a)$ on the ledger.


Fig. 5. Graphical description of modified biased coin-tossing (for match $\pi_{i}$ ). We denote with $\pi_{a}$ and $\pi_{b}$ child nodes of $\pi_{i}$. Note that Win and Lose redeemed by Turn1 are omitted.


Fig. 6. Graphical description of biased coin-tossing (for match $\pi_{r}$ ) for the final match of $(k, k+1)$-lottery.

See Protocol 2 and Fig. 5 that show the modified protocol. The Loser function described in the Lose transaction returns the inverse of Winner function, i.e., it specifies the loser. That is, unlike Protocol 5.1, the loser also puts Lose transaction of which input is Turn2, and receives coins used in the next match. Moreover, since Turn2 has two output scripts, we set the timeouts for each of Win and Lose by preparing Timeout2Win and Timeout2Lose transactions. If Win or Lose is not published within the time limit, it is dealt with by publishing Timeout2Win or Timeout2Lose respectively. Fig. 7 shows flows of procedures when a timeout occurs.

### 6.2 Our Construction of $(k, k+1)$-Lottery Protocol

Let the bet mount be $\$ k$ for each player in this section.
The biased probability of each match in $(k, k+1)$-Lottery. Suppose a match between $p_{i+1}$ and $l_{i-1}$ in $i$-th match $\pi_{i}$, where $l_{i-1}$ is the loser of $(i-1)$ th match. From Lemma 2, for $i=1 \ldots k-1$, the winning probability of $p_{i+1}$ in $\pi_{i}$ is set as $i /(i+1)$.

Based on the biased probability, our protocol proceeds as follows.
Precondition: for all players, the ledger contains a transaction $\operatorname{Bet}_{p}$ with value $\$(1+d)$, and redeemable with key $K_{p}\left(\operatorname{Bet}_{p}\right)$.

## Initialization phase:

1. For all player $p \in P, p$ generates the following secret keys locally.

- For all $\pi$ such that $|\pi|=L$ : $K_{p}\left(\operatorname{Bet}_{p}\right), K_{p}($ CollectW $), K_{p}($ CollectL $), K_{p}(\operatorname{Init}, a)$
- For all $\pi$ such that $1 \leq|\pi| \leq L$ : $K_{p}($ Win $, \pi, a), K_{p}($ WinTo, $\pi, a), K_{p}($ Lose $, \pi, a), K_{p}($ LoseTo, $\pi, a)$
- For all $\pi$ such that $1 \leq|\pi|<L$ : $K_{p}$ (Turn1To, $\left.\pi, a, b\right), K_{p}($ Turn1, $\pi, a, b)$, $K_{p}($ Turn2ToWin, $\pi, a, b), K_{p}($ Turn2ToLose, $\pi, a, b), K_{p}($ Turn2, $\pi, a)$, $K_{p}$ (Timeout1Win, $\left.\pi, a, b\right), K_{p}$ (Timeout1Lose, $\pi, a, b$ ), $K_{p}($ Timeout2Win, $\pi, a, b), K_{p}($ Timeout2Lose, $\pi, a, b)$

2. For all player $p \in P, p$ generates secrets $s_{p}^{\pi_{p}}$ for each $\pi_{p}$, such that $\left(\left|\pi_{p}\right|<L\right)$, and broadcasts to the other players his/her public keys and hashes $h_{p}^{\pi_{p}}=H\left(s_{p}^{\pi_{p}}\right)$.
3. If $h_{p}^{\pi_{p}}=h_{p^{\prime}}^{\pi_{p}{ }^{\prime}}$ for some $\left(p, \pi_{p}\right) \neq\left(p^{\prime}, \pi_{p}{ }^{\prime}\right)$, the players abort.
4. Parties agree the time $\tau_{\text {Init }}$ large enough to fall after the initialization phase. (This step is necessary to determine lockTime values built in the subsequent steps.)
5. Each player signs all the transaction templates in Fig. 3 and 4 except for Init and broadcasts the signatures.
6. Each player verifies the signatures received by the others. some signature is not valid or missing, the player aborts the protocol.
7. Each player signs Init, and sends the signature to the first player.
8. The first player puts the (signed) transaction Init on the ledger.
9. If Init does not appear within one $\tau_{\text {Ledger }}$, then each $p$ redeems $\operatorname{Bet}_{p}$ and aborts.
10. The players put the signed transactions $\operatorname{Win}(p, p)$ on the ledger, for all $p \in P$.
Tournament execution phase: For levels $i=k-1 \ldots 2$, players proceed as follows: Run $\Pi_{a, b}^{\mathrm{WL}}\left(s_{a}^{\pi_{i}}, s_{b}^{\pi_{i}}, v_{a}^{\pi_{i}}, v_{b}^{\pi_{i}}\right)$ for each $\pi$, such that $|\pi|=i-1$.
For level $i=1$, players proceed as follows: Run $\Pi_{a, b}^{\mathrm{W}}\left(s_{a}^{\pi_{i}}, s_{b}^{\pi_{i}}, v_{a}^{\pi_{i}}, v_{b}^{\pi_{i}}\right)$. Then, $v_{a}^{\pi}, v_{b}^{\pi}$ denote the biased probability determined in the manner shown in the previous subsection.
Garbage collection phase: If there is some unredeemed $\operatorname{Win}(\pi, p)$ such that $\pi$ is a leaf on the ledger, players put CollectOrphanWin $(\pi, p)$ on the ledger. Similarly, if there is some unredeemed $\operatorname{Lose}(\pi, p)$ on the ledger, players put CollectOrphanLose $(\pi, p)$ on the ledger.

At the end of the tournament execution phases, all champions can freely redeem $\operatorname{Win}(\pi, a)$ or $\operatorname{Win}\left(\pi_{r}, a\right)$ as rewards. Also, $\operatorname{Win}\left(\pi_{r}, a\right)$ guarantees that every honest party can collect their deposits. As in Protocol 5.1, the number of transactions prepared at step 5 is $O\left(n^{2}\right)$.

Theorem 2 Our $(k, k+1)$-lottery protocol is secure and constant deposit.

Proof (Sketch). We prove that our protocol fulfills Definition 2. In the case of $C=\emptyset$, it is obvious from Theorem 2. As in the proof of Theorem 1, we suppose


Fig. 7. Graph of the transactions in a tournament round. An arrow from transaction $T$ to $T^{\prime}$ means that $T$ redeems $T^{\prime}$. Thick arrows mean any player can redeem; dashed edges mean any player can redeem, but only after a timeout. Thin arrows mean that only the player who knows the secret on the label can redeem it. $\tau_{\text {Round }}:=6 \tau_{\text {Ledger }}$ refers to the number of rounds in each match.
that the initialization phase completes correctly and focuses on the tournament execution phase.

Below, we discuss two cases in the tournament execution phase: (i) an adversary rejects to reveal its secrets, and (ii) an adversary applies the transaction insertion attack, described in Section 5.3. The proof of case (i) is omitted since the same argument holds for Theorem 1. For case (ii), we consider further dividing it into the following two cases: (a) an adversary applies the transaction insertion attack to player $p_{i+1}$ at match $\pi_{i}$, where $\pi_{i}$ is the first match for $p_{i+1}$, (b) an adversary applies the attack to player $l_{i+1}$ at match $\pi_{i}$, where $l_{i+1}$ is the loser of the previous match.

Then, the player $p_{i+1}$ obtains payoff $\$(k+d)$ at the end of the protocol. Also, player $l_{i-1}$ at match $\pi_{i}$ obtains payoff $\$(1-i+d)$ by CollectOrphanWin at the end of the protocol. Thus, to confirm that the honest party does not lose by the attack, it requires that the obtained payoff is more than the expected payoff at match $\pi_{i}$. In the case of (a), for any $\pi_{i}$, the expected payoff of player $p_{i}$ is 0 because $p_{i}$ because it is fair to the players from Theorem 2.

In the case of (b), The expected payoff of honest $l_{i-1}$ is as follows.

$$
\begin{equation*}
\frac{k+1-i}{k+1} \times \$ 1+\left(\frac{i}{k+1}\right) \times \$(-k)=\$(1-i) \tag{4}
\end{equation*}
$$

From this Eq.(4), we can see $E\left(\Phi\left(p, s t_{0}, \sigma_{\mathcal{A}}\right)\right)-E\left(\Phi\left(p, s t_{0}, \perp\right)\right)>0$ since $1-i+$ $d>1-i$ for $i \in[k]$ if $d>0$. It implies that the $l_{i-1}$ 's expected payoff when the adversary applies the transaction insertion attack is larger than their expected payoff when all parties behave honestly.

Next, we confirm that every subset has the same winning probability for all $S=\left\{s_{1}, \ldots, s_{k}\right\} \subset P$. It is obvious if all honest parties behave honestly
since one loser is determined uniformly at random. To change the probability, adversaries can make two attacks: rejections of their secret or the transaction insertion attack. In both cases, since the expected payoff of the adversary is negative, the protocol fulfills the requirement. This ( $k, k+1$ )-lottery protocol is secure from the above.

### 6.3 Construction of $(k, n)$-Lottery from $(k, k+1)$-Lottery

Let a bet amount of each party be $\alpha=n!/ k!$. There are two technical challenges to realizing a secure $(k, n)$-lottery based on this strategy. The first one is to connect each $\left(k^{\prime}, k^{\prime}+1\right)$-lottery protocol such that malicious parties cannot escape the protocol in the middle. This issue is derived from the fact that if parties run several lottery protocols sequentially, corrupted players can abort without losing at the initialization process of the next lottery. To circumvent this issue, we aggregate the initialization processes of all protocols in the first $(n-1, n)$ lottery protocol. That is, players prepare all of the secrets, signing (verification) keys, and transactions used in the entire ( $k, n$ )-lottery in the initialization of ( $n-1, n$ )-lottery protocol. By this modification, parties can skip all initialization phases after the completion of $(n-1, n)$-lottery protocol. Note that the number of transactions created in the initialization phase is $O\left(n^{3}\right)$, which can be derived from the number of possible match combinations.

Further, we also slightly change the tournament execution phase, except for the last $(k, k+1)$ lottery protocol. More concretely, we modify each match protocol, i.e., Fig. 5 and 6, such that Win transactions connect two tournament execution phases. See Appendix B for the modification details.

As the second challenge, it is necessary to ensure that $(k, n)$-lottery composed of sequential executions of $(n-j, n-j+1)$-lottery for $j \in[n-k]$ is indeed fair. We present the security proof of our ( $k, n$ )-lottery protocol below.

Theorem 3 Our ( $k, n$ )-lottery protocol is secure and constant deposit.

Proof (Sketch). We prove that our ( $k, n$ )-lottery protocol fulfills Definition 2.
As in the proof of our $(k, k+1)$-lottery protocol, we focus on the payoff obtained by a player who is affected by the transaction insertion attack at $\pi_{i}^{j}$, where $\pi_{i}^{j}$ is $i$-th match of $j$-th $(n-j, n-j+1)$-lottery protocol. We denote by $w_{i}^{j}$ and $l_{i}^{j}$ the winner and loser of match $\pi_{i}^{j}$, respectively. As in the proof of Theorem 2, we consider further dividing case (ii) into the following two cases: (a) an adversary applies the transaction insertion attack to player $l_{i-1}^{j}$ at match $\pi_{i}$, (b) the attack to player $w_{i+1}^{j-1}$ at match $\pi_{i}^{j}$. In the case of (b), the player $l_{i-1}^{j}$ obtains payoff $\$\left((n-1)!/\{(k-1)!(n-j+1)(n-j)\} \times\left(n j-n i-j^{2}+j\right)+d\right)$ at the end of the protocol. Thus, to confirm that the honest party does not lose by the attack, it requires that the obtained payoff is more than the expected payoff at match $\pi_{i}^{j}$. The expected payoff of honest $l_{i-1}^{j}$ is as follows.

$$
\begin{aligned}
& \frac{(n-j+1-i) k}{(n-j+1)(n-j)} \times \$ \frac{(n-1)!(n-k)}{k!}+\left(1-\frac{(n-j+1-i) k}{(n-j+1)(n-j)}\right) \times \$\left(-\frac{(n-1)!}{k!}\right) \\
& =\$ \frac{(n-1)!}{(k-1)!(n-j+1)(n-j)}\left(n j-n i-j^{2}+j\right) .
\end{aligned}
$$

Note that $((n-j+1-i) k) /((n-j+1)(n-j))$ is the winning probability of $l_{i-1}^{j}$. From then on, we could prove similar to the proof of our $(k, k+1)$-lottery protocol. A similar calculation in the case of (a) shows no loss. Hence, our ( $k, n$ )lottery is secure.

Acknowledgment This work was supported by JSPS KAKENHI Grant Numbers JP18H05289, JP21H03395, JP21H03441, JP22H03590, JP23H00468, JP23H00479, 23K17455, JP23K16880, JST CREST JPMJCR22M1, JPMJCR23M2, and MEXT Leading Initiative for Excellent Young Researchers.

## References

1. Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, L.: Secure multiparty computations on bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 443-458 (2014). https://doi.org/10.1109/SP. 2014.35
2. Back, A., Bentov, I.: Note on fair coin toss via bitcoin. CoRR abs/1402.3698 (2014), http://arxiv.org/abs/1402.3698
3. Badertscher, C., Maurer, U., Tschudi, D., Zikas, V.: Bitcoin as a transaction ledger: A composable treatment. In: Advances in Cryptology - CRYPTO 2017. pp. 324356. Springer International Publishing (2017)
4. Bartoletti, M., Zunino, R.: Constant-deposit multiparty lotteries on bitcoin. In: International Conference on Financial Cryptography and Data Security. pp. 231247. Springer (2017)
5. Baum, C., David, B., Dowsley, R.: Insured mpc: Efficient secure computation with financial penalties. In: Financial Cryptography and Data Security. pp. 404-420. Springer International Publishing, Cham (2020)
6. Belenkiy, M., Chase, M., Erway, C.C., Jannotti, J., Küpçü, A., Lysyanskaya, A., Rachlin, E.: Making p2p accountable without losing privacy. In: Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society. p. 31-40. Association for Computing Machinery (2007). https://doi.org/10.1145/1314333.1314339
7. Bentov, I., Kumaresan, R.: How to use bitcoin to design fair protocols. In: Advances in Cryptology - CRYPTO 2014. pp. 421-439 (2014)
8. Bentov, I., Kumaresan, R., Miller, A.: Instantaneous decentralized poker. In: Advances in Cryptology - ASIACRYPT 2017. pp. 410-440. Springer International Publishing (2017)
9. Chaum, D.: Blind signatures for untraceable payments. In: Advances in Cryptology. pp. 199-203. Springer US (1983)
10. Choudhuri, A.R., Goyal, V., Jain, A.: Founding secure computation on blockchains. In: Advances in Cryptology - EUROCRYPT 2019. pp. 351-380. Springer International Publishing (2019)
11. Cleve, R.: Limits on the security of coin flips when half the processors are faulty. In: Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing. p. 364-369. STOC '86, Association for Computing Machinery, New York, NY, USA (1986). https://doi.org/10.1145/12130.12168
12. Goldschlag, D.M., Stubblebine, S.G.: Publicly verifiable lotteries: Applications of delaying functions. In: Financial Cryptography. pp. 214-226. Springer Berlin Heidelberg (1998)
13. Hall, C., Schneier, B.: Remote electronic gambling. In: Computer Security Applications Conference, Annual. p. 232. IEEE Computer Society (1997). https://doi.org/10.1109/CSAC.1997.646195
14. Konstantinou, E., Liagkou, V., Spirakis, P., Stamatiou, Y.C., Yung, M.: Electronic national lotteries. In: Financial Cryptography. pp. 147-163. Springer Berlin Heidelberg (2004)
15. Kumaresan, R., Bentov, I.: How to use bitcoin to incentivize correct computations. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. p. 30-41. Association for Computing Machinery (2014). https://doi.org/10.1145/2660267.2660380
16. Kumaresan, R., Moran, T., Bentov, I.: How to use bitcoin to play decentralized poker. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. p. 195-206. Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2810103.2813712
17. Küpçü, A., Lysyanskaya, A.: Usable optimistic fair exchange. In: Topics in Cryptology - CT-RSA 2010. pp. 252-267. Springer Berlin Heidelberg (2010)
18. Kushilevitz, E., Rabin, T.: Fair e-lotteries and e-casinos. In: Topics in Cryptology - CT-RSA 2001. pp. 100-109. Springer Berlin Heidelberg (2001)
19. Lindell, A.Y.: Legally-enforceable fairness in secure two-party computation. In: Malkin, T. (ed.) Topics in Cryptology - CT-RSA 2008. pp. 121-137. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
20. Miller, A., Bentov, I.: Zero-collateral lotteries in bitcoin and ethereum. In: 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS\&PW). pp. 4-13 (2017). https://doi.org/10.1109/EuroSPW.2017.44
21. Nakai, T., Shinagawa, K.: Constant-round linear-broadcast secure computation with penalties. Theoretical Computer Science 959, 113874 (2023). https://doi.org/10.1016/j.tcs.2023.113874
22. Nakai, T., Shinagawa, K.: Secure multi-party computation with legally-enforceable fairness. In: Information and Communications Security. pp. 161-178. Springer Nature Singapore (2023)
23. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Decentralized business review (2008)
24. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper 151, 1-32 (2014)

## A Proofs of Lemmas

Proof of Lemma 1: Let $\pi_{l \in[L]}$ such that $\left|\pi_{l}\right|=l$ be the $l$-th match for player $p$. Suppose $v_{p}^{l} / v_{p}^{l-1}$ be the probability that $p$ wins at $\pi_{l}$. Then, the probability that $p$ wins the tournament holds:

$$
\frac{1}{v_{p}^{l}} \times \frac{v_{p}^{l}}{v_{p}^{l-1}} \times \cdots \times \frac{v_{p}^{1}}{v_{p}^{0}}=\frac{1}{n_{p}^{0}}=\frac{1}{N}
$$

This is also true for any player.
Proof of Lemma 2: Let $\pi_{i}$ such that $\left|\pi_{i}\right|=i \in[k]$ be the $i$-th match for player $p_{i+1}$ and $l_{i-1}$, where $l_{i-1}$ is the loser of $(i-1)$-th match The probability that $p$ wins the tournament holds:

$$
1-\frac{i}{i+1} \times \frac{i+1}{i+2} \times \cdots \times \frac{k}{k+1}=\frac{k}{k+1}
$$

This is also true for any player. Moreover, the probability of winning the parties in $S$ simultaneously equals the probability of losing $p \notin S$. Thus, the probability of winning the parties in $S$ simultaneously is equivalent for any $S \subset P$ such that $|S|=k$.
Proof of Lemma 3: For any $j \in[k]$, the winning probability in $(n-j, n-j+1)$ lottery can be expressed by $(n-j-1) /(n-j)$, as shown in Lemma 2. Since the probability of each $\left(k^{\prime}, k^{\prime}+1\right)$-lottery is independent, the probability that a player wins the entire $(k, n)$-lottery can be written as:

$$
\frac{n-1}{n} \times \frac{n-2}{n-1} \times \cdots \times \frac{k}{k+1}=\frac{k}{n} .
$$

Moreover, since the losers are chosen uniformly at random in each $\left(k^{\prime}, k^{\prime}+1\right)$ lottery, it is obvious that the winning probability of any set of $k$ players is equivalent.

## B Transaction Templates for Constructing ( $k, n$ )-Lottery

To combine multiple $(k, k+1)$-lottery protocols, we modify Win transactions. See Fig. 8 that shows the point of connection between $j$-th lottery and $(j+1)$ th lottery protocols. The output scripts of $\operatorname{Win}\left(\pi^{j}, a\right)$ in $j$-th lottery are used as input of $\operatorname{Win}\left(\pi^{j+1}, a\right)$ in $(j+1)$-th lottery protocol. Furthermore, $\operatorname{Win}\left(\pi_{r}^{j}, a\right)$ redistributes $\$ d$ to $\operatorname{Win}(\pi, a)$ for deposits of the next lottery. With this modification, $K_{p}($ Winlnit, $\pi, a)$ and $K_{p}($ Return, $\pi, a)$ are added to the key pairs prepared in the initialization phase.


Fig. 8. Graphical description of the connection between $j$-th lottery and $(j+1)$-th lottery protocols

# Single-Shuffle Card-Based Protocols with Six Cards per Gate 

Tomoki Ono ${ }^{1}$, Kazumasa Shinagawa ${ }^{2,3}$, Takeshi Nakai ${ }^{4}$, Yohei Watanabe ${ }^{1,3}$, and Mitsugu Iwamoto ${ }^{1}$<br>${ }^{1}$ The University of Electro-Communications, Tokyo, Japan<br>\{onotom, watanabe, mitsugu\}@uec.ac.jp<br>${ }^{2}$ Ibaraki University, Ibaraki, Japan kazumasa.shinagawa.np92@vc.ibaraki.ac.jp<br>${ }^{3}$ National Institute of Advanced Industrial Science and Technology, Tokyo, Japan<br>${ }^{4}$ Toyohashi University of Technology, Aichi, Japan nakai@cs.tut.ac.jp


#### Abstract

Card-based cryptography refers to a secure computation with physical cards, and the number of cards and shuffles measures the efficiency of card-based protocols. This paper proposes new card-based protocols for any Boolean circuits with only a single shuffle. Although our protocols rely on Yao's garbled circuit as in previous single-shuffle card-based protocols, our core construction idea is to encode truth tables of each Boolean gate with fewer cards than previous works while being compatible with Yao's garbled circuit. As a result, we show single-shuffle card-based protocols with six cards per gate, which are more efficient than previous single-shuffle card-based protocols.


Keywords: Card-based cryptography • Secure computation • Garbled circuit.

## 1 Introduction

### 1.1 Background and Motivation

Secure computation protocols allow parties to collaboratively compute a function while keeping each party's input hidden from the other party. Although secure computation protocols are usually implemented on computers, card-based cryptography $[3,4]$, which is an area focusing on secure computation using physical cards (without computers), has also been eagerly investigated. Let us give an example of a secure card-based AND protocol called the five-card trick [3]. Suppose that each of Alice and Bob has two cards, $\boldsymbol{\infty}$ and $Q$, and a $\varphi$ is placed face-down on a table. Alice (resp., Bob) puts their cards face-down on the left side (resp., the right side) of $\bar{\Omega}$, following the encoding rule: the order of the cards is $\boldsymbol{\phi} Q$ if the input is zero; it is $Q$ if the input is one. After shuffling the five face-down cards without changing the order of the sequence, they face up the cards. The output of the AND protocol is one if the consecutive three heart cards appear; it is zero otherwise.

The major efficiency measures of card-based cryptography are the number of cards and shuffles. The fewer cards and shuffles card-based protocols are realized, the easier it is to execute them. In this work, we focus on the implementability of shuffles; it is unclear how to implement shuffles that yield desired probability distributions, though various attempts have been made thus far $[7,11,17,18,22$, $23,28,29]$. For this reason, we devote effort to constructing card-based protocols with the minimum number of shuffles and as few cards as possible.

A well-known approach to constructing card-based protocols for any function is to realize card-based protocols for a Boolean gate such as AND and XOR since any function can be realized by combining Boolean gates [4, 12, 15]. Hence, improving card-based protocols for Boolean gates is one of the mainstream research topics $[1,2,4-6,8,9,12-14,19-21,25,26]$. However, this approach increases the number of shuffles required for the resulting card-based protocols for any function (or Boolean circuit) since the number of shuffles depends on the number of gates consisting of the Boolean circuit. Therefore, we aim to directly propose card-based protocols for any Boolean circuit consisting of various Boolean gates, not any Boolean gate, with a single shuffle. Note that, as stated in [24], it is impossible to realize card-based protocol for any non-trivial function without shuffles; The lower bound of shuffles required for secure card-based protocols is one.

### 1.2 Prior Works

Shinagawa and Nuida [24] showed a single-shuffle card-based protocol for any $n$-variable Boolean circuit $f:\{0,1\}^{n} \rightarrow\{0,1\}^{m}$ with $24 q+2 n$ cards, where $q$ is number of gates in the Boolean circuit. Tozawa et al. [27] improved the Shinagawa-Nuida protocol and reduced the number of cards to $8 q+2 n$ without additional shuffles. Kuzuma et al. [10] focused on a restricted class of Boolean circuits and showed single-shuffle card-based protocols for an $n$-variable AND function with $4 n-2$ cards and an $n$-variable XOR function with $2 n$ cards. Note that, allowing multiple shuffles, Nishida et al. [16] showed a card-based protocol with $2 n+6$ cards for any $n$-variable Boolean circuit $f:\{0,1\}^{n} \rightarrow\{0,1\}$, which is the most efficient protocol in terms of the number of cards.

### 1.3 Our Contribution

This paper proposes new single-shuffle card-based protocols based on Yao's garbled circuits [30]. The core construction idea is to encode truth tables of each Boolean gate with fewer cards than previous protocols while being compatible with Yao's garbled circuit. Unlike previous single-shuffle card-based protocols such as Shinagawa-Nuida [24] and Tozawa et al. [27], each output of the truth table is represented by a single card, and we add two more extra cards to make the truth tables encoded with single cards compatible with Yao's technique. As a result, we show two single-shuffle card-based protocols for any Boolean circuit $f:\{0,1\}^{n} \rightarrow\{0,1\}^{m}$ with six cards per gate: One is a non-committed-format protocol with $2 n+6 q$ cards, and the other is a committed-format protocol with

Table 1. A comparison among protocols with one shuffle for any Boolean circuit. $q, n$, and $m$ are the number of gates, bit-length of the input, and bits-length of the output, respectively.

|  | Format | Number of cards | Shuffle type |
| :---: | :---: | :---: | :---: |
| Shinagawa-Nuida [24] | committed | $24 q+2 n$ | uniform closed |
| Tozawa et al. [27] | committed | $8 q+2 n$ | uniform closed |
| Our protocol (Section 3.6) | non-committed | $6 q+2 n$ | uniform |
| Our protocol (Section 3.7) | committed | $6 q+2(n+m)$ | uniform |

$2(n+m)+6 q$ cards, where $q$ is the number of gates in $f$, and a protocol is said to be committed if it outputs cards face-down and the output follows the same encoding rule as the input.

Table 1 shows a comparison among the existing protocols and our protocols. Since the number of gates $q$ is greater than or equal to the number of the output gates $m$, our protocol is more efficient than those of Shinagawa-Nuida [24] and Tozawa et al. [27] in terms of the number of cards. It should be noted that our protocols use a uniform shuffle, which is not closed (see Section 2.2 for the definition), although Shinagawa-Nuida [24] and Tozawa et al. [27] used a uniform closed shuffle.

### 1.4 Organization

In Section 2, we introduce basic definitions. In Section 3, we construct our singleshuffle protocols both in the non-committed-format setting and the committedformat setting. In Section 4, we conclude our paper.

## 2 Preliminaries

For an integer $k \geq 1$, we denote the $k$-th symmetric group by $S_{k}$. For two permutations $\pi_{1}, \pi_{2} \in S_{k}$, the composition of them is denoted by $\pi_{2} \circ \pi_{1}$. Here, permutations are applied from right to left, i.e., $\pi_{2} \circ \pi_{1} \in S_{k}$ means that permutation $\pi_{1}$ is applied and then $\pi_{2}$ is applied. For two subsets $A, B \subseteq S_{k}$, we define $A B:=\left\{\pi_{A} \circ \pi_{B} \mid \pi_{A} \in A, \pi_{B} \in B\right\}$.

### 2.1 Syntax of Boolean Circuits

A Boolean circuit $C$ is defined by a 6 -tuple ( $n, m, q, L, R, G$ ) where $n \geq 1$ is the number of input wires, $m \geq 1$ is the number of output wires, $q \geq 1$ is the number of gates, $L$ and $R$ are functions that specify the left and right wires in each gate, respectively, and $G$ is a function that specifies the truth table of each gate. The detailed specification is given in the following.

- The number of wires in $C$ is $n+q$, where $n$ wires are the input wires and $q$ wires are the output wires of gates. Each input wire corresponds to
$1,2, \ldots, n$, and each output wire of gates corresponds to $n+1, n+2, \ldots, n+q$. The last $m$ wires $n+q-m+1, n+q-m+2, \ldots, n+q$ correspond to the output wires of $C$. A gate $g$ is identified with the output wire of $g$, i.e., each gate also corresponds to $n+1, n+2, \ldots, n+q$.
- Each gate $g$ has two input wires: the left input wire of $g$ is $L(g)$ and the right input wire of $g$ is $R(g)$. We assume that $L(g) \leq R(g)<g$, i.e., the input wires $L(g), R(g)$ are smaller than $g$, and the left wire is smaller than or equal to the right wire. This restriction prevents the loop of the circuit.
- A wire $w$, which is not an output wire of $C$ is called the inner wire, i.e., each inner wire corresponds to $1,2, \ldots, n+q-m$. An inner wire is an input wire or an input wire of some gate. An inner wire $w$ can be branched, i.e., there might exist two or more gates having $w$ as its input wire, or some gate can be taken $w$ as the left and right input wires.
- For an inner wire $w, L^{-1}(w)$ is defined by the set of all gates whose left input wire is $w$, i.e., $L^{-1}(w)=\{g \in\{n+1, n+2, \ldots, n+q\} \mid L(g)=w\}$. We define $R^{-1}(w)$ in the same way.
- For a gate $g, G(g)$ represent the truth table of $g$. When $g$ computes a function $f:\{0,1\}^{2} \rightarrow\{0,1\}, G(g)$ represents a 4-bit binary string defined by

$$
G(g)=(f(0,0), f(0,1), f(1,0), f(1,1))
$$

In this paper, for simplicity, we assume that all gates are the NAND gates, i.e., $G(g)=(1,1,1,0)$ for all gates $g$. This is based on the fact that any Boolean function $f:\{0,1\}^{n} \rightarrow\{0,1\}^{m}$ can be constructed by only NAND gates. We note that our protocol can also be applied to a circuit with other gates.

Example of Boolean Circuit. A Boolean circuit $C=(3,1,3, L, R, G)$ is given in Figure 1, where the number of the input wires is $n=3$, the number of the output wires is $m=1$, and the number of the gates is $q=3$. Each input wire corresponds to $1,2,3$ and each gate corresponds to $4,5,6$. The functions $L$ and $R$ are defined by $L(4)=1, R(4)=2, L(5)=3, R(5)=4, L(6)=4$ and $R(6)=5$. Then we have $L^{-1}(1)=\{4\}, R^{-1}(1)=\emptyset, L^{-1}(2)=\emptyset, R^{-1}(2)=\{4\}, L^{-1}(3)=$ $\{5\}, R^{-1}(3)=\emptyset, L^{-1}(4)=\{6\}, R^{-1}(4)=\{5\}, L^{-1}(5)=\emptyset$ and $R^{-1}(5)=\{6\}$. Since all gates are the NAND gates, $G(g)=(1,1,1,0)$ for all $4 \leq g \leq 6$.

### 2.2 Card-based Protocols

In this paper, we use two-colored cards: the front side of a card is either \& or $\square$, and the back side is $\boldsymbol{?}$. All cards with the same suit are indistinguishable, and the backs of all cards are also indistinguishable.

In card-based protocols, three operations are used: permutation, shuffle, and turn. Let $k$ be the number of cards. A permutation operation (perm, $\pi$ ) for $\pi \in S_{k}$ is a deterministic operation that rearranges the order of the cards according to $\pi$. A shuffle operation (shuffle, $\Pi, \mathcal{F}$ ) for a subset $\Pi \subseteq S_{k}$ and a probability distribution $\mathcal{F}$ over $\Pi$ is a probabilistic operation that randomly rearranges the order of the cards according to a permutation $\pi \in \Pi$ drawn from $\mathcal{F}$. It is assumed


Fig. 1. An example of Boolean circuits
that no player can know which permutation $\pi$ is actually drawn from $\mathcal{F}$. A turn operation (turn, $T$ ) for $T \subseteq\{1,2, \ldots, k\}$ is a deterministic operation that turns over cards in $T$ from face-down to face-up or from face-up to face-down.

Let $S$ be a shuffle (shuffle, $\Pi, \mathcal{F}$ ). If $\mathcal{F}$ is a uniform distribution, $S$ is called a uniform shuffle. If $\Pi$ is a subgroup of $S_{k}, S$ is called a closed shuffle. If $S$ is uniform and closed, it is called a uniform closed shuffle.

### 2.3 Card-based Garbled Circuits

Shinagawa-Nuida [24] developed a card-based garbled circuit, which is a cardbased protocol based on garbled circuits. Tozawa et al. [27] improved the cardbased garbled circuit in terms of the number of cards. A card-based garbled circuit consists of three phases: initialization phase, garbling phase, and evaluation phase as follows:

Initialization phase: Given a Boolean circuit $f:\{0,1\}^{n} \rightarrow\{0,1\}^{m}$ and a sequence of input commitments to $x_{1}, x_{2}, \ldots, x_{n}$, it outputs a sequence of face-down cards $I$, which we call an initial state. The objective of this phase is to encode the circuit and its input into a sequence of face-down cards.
Garbling phase: Given an initial state $I$, it outputs two sequences of facedown cards $\widetilde{C}$ and $\widetilde{X}$, which we call a garbled circuit and a garbled input, respectively. The objective of this phase is to randomize the inputs and the intermediate values of the circuit without changing the functionality of the circuit.
Evaluation phase: Given a garbled circuit $\widetilde{C}$ and a garbled input $\widetilde{X}$, it outputs the output value or a commitment of the output value. The purpose of this phase is to obtain the output value by evaluating each gate of the garbled circuit $\widetilde{C}$ with the garbled input $\widetilde{X}$.

## 3 Our Single-Shuffle Protocols

### 3.1 Idea of Our Protocol

In many card-based protocols, 0 and 1 are represented by $Q$ and $\wp$, respectively, and [27] succeeded in realizing garbled circuits with eight cards that by encoding every $2 \times 2$ truth table with eight cards.

Here, we briefly describe our idea for realizing the garbled circuits with six cards that consist of three hearts and three clubs. For instance, we represent the truth table of the NAND and AND gates as follows, where 0 and 1 are represented by $\boldsymbol{\infty}$ and $\circlearrowleft$, respectively.


Facing down cards in the above truth table conceals all values in the truth table, but the negation of them is not possible due to the encoding rule with one card that prevents us from converting the NAND gate to the AND gate ${ }^{5}$. To overcome this obstacle, we append two to the NAND truth table as the third column and permute it. Then we have the following and by deleting the third column, we obtain the truth table of AND in a committed format.


### 3.2 Preliminaries for Our Protocol

In our protocol, each input wire is represented by two cards $\boldsymbol{\phi} \rho$ and each gate is represented by six cards $\boldsymbol{\&} 0$ gates, we use $2 n+6 q$ cards in total.

To clarify the position of the cards, we define $2 n+6 q$ indices: $P_{i}[a](1 \leq$ $i \leq n$ and $a \in\{0,1\})$ and $P_{g}[b][c](n+1 \leq g \leq n+q, b \in\{0,1\}$, and $c \in$ $\{0,1,2\})$. Two indices $P_{i}[0], P_{i}[1]$ correspond to the input wire $i$ and six indices $P_{g}[0][0], P_{g}[1][0], P_{g}[0][1], P_{g}[1][1], P_{g}[0][2], P_{g}[1][2]$ correspond to the gate $g$. We assume that all indices are distinct. We give an example of distinct $2 n+6 q$ indices in the following:
$-P_{i}[a]=2 i-1+a$ for $1 \leq i \leq n$ and $a \in\{0,1\} ;$
$-P_{g}[b][c]=2 n+1+6(g-(n+1))+3 b+c$ for $n+1 \leq g \leq n+q, b \in\{0,1\}$, and $c \in\{0,1,2\}$.
The above indices are consecutive from 1 to $2 n+6 q$.

[^15]
### 3.3 Initialization Phase

Given a Boolean circuit $f:\{0,1\}^{n} \rightarrow\{0,1\}^{m}$ and a sequence of input commitments to $x_{1}, x_{2}, \ldots, x_{n}$, the initialization phase makes a sequence of face-down cards on the indices of the position $P_{i}[a]$ and $P_{g}[b][c]$.

First, the sequence of input commitments is arranged as follows:


For each gate $g \in\{n+1, n+2, \ldots, n+q\}$, we identify the indices $P_{g}[a][b]$ ( $a \in\{0,1\}$ and $b \in\{0,1,2\}$ ) with the cells of a $2 \times 3$ matrix as follows:

$$
\begin{array}{|l|l|l|}
\hline P_{g}[0][0] & P_{g}[0][1] & P_{g}[0][2] \\
\hline P_{g}[1][0] & P_{g}[1][1] & P_{g}[1][2] \\
\hline
\end{array}
$$

Then, we place six cards \& $\& \leftrightarrow \leftrightarrow \infty \mid O$ as follows:


We note that the above matrix represents the NAND gate: for two inputs $a, b \in$ $\{0,1\}$, the card on $P_{g}[a][b]$ is if $a=b=1$ and $\cap$ otherwise. It can be regarded as the NAND gate by $\boldsymbol{\varphi}=0, \odot=1$. We also note that two additional $\boldsymbol{\wp}$ s on $P_{g}[0][2], P_{g}[1][2]$ are needed to garble the gate as explained in Section 3.1.

Then, we apply a turn operation so that all cards are face-down. Now we have a sequence of $2 n+6 q$ face-down cards on the indices of the position $P_{i}[a]$ and $P_{g}[b][c]$. This is the output of this phase.

### 3.4 Garbling Phase

Next, the protocol proceeds to the garbling phase. This phase just applies a uniform shuffle (shuffle, $\Pi, \mathcal{F}$ ) to the sequence of $2 n+6 q$ cards outputted by the initialization phase. In the following, we will define $\Pi \subseteq S_{2 n+6 q}$ by three steps: (1) defining four permutations, (2) defining a shuffle for randomizing a wire, and (3) composing all shuffles.

Defining Four Permutations. For an input wire $i \in\{1,2, \ldots, n\}$, a permutation $\pi_{i}$ is defined by

$$
\pi_{i}:=\left(P_{i}[0], P_{i}[1]\right) .
$$

It represents the bit flip of the $i$-th input commitment. For a gate $g \in\{n+1, n+$ $2, \ldots, n+q\}$, a permutation $\pi_{g}$ is defined by

$$
\pi_{g}:=\left(P_{g}[0][0], P_{g}[0][2]\right) \circ\left(P_{g}[1][0], P_{g}[1][2]\right) \circ\left(P_{g}[0][1], P_{g}[1][1]\right) .
$$

It represents the bit flip of the truth table of $g$ as follows:

For a gate $g$, a permutation $\tau_{g}$ is defined by

$$
\tau_{g}:=\left(P_{g}[0][0], P_{g}[1][0]\right) \circ\left(P_{g}[0][1], P_{g}[1][1]\right) .
$$

It represents a swap of the rows of the truth table of $g$ as follows:

$$
\begin{array}{|l|l|l|l|l|l|}
\hline 1 & 3 & 5 \\
\hline 2 & 4 & 6 \\
\hline
\end{array} \xrightarrow{\tau_{g}} \begin{array}{|l|l|l|}
\hline 2 & 4 & 5 \\
\hline 1 & 3 & 6 \\
\hline
\end{array} .
$$

A permutation $\sigma_{g}$ is defined by

$$
\sigma_{g}:=\left(P_{g}[0][0], P_{g}[0][1]\right) \circ\left(P_{g}[1][0], P_{g}[1][1]\right) .
$$

It represents a swap of the columns of the truth table of $g$ as follows:

Defining a Shuffle for Randomizing a Wire. For a wire $w \in\{1,2, \ldots, n+$ $q\}$, a permutation $\widehat{\pi}_{w}$ is defined by

$$
\widehat{\pi}_{w}:=\pi_{w} \circ \prod_{g \in L^{-1}(w)} \tau_{g} \circ \prod_{g^{\prime} \in R^{-1}(w)} \sigma_{g^{\prime}} .
$$

By applying it, the value of the wire $w$ is flipped and for each gate $g$, the rows of $g$ are swapped if $w$ is the left input wire of $g$ and the columns of $g$ are swapped if $w$ is the right input wire of $g$. Define $\Pi_{w}:=\left\{\right.$ id, $\left.\widehat{\pi}_{w}\right\}$. A uniform shuffle (shuffle, $\Pi_{w}, \mathcal{F}_{w}$ ) is a shuffle that randomizes the value of the wire $w$ and all gates having $w$ as input.

Composing All Shuffles. The subset $\Pi \subseteq S_{n+q}$ is defined by

$$
\Pi:=\Pi_{1} \Pi_{2} \cdots \Pi_{n+q-m}=\left\{\pi_{1}^{\prime} \circ \pi_{2}^{\prime} \circ \cdots \pi_{n+q-m}^{\prime} \mid \pi_{i}^{\prime} \in \Pi_{i}\right\} .
$$

The uniform shuffle (shuffle, $\Pi, \mathcal{F}$ ) is now obtained. We note that it is a shuffle by composing $n+q$ uniform shuffles (shuffle, $\Pi_{w}, \mathcal{F}_{w}$ ).

### 3.5 Evaluation Phase

Finally, the protocol proceeds to the evaluation phase. In this phase, the players evaluate the circuit by opening cards as follows. Let $v_{w}(1 \leq i \leq n+q)$ be an indeterminate on $\{0,1\}$. The protocol proceeds by determining these values and finally outputs $v_{n+q-m+1}, v_{n+q-m+2}, \ldots, v_{n+q}$ as the output values.

First, the players open all cards corresponding to the input wires: the value of $v_{i}(1 \leq i \leq n)$ is set to the value of the $i$-th commitment according to the encoding rule $\boldsymbol{\&} O=0$ and $\varphi=1$. Next, each gate $g=n+1, \ldots, n+q$ is evaluated (in this order) by opening the card on the position $P_{g}\left[v_{L(g)}\right]\left[v_{R(g)}\right]$ : the value of $v_{g}$ is set to the value of the card according to the encoding rule $\boldsymbol{\infty}=0$ and $\wp=1$. Note that the values of $v_{L(g)}$ and $v_{R(g)}$ are determined before $g$ is executed since $L(g) \leq R(g)<g$. By repeating this process, we finally obtain the output values $v_{n+q-m+1}, v_{n+q-m+2}, \ldots, v_{n+q}$.

### 3.6 Description of Our Protocol in the Non-committed Format

We summarize our protocol in the following.

1. First, we enter the initialization phase. Given a Boolean circuit $f$ and the input commitments to $x_{1}, \ldots, x_{n}$, this phase outputs a sequence of $6 q+2 n$ face-down cards as an initial state.
2. Next, we enter the garbling phase. Given an initial state, this phase applies a shuffle (shuffle, $\Pi, \mathcal{F}$ ) defined in Section 3.4. We regard the resulting sequence of $2 n$ cards corresponding to the input commitments as the garbling input and the remaining sequence of $6 q$ cards as the garbled circuit.
3. Finally, we enter the evaluation phase. This phase opens the garbled input and some cards of the garbled circuit. We output a $m$-bit string corresponding to the cards of the output gates.

In the following, we prove the correctness and security of our protocol.
Correctness: Recall that for each wire $1 \leq w \leq n+q-m$, the permutation $\widehat{\pi}_{w}$ is defined as follows:

$$
\widehat{\pi}_{w}:=\pi_{w} \circ \prod_{g \in L^{-1}(w)} \tau_{g} \circ \prod_{g^{\prime} \in R^{-1}(w)} \sigma_{g^{\prime}}
$$

Let $w$ be a wire and $g$ be a gate such that $L(g)=w$ (resp., $R(g)=w$ ). From the definition of $\widehat{\pi}_{w}$, we can observe that the bit flip introduced by $\widehat{\pi}_{w}$ and the swap of columns (resp., rows) introduced by $\tau_{g}$ (resp., $\sigma_{g^{\prime}}$ ) is synchronized, which guarantees that the functionality of the circuit remains the same. Therefore, this protocol is correct.
Security: In order to prove the security, it is sufficient to show that the opened values $v_{i}(1 \leq i \leq n+q-m)$ except the output values are independently and uniformly random bits. (We note that once this fact is proven, a simulator can be constructed in the same way as Shinagawa-Nuida [24].) In the following, we prove this fact by reverse induction from $n+q-m$ to 1 .

Let $A_{w}$ be the shuffle for randomizing a wire $w$, i.e., $A_{w}:=\left(\right.$ shuffle, $\left.\Pi_{w}, \mathcal{F}_{w}\right)$. First, $v_{n+q-m}$ is a uniformly random bit due to the effect of uniform shuffles $A_{n+q-m}$ and $\left\{A_{w} \mid w \in L^{-1}(n+q-m) \cup R^{-1}(n+q-m)\right\}$. Next, suppose that $v_{i+1}, v_{i+2}, \ldots, v_{n+q-m}$ are independently and uniformly random bits.

The uniform property of $v_{i}$ is obvious due to the effect of uniform shuffles $A_{i}$ and $\left\{A_{w} \mid w \in L^{-1}(i) \cup R^{-1}(i)\right\}$. Since $A_{i}$ does not appear in the wires greater than $i$, the randomness introduced by the shuffle $A_{i}$ is independent from $v_{i+1}, v_{i+2}, \ldots, v_{n+q-m}$. Thus $v_{i}, v_{i+1}, \ldots, v_{n+q-m}$ are also independently and uniformly random bits.

Therefore, $v_{i}(1 \leq i \leq n+q-m)$ are independently and uniformly random bits. This proves the security.

### 3.7 Our Protocol in the Committed Format

Although our protocol in Section 3.6 is a non-committed-format protocol, we can convert it to a committed-format protocol by appending $2 m$ additional cards, where $m$ is the number of the output wires. The committed-format protocol is the same as our -committed-format protocol except that for each output gate $g \in\{n+q-m+1, \ldots, n+q\}$, we use the eight-card truth table as in Tozawa et al. [27] instead of our six-card truth table. More concretely, we use a truth table of an output gate $g$ as follows:


The shuffle in the committed-format protocol can be defined in the same way as in Section 3.4. By applying it, we obtain a committed-format protocol. Since each output gate requires two additional cards, the number of cards in this protocol is $6 q+2 n+2 m$.

## 4 Conclusion

This paper proposed new single-shuffle card-based protocols for any Boolean circuit. Our protocols are based on Yao's garbled circuit as in previous singleshuffle protocols [24,27]. Namely, the truth tables of gates in the Boolean circuit are garbled (or randomized) while keeping the output of the circuit consistent. Our core technique to reduce the number of cards is to propose a new encoding of the truth table: each value of the truth table is represented by one card, whereas the previous works used two cards per value. We also used two additional cards to apply Yao's technique to our protocol. Therefore, our protocols require only six cards per gate. Specifically, we proposed a non-committed single-shuffle cardbased protocol with $6 q+2 n$ cards and then modified it to make it a committed protocol with $2 m$ additional cards. Since our protocols require uniform shuffles, it would be interesting to construct a committed card-based protocol with single uniform closed shuffles and a comparable number of cards to ours.

Acknowledgment This work was supported by JSPS KAKENHI Grant Numbers JP23H00468, JP23H00479, JP23K17455, JP23K16880, JP22H03590, JP21K17702, JP21H03395, JP21H03441, JP18H05289, JST CREST JPMJCR22M1, JPMJCR23M2, and MEXT Leading Initiative for Excellent Young Researchers.

## References

1. Abe, Y., Hayashi, Y., Mizuki, T., Sone, H.: Five-card AND protocol in committed format using only practical shuffles. In: 5th ACM on ASIA PublicKey Cryptography Workshop. pp. 3-8. APKC '18, ACM, New York (2018), https://doi.org/10.1145/3197507.3197510
2. Abe, Y., Hayashi, Y., Mizuki, T., Sone, H.: Five-card AND computations in committed format using only uniform cyclic shuffles. New Gener. Comput. 39(1), 97114 (2021), https://doi.org/10.1007/s00354-020-00110-2
3. den Boer, B.: More efficient match-making and satisfiability: The Five Card Trick. In: Advances in Cryptology - EUROCRYPT 1989, Workshop on the Theory and Application of of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, Proceedings. pp. 208-217 (1989)
4. Crépeau, C., Kilian, J.: Discreet solitary games. In: Advances in Cryptology CRYPTO' 93. pp. 319-330. Springer Berlin Heidelberg (1994)
5. Kastner, J., Koch, A., Walzer, S., Miyahara, D., Hayashi, Y., Mizuki, T., Sone, H.: The minimum number of cards in practical card-based protocols. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology—ASIACRYPT 2017. LNCS, vol. 10626, pp. 126-155. Springer, Cham (2017), https://doi.org/10.1007/978-3-319-70700-6_5
6. Koch, A.: The landscape of security from physical assumptions. In: IEEE Information Theory Workshop. pp. 1-6. IEEE, NY (2021), https://doi.org/10.1109/ITW48936.2021.9611501
7. Koch, A., Walzer, S.: Foundations for actively secure card-based cryptography. In: 10th International Conference on Fun with Algorithms (FUN 2021). Ed.: M. Farach-Colton. Leibniz International Proceedings in Informatics (LIPIcs), vol. 157, pp. Art.-Nr.: 17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik GmbH (LZI) (2020). https://doi.org/10.4230/LIPIcs.FUN. 2021.17
8. Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: Advances in Cryptology - ASIACRYPT 2015. pp. 783807. Springer Berlin Heidelberg (2015)
9. Koyama, H., Toyoda, K., Miyahara, D., Mizuki, T.: New card-based copy protocols using only random cuts. In: ASIA Public-Key Cryptography Workshop. pp. 13-22. ACM, NY (2021), https://doi.org/10.1145/3457338.3458297
10. Kuzuma, T., Isuzugawa, R., Toyoda, K., Miyahara, D., Mizuki, T.: Card-based single-shuffle protocols for secure multiple-input AND and XOR computations. In: APKC '22: Proceedings of the 9th ACM on ASIA Public-Key Cryptography Workshop, APKC@AsiaCCS 2022, Nagasaki, Japan, 30 May 2022. pp. 51-58. ACM (2022)
11. Miyamoto, K., Shinagawa, K.: Graph automorphism shuffles from pile-scramble shuffles. New Gener. Comput. 40, 199-223 (2022), https://doi.org/10.1007/s00354-022-00164-4
12. Mizuki, T., Sone, H.: Six-card secure AND and four-card secure XOR. In: Frontiers in Algorithmics, Third International Workshop, FAW 2009, Hefei, China, June 2023, 2009. Proceedings. pp. 358-369 (2009)
13. Mizuki, T., Uchiike, F., Sone, H.: Securely computing XOR with 10 cards. The Australasian Journal of Combinatorics 36, 279-293 (2006)
14. Niemi, V., Renvall, A.: Secure multiparty computations without computers. Theor. Comput. Sci. 191(1-2), 173-183 (1998), http://dx.doi.org/10.1016/S0304-3975(97)00107-2
15. Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols for any boolean function. In: Theory and Applications of Models of Computation - 12th Annual Conference, TAMC 2015, Singapore, May 18-20, 2015, Proceedings. pp. 110-121 (2015). https://doi.org/10.1007/978-3-319-17142-5•11
16. Nishida, T., Hayashi, Y.i., Mizuki, T., Sone, H.: Card-based protocols for any boolean function. In: Jain, R., Jain, S., Stephan, F. (eds.) Theory and Applications of Models of Computation. pp. 110-121. Springer International Publishing, Cham (2015)
17. Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: An implementation of nonuniform shuffle for secure multi-party computation. In: ACM International Workshop on ASIA Public-Key Cryptography. pp. 49-55. AsiaPKC '16, ACM, New York (2016), https://doi.acm.org/10.1145/2898420.2898425
18. Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: Pile-shifting scramble for card-based protocols. IEICE Trans. Fundam. 101(9), 1494-1502 (2018), https://doi.org/10.1587/transfun.E101.A. 1494
19. Nishimura, A., Nishida, T., ichi Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols using unequal division shuffles. Soft Computing 22, 361-371 (2018)
20. Nishimura, A., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Five-card secure computations using unequal division shuffle. In: Dediu, A.H., Magdalena, L., Martín-Vide, C. (eds.) Theory and Practice of Natural Computing. LNCS, vol. 9477, pp. 109-120. Springer, Cham (2015), https://doi.org/10.1007/978-3-319-26841-5_9
21. Ruangwises, S., Itoh, T.: AND protocols using only uniform shuffles. In: van Bevern, R., Kucherov, G. (eds.) Computer Science-Theory and Applications. LNCS, vol. 11532, pp. 349-358. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-19955-5_30
22. Saito, T., Miyahara, D., Abe, Y., Mizuki, T., Shizuya, H.: How to implement a nonuniform or non-closed shuffle. In: Martín-Vide, C., Vega-Rodríguez, M.A., Yang, M.S. (eds.) Theory and Practice of Natural Computing. LNCS, vol. 12494, pp. 107-118. Springer, Cham (2020), https://doi.org/10.1007/978-3-030-63000-3_9
23. Shinagawa, K., Miyamoto, K.: Automorphism shuffles for graphs and hypergraphs and its applications. IEICE Trans. Fundam. E106.A(3), 306-314 (2023), https://doi.org/10.1587/transfun.2022CIP0020
24. Shinagawa, K., Nuida, K.: A single shuffle is enough for secure card-based computation of any boolean circuit. Discrete Applied Mathematics 289, 248-261 (2021)
25. Stiglic, A.: Computations with a deck of cards. Theor. Comput. Sci. 259(1-2), 671-678 (2001). https://doi.org/10.1016/S0304-3975(00)00409-6
26. Toyoda, K., Miyahara, D., Mizuki, T., Sone, H.: Six-card finite-runtime XOR protocol with only random cut. In: ACM Workshop on ASIA Public-Key Cryptography. pp. 2-8. APKC '20, ACM, New York (2020), https://doi.org/10.1145/3384940.3388961
27. Tozawa, K., Morita, H., Mizuki, T.: Single-shuffle card-based protocol with eight cards per gate. In: Unconventional Computation and Natural Computation. pp. 171-185. Springer Nature Switzerland, Cham (2023)
28. Ueda, I., Miyahara, D., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: Secure implementations of a random bisection cut. Int. J. Inf. Sec. 19(4), 445-452 (2020). https://doi.org/10.1007/s10207-019-00463-w
29. Ueda, I., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: How to implement a random bisection cut. In: Theory and Practice of Natural Computing - 5th International Conference, TPNC 2016, Sendai, Japan, December 12-13, 2016, Proceedings. pp. 58-69 (2016). https://doi.org/10.1007/978-3-319-49001-4.5
30. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: FOCS. pp. 162-167. IEEE Computer Society (1986), http://dblp.unitrier.de/db/conf/focs/focs86.html/Yao86

# 1-out-of- $\boldsymbol{n}$ Oblivious Signatures: Security Revisited and a Generic Construction with an Efficient Communication Cost* 

Masayuki Tezuka $\left.{ }^{(\boxed{ }}\right)$ and Keisuke Tanaka<br>Tokyo Institute of Technology, Tokyo, Japan<br>tezuka.m.ac@m.titech.ac.jp


#### Abstract

Chen (ESORIC 1994) is a protocol between the user and the signer. In this scheme, the user makes a list of $n$ messages and chooses the message that the user wants to obtain a signature from the list. The user interacts with the signer by providing this message list and obtains the signature for only the chosen message without letting the signer identify which messages the user chooses. Tso et al. (ISPEC 2008) presented a formal treatment of 1-out-of- $n$ oblivious signatures. They defined unforgeability and ambiguity for 1 -out-of- $n$ oblivious signatures as a security requirement. In this work, first, we revisit the unforgeability security definition by Tso et al. and point out that their security definition has problems. In particular, we point out that a trivial attack exists in their unforgeability security model and address this problem by modifying their security model and redefining unforgeable security. Second, we improve the generic construction of a 1-out-of-n oblivious signature scheme by Zhou et al. (IEICE Trans 2022). The bottleneck of their construction is the size of the communication cost. We reduce the communication cost by modifying their scheme with a Merkle tree. Then we prove the security of our modified scheme.


Keywords: 1-out-of- $n$ oblivious signatures • Generic construction • Roundoptimal • Merkle tree • Efficient communication cost

## 1 Introduction

### 1.1 Background

Oblivious Signatures. The notion of 1-out-of- $n$ oblivious signatures by Chen $[6]$ is an interactive protocol between a signer and a user. In an oblivious signature scheme, first, the user makes a list of $n$ messages $M=\left(m_{i}\right)_{i \in\{1, \ldots, n\}}$ and chooses one of message $m_{j}$ in $M$ that the user wants to obtain a signature. Then the user interacts with the signer by sending the list $M$ with a first message $\mu$ at the beginning of the interaction. The signer can see the candidate messages $M$ that

[^16]the user wants to get signed, but cannot identify which one of the messages in $M$ is chosen by the user. After completing the interaction with the signer, the user can obtain a signature $\sigma$ for only the chosen message $m_{j}$.

1 -out-of- $n$ oblivious signatures should satisfy ambiguity and unforgeability. Ambiguity prevents the signer from identifying which one of the messages the signer wants to obtain the signature in the interaction. Unforgeability requires that for each interaction, the user cannot obtain a signature of a message $m \notin M$ and can obtain a signature for only one message $m \in M$ where $M$ is a list of message that the user sends to the signer at the beginning of the interaction.

Oblivious signatures can be used to protect the privacy of users. Chen [6] explained an application of oblivious signatures as follows. The user will buy software from the seller and the signature from the seller is needed to use the software. However, information about which software the user is interested in may be sensitive at some stage. In this situation, by using oblivious signatures, the user can make a list of $n$ software and obtain a signature only for the one software that the user honestly wants to obtain without revealing it to the seller (signer). The oblivious signature can be used for e-voting systems [7,18].

Oblivious Signatures and Blind Signatures. Signatures with a similar flavor to oblivious signatures are blind signatures proposed by Chaum [5]. In a blind signature scheme, similar to an oblivious signature scheme, a user chooses a message and obtains a corresponding signature by interacting with the signer. Typically, blind signatures satisfy blindness and one-more unforgeability (OMUF). Blindness prevents the signer from linking a message/signature pair to the run of the protocol where it was created. OMUF security prevents the user from forging a new signature.

From the point of view of hiding the contents of the message, it may seem that blind signatures are superior than oblivious signatures. But compared to blind signatures, oblivious signature has merits listed as follows.

- Avoid Signing Disapprove Messages: In blind signatures, since the signer has no information about the message that the user wants to obtain the signature, the signer cannot prevent users from obtaining a signature on the message that the signer does not want to approve.
Partially blind signatures proposed by Abe and Fujisaki [1] mitigate this problem. This scheme allows the user and the signer to agree on a predetermined piece of common information info which must be included in the signed message. However, similar to blind signatures, the signer has no information for the blinded part of a message, partially blind signatures do not provide a full solution for the above problem.
By contrast, oblivious signatures allow the signer to view a list of messages. If the message that the signer does not want to approve is included in the message list, the signer can refuse to sign. Thus, the ambiguity of oblivious signatures provides a better solution for the above problem.
- Based on Weaker Assumptions: Recent works on blind signatures are dedicated to constructing efficient round-optimal (i.e., 2-move signing interaction) blind signature schemes $[2,4,8,9,10,11,12,13,14,15,16]$. However, these
schemes either rely on at least one of strong primitives, models, or assumptions such as pairing groups $[4,9,10,11,13,14]$, non-interactive zero-knowledge (NIZK) [2,8,15,16], the random oracle model (ROM) [8,14], the generic group model (GGM) [9], interactive assumptions [4,10,11,13], $q$-type assumptions [12], one-more assumptions [2], or knowledge assumptions [12].
By contrast, a generic construction of a round-optimal oblivious signature scheme without the ROM was proposed in the recent work by Zhou, Liu, and Han [21]. This construction uses a digital signature scheme and a commitment scheme. This leads to instantiations in various standard assumptions (e.g., DDH, DCR, Factoring, RSA, LWE) without the ROM. Thus, the round-optimal oblivious signature schemes can be constructed with weaker assumptions than round-optimal blind signature schemes.

Previous Works on Oblivious Signatures. The notion of oblivious signatures was introduced by Chen [6] and proposed 1-out-of-n oblivious signature schemes in the ROM. Following this seminal work, several 1-out-of- $n$ oblivious signature schemes have been proposed.

Tso, Okamoto, and Okamoto [19] formalized the syntax and security definition of the 1-out-of- $n$ oblivious signature scheme. They gave the efficient roundoptimal (i.e., 2-move) 1-out-of- $n$ oblivious signature scheme based on the Schnorr signature scheme. The security of this scheme can be proven under the DL assumption in the ROM.

Chiou and Chen [7] proposed a $t$-out-of- $n$ oblivious signature scheme. This scheme needs 3 rounds for a signing interaction and the security of this scheme can be proven under the RSA assumption in the ROM.

You, Liu, Tso, Tseng, and Mambo [20] proposed the lattice-based 1-out-of- $n$ oblivious signature scheme. This scheme is round-optimal and the security can be proven under the short integer solution (SIS) problem in the ROM.

In recent work by Zhou, Liu, and Han [21], a generic construction of a roundoptimal 1-out-of- $n$ oblivious signature scheme was proposed. Their scheme is constructed from a commitment scheme and a digital signature scheme without the ROM. By instantiating a signature scheme and commitment scheme from standard assumptions without the ROM, this generic construction leads 1-out-of- $n$ oblivious signature schemes from standard assumptions without the ROM. As far as we know, their scheme is the first generic construction of a 1-out-of-n oblivious signature scheme without the ROM.

### 1.2 Motivation

The security model for a 1-out-of- $n$ oblivious signature scheme is formalized by Tso [19]. Their security model is fundamental for subsequent works [21,20]. However, this security model has several problems. Here, we briefly review the unforgeability security model in [19] and explain the problems of their model. The formal description of this security game is given in Section 3.2

Definition of Unforgeability in [19]. Informally, the unforgeability for a 1 -out-of- $n$ oblivious signature scheme in [19] is defined by the following game.

Let $A$ be an adversary that executes a user part and tries to forge a new signature. A engages in the signing interaction with the signer. A can make any message list $M_{i}$ and any one message $m_{i, j_{i}} \in M_{i}$. Then, A engages the $i$-th signing interaction with $M_{i}$ at the beginning of the interaction. By interacting with the signer, A can obtain a signature $\sigma_{i}$ on a message $m_{i, j_{i}}$. Let $t$ be the number of signing interaction with the signer and $A$. Let $\mathbb{L}^{\text {Sign }}=\left\{m_{i, j_{i}}\right\}_{i \in\{1, \ldots, t\}}$ be all messages that A obtained signatures. A wins this game if A outputs a valid signature $\sigma^{*}$ on a message $m^{*} \notin \mathbb{L}^{\text {Sign }}$. A 1-out-of- $n$ oblivious signature scheme satisfies unforgeability if for all PPT adversaries A cannot win the above game in non-negligible probability.

However, the above security game has several problems listed below.

- Problem 1: How to Store Messages in $\mathbb{L}^{\text {Sign }}$ : In the above security game, we need to store corresponding messages that the signer obtains signatures. However, by ambiguity property, we cannot identify the chosen message $m_{i, j_{i}}$ that the signer wants to obtain a signature from a transcription of the $i$ th interaction with $M_{i}$. This problem can be addressed by forcing A to output ( $m_{i, j_{i}}, \sigma_{i}$ ) at the end of each interaction. However, the next problem is serious.
- Problem 2: Trivial Attack: One flaw is the existence of a trivial attack on the security game. Let us consider the following adversary A that runs signing protocol execution twice. A chooses $M=\left(m_{0}, m_{1}\right)$ where $m_{0}$ and $m_{1}$ are distinct, and sets lists as $M_{1}=M_{2}=M$. In the 1st interaction, A chooses $m_{0} \in M_{1}$, obtains a signature $\sigma_{0}$ on a message $m_{0}$, and outputs ( $m_{0}, \sigma_{0}$ ) at the end of interaction. In the 2nd interaction, A chooses $m_{1} \in M_{2}$, obtains a signature $\sigma_{1}$ on a message $m_{1}$, and outputs ( $m_{0}, \sigma_{0}$ ) at the end of interaction. Then, A outputs a trivial forgery $\left(m^{*}, \sigma^{*}\right)=\left(m_{1}, \sigma_{1}\right)$. This attack is caused by the reuse of a signature $\left(m_{0}, \sigma_{0}\right)$ at the end of the signing interaction. The unforgeability security models in previous works $[20,21]$ are based on the model by Tso et al. [19]. This trivial attack also works for these models as well. This fact invalidates unforgeability security proofs in $[6,19,20,21]$ for 1-out-of- $n$ oblivious signature scheme.
Note that we only claim that the security model in [19] has a flaw. We do not intend to claim that existing schemes in $[6,19,20,21]$ are insecure.
- Problem 3: Missing Adversary Strategy: The security game does not capture an adversary with the following strategy. Let us consider an adversary A that executes the signing protocol only once. A interacts with the signer with a message list $M$ and intends to a signature $\sigma^{*}$ on a message $m^{*} \notin M$, but give up outputting ( $m, \sigma$ ) where $m \in M$ at the end of signing interaction. Since the security game only considers the adversary that outputting $(m, \sigma)$ where $m \in M$ at the end of the signing execution, the security game cannot capture the adversary A give up outputting $(m, \sigma)$ where $m \in M$.


### 1.3 Our Contribution

The first contribution is providing a new security definition of the unforgeability security for a 1-out-of- $n$ oblivious signature scheme. We address the problems described in the previous section. We refer the reader to Section 3.3 for more detail on our definition of unforgeability security.

The second contribution is an improvement of a generic construction of 1-out-of- $n$ oblivious signature schemes by [20]. This round-optimal construction is obtained by a simple combination of a digital signature scheme and a commitment scheme. However, a bottleneck of this scheme is the communication cost (See Fig. 1).

| Scheme | $\mid \mathrm{vk}{ }^{\text {O5 }}$ | $\|\mu\|$ | $\|\rho\|$ | $\left\|\sigma^{05}\right\|$ |
| :---: | :---: | :---: | :---: | :---: |
| $\begin{gathered} \hline \mathrm{OS}_{\mathrm{zLH}} \\ {[21]} \\ \hline \end{gathered}$ | \|vk ${ }^{\text {DS }} \mid$ | $\left\|c^{\text {COM }}\right\|$ | $n\left\|\sigma^{\text {DS }}\right\|$ | $\left\|\sigma^{\mathrm{DS}}\right\|+\left\|c^{\text {COM }}\right\|+\left\|r^{\text {COM }}\right\|$ |
| $\begin{gathered} \text { OSOurs } \\ \$ 4.2 \\ \hline \end{gathered}$ | \|vk ${ }^{\text {DS }} \mid$ | $\left\|c^{\text {COM }}\right\|$ | $\left\|\sigma^{\text {DS }}\right\|$ | $\left\|\sigma^{\mathrm{DS}}\right\|+\left\|c^{\text {com }}\right\|+\left\|r^{\text {COM }}\right\|+\left(\left\lceil\log _{2} n\right\rceil+1\right) \lambda+\left\lceil\log _{2} n\right\rceil$ |

Fig. 1. Comparison with generic construction of 1-out-of- $n$ oblivious signature schemes. $\left|\mathrm{vk}^{\mathrm{OS}}\right|$ represents the bit length of the verification key, $|\mu|$ represents the bit length of the first communication, $|\rho|$ represents the bit length of the second communication, and $\left|\sigma^{\mathrm{OS}}\right|$ represents the bit length of the 1-out-of- $n$ oblivious signature scheme. In columns, $\lambda$ denotes a security parameter. $\left|c^{\mathrm{COM}}\right|$ (resp. $\left.\left|r^{\mathrm{COM}}\right|\right)$ denotes the bit length of a commitment (resp. randomness) and $\left|\sigma^{\mathrm{DS}}\right|$ (resp. $\left|\mathrm{vk}^{\mathrm{DS}}\right|$ ) denotes the bit length of a digital signature (resp. verification key) used to instantiate the 1 -out-of- $n$ oblivious signature scheme.

Particular, if the user interacts with the signer with a message list $M=$ $\left(m_{i}\right)_{i \in\{1, \ldots, n\}}$ and the first communication message $\mu$, then the signer sends $n$ digital signatures $\left(\sigma_{i}^{\text {DS }}\right)_{i \in\{1, \ldots, n\}}$ to the user as the second communication message where $\sigma_{i}^{\mathrm{DS}}$ is a signature on a message $\left(m_{i}, \mu\right)$. This means that the second communication message cost (size) is proportional to $n$.

We improve the second communication cost by using a Merkle tree. Concretely, instead of signing each $\left(m_{i}, \mu\right)$ where $m_{i} \in M$, we modify it to sign a message (root, $\mu$ ) where root is a root of the Merkle tree computed from $M$. By this modification, we reduce the communication cost of the second round from $n$ digital signatures to only one digital signature. As a side effect of our modification, the size of the obtained 1-out-of- $n$ oblivious signature is increasing, but it is proportional to $\log n$. Our modification has the merit that the sum of a second communication message size and a signature size is improved from $O(n)$ to $O(\log n)$.

### 1.4 Road Map

In Section 2, we introduce notations and review commitments, digital signatures, and Merkle tree. In Section 3, we review 1-out-of- $n$ oblivious signatures, revisit the definition of unforgeability by Tuo et al. [19], and redefine unforgeability. In Section 4, we give a generic construction of 1-out-of-n oblivious signature schemes with efficient communication cost by improving the construction by Zhou et al. [21] and prove security for our scheme. In Section 5, we conclude our result and discuss open problems.

## 2 Preliminaries

In this section, we introduce notations and review fundamental cryptographic primitives for constructing our 1-out-of- $n$ oblivious signature scheme.

### 2.1 Notations

Let $1^{\lambda}$ be the security parameter. A function $f$ is negligible in $k$ if $f(k) \leq$ $2^{-\omega(\log k)}$. For a positive integer $n$, we define $[n]:=\{1, \ldots, n\}$. For a finite set $S$, $s \stackrel{\$}{\leftarrow} S$ represents that an element $s$ is chosen from $S$ uniformly at random.

For an algorithm $\mathrm{A}, y \leftarrow \mathrm{~A}(x)$ denotes that the algorithm A outputs $y$ on input $x$. When we explicitly show that A uses randomness $r$, we denote $y \leftarrow$ $\mathrm{A}(x ; r)$. We abbreviate probabilistic polynomial time as PPT.

We use a code-based security game [3]. The game Game is a probabilistic experiment in which adversary A interacts with an implied challenger C that answers oracle queries issued by A. The Game has an arbitrary amount of additional oracle procedures which describe how these oracle queries are answered. When the game Game between the challenger C and the adversary A outputs $b$, we write $\mathrm{Game}_{\mathrm{A}} \Rightarrow b$. We say that A wins the game Game if $\mathrm{Game}_{\mathrm{A}} \Rightarrow 1$. We implicitly assume that the randomness in the probability term $\operatorname{Pr}\left[\operatorname{Game}_{\mathrm{A}} \Rightarrow 1\right]$ is over all the random coins in the game.

### 2.2 Commitment Scheme

We review a commitment scheme and its security notion.
Definition 1 (Commitment Scheme). A commitment scheme COM consists of a following tuple of algorithms (KeyGen, Commit).

- KeyGen $\left(1^{\lambda}\right)$ : A key-generation algorithm takes as an input a security parameter $1^{\lambda}$. It returns a commitment key ck. In this work, we assume that ck defines a message space, randomness space, and commitment space. We represent these space by $\mathcal{M}_{\mathrm{ck}}, \Omega_{\mathrm{ck}}$, and $\mathcal{C}_{\mathrm{ck}}$, respectively.
- Commit(ck, $m ; r$ ) : A commit algorithm takes as an input a commitment key ck, a message m, and a randomness $r$. It returns a commitment c. In this work, we use the randomness $r$ as the decommitment (i.e., opening) information for $c$.

Definition 2 (Computational Hiding). Let COM $=($ KeyGen, Commit) $a$ commitment scheme and A a PPT algorithm. We say that the COM satisfies computational hiding if for all A, the following advantage of the hiding game

$$
\begin{aligned}
& \operatorname{Adv}_{\mathrm{COM}, \mathrm{~A}}^{\text {Hide }}:= \\
& \qquad\left|\operatorname{Pr}\left[b=b^{*} \left\lvert\, \begin{array}{l}
\mathrm{ck} \leftarrow \operatorname{COM} . \operatorname{KeyGen}\left(1^{\lambda}\right),\left(m_{0}, m_{1}, \mathrm{st}\right) \leftarrow \mathrm{A}(\mathrm{ck}), \\
b \stackrel{\$}{\leftarrow}\{0,1\}, c^{*} \leftarrow \operatorname{COM} \cdot \operatorname{Commit}\left(\mathrm{ck}, m_{b}\right), b^{*} \leftarrow \mathrm{~A}\left(c^{*}, \mathrm{st}\right)
\end{array}\right.\right]-\frac{1}{2}\right|
\end{aligned}
$$

is negligible in $\lambda$.
Definition 3 (Strong Computational Binding). Let COM = (KeyGen, Commit) a commitment scheme and A a PPT algorithm. We say that the COM satisfies strong computational binding if the following advantage

$$
\begin{aligned}
& \operatorname{Adv}_{\text {COM }, A}^{\text {sBind }}:= \\
& \qquad \operatorname{Pr}\left[\left.\begin{array}{l}
\text { Commit }(\mathrm{ck}, m ; r)=\operatorname{Commit}\left(\mathrm{ck}, m^{\prime} ; r^{\prime}\right) \\
\wedge(m, r) \neq\left(m^{\prime}, r^{\prime}\right)
\end{array} \right\rvert\, \begin{array}{l}
\mathrm{ck} \leftarrow \operatorname{KeyGen}\left(1^{\lambda}\right), \\
\left((m, r),\left(m^{\prime}, r^{\prime}\right)\right) \leftarrow \mathrm{A}(\mathrm{ck})
\end{array}\right]
\end{aligned}
$$

is negligible in $\lambda$.
A commitment scheme with computational hiding and strong computational binding property can be constructed from a public key encryption (PKE) scheme with indistinguishable under chosen plaintext attack (IND-CPA) security. We refer the reader to [21] for a commitment scheme construction from a PKE scheme.

### 2.3 Digital Signature Scheme

We review a digital signature scheme and its security notion.
Definition 4 (Digital Signature Scheme). A digital signature scheme DS consists of following four algorithms (Setup, KeyGen, Sign, Verify).

- Setup $\left(1^{\lambda}\right): A$ setup algorithm takes as an input a security parameter $1^{\lambda}$. It returns the public parameter pp . In this work, we assume that pp defines a message space and represents this space by $\mathcal{M}_{\mathrm{pp}}$. We omit a public parameter pp in the input of all algorithms except for KeyGen.
- KeyGen(pp) : A key-generation algorithm takes as an input a public parameter pp. It returns a verification key vk and a signing key sk.
- Sign(sk, m) : A signing algorithm takes as an input a signing key sk and a message $m$. It returns a signature $\sigma$.
- Verify (vk, $m, \sigma$ ) : A verification algorithm takes as an input a verification key vk , a message $m$, and a signature $\sigma$. It returns a bit $b \in\{0,1\}$.

Correctness. DS satisfies correctness if for all $\lambda \in \mathbb{N}$, $\mathrm{pp} \leftarrow \operatorname{Setup}\left(1^{\lambda}\right)$ for all $m \in \mathcal{M}_{\mathrm{pp}},(\mathrm{vk}, \mathrm{sk}) \leftarrow \operatorname{KeyGen}(\mathrm{pp})$, and $\sigma \leftarrow \operatorname{Sign}(\mathrm{sk}, m)$, Verify $(\mathrm{vk}, m, \sigma)=1$ holds.

We review a security notion called the strong existentially unforgeable under chosen message attacks (sEUF-CMA) security for digital signature.

Definition 5 (sEUF-CMA Security). Let DS $=$ (Setup, KeyGen, Sign, Verify) be a signature scheme and A a PPT algorithm. The strong existentially unforgeability under chosen message attacks (sEUF-CMA) security for DS is defined by the sEUF-CMA security game Game ${ }_{\mathrm{DS}, \mathrm{A}}^{\mathrm{sEUFCMA}}$ between the challenger C and A in Fig. 2.

```
GAME Game \({ }_{\text {ESS,A }}^{\text {SEDFCMA }}\left(1^{\lambda}\right)\) :
    \(\mathbb{L}^{\text {Sign }} \leftarrow\{ \}, \operatorname{pp} \leftarrow \operatorname{Setup}\left(1^{\lambda}\right),(\mathrm{vk}, \mathrm{sk}) \leftarrow \operatorname{KeyGen}(\mathrm{pp}),\left(m^{*}, \sigma^{*}\right) \leftarrow \mathrm{A}^{\mathcal{O}^{\text {Sign }}(\cdot)}(\mathrm{pp}, \mathrm{vk})\)
    If Verify \(\left(\mathrm{vk}, m^{*}, \sigma^{*}\right)=1 \wedge\left(m^{*}, \sigma^{*}\right) \notin \mathbb{L}^{\text {Sign }}\), return 1 . Otherwise return 0.
Oracle \(\mathcal{O}^{\text {Sign }}(m)\) :
    \(\sigma \leftarrow \operatorname{Sign}(\mathbf{s k}, m), \mathbb{L}^{\text {Sign }} \leftarrow \mathbb{L}^{\text {Sign }} \cup\{(m, \sigma)\}\), return \(\sigma\).
```

Fig. 2. The sEUF-CMA security game Game ${ }_{\mathrm{DS}, \mathrm{A}}^{\mathrm{sEUFCMA}}$.

The advantage of an adversary A for the sEUF-CMA security game is defined by $\operatorname{Adv}_{\mathrm{DS}, \mathrm{A}}^{\mathrm{sEUFCMA}}:=\operatorname{Pr}\left[\operatorname{Gam}_{\mathrm{DS}, \mathrm{A}}^{\mathrm{sELFCMA}} \Rightarrow 1\right]$. DS satisfies sEUF-CMA security if for all PPT adversaries $\mathrm{A}, \operatorname{Adv}_{\mathrm{DS}, \mathrm{A}}^{\mathrm{sEUFCMA}}\left(1^{\lambda}\right)$ is negligible in $\lambda$.

### 2.4 Merkle Tree Technique

We review the collision resistance hash function family and the Merkle tree technique.

Definition 6 (Collision Resistance Hash Function Family). Let $\mathcal{H}=$ $\left\{H_{\lambda}\right\}$ be a family of hash functions where $H_{\lambda}=\left\{H_{\lambda, i}:\{0,1\}^{*} \rightarrow\{0,1\}^{\lambda}\right\}_{i \in \mathcal{I}_{\lambda}}$. $\mathcal{H}$ is a family of collision-resistant hash functions if for all PPT adversaries A , the following advantage

$$
\operatorname{Adv}_{\mathcal{H}, \mathrm{A}}^{\text {Coll }}\left(1^{\lambda}\right):=\operatorname{Pr}\left[H(x)=H\left(x^{\prime}\right) \mid H \stackrel{\$}{\leftarrow} H_{\lambda},\left(x, x^{\prime}\right) \leftarrow \mathrm{A}(H)\right]
$$

is negligible in $\lambda$.
Definition 7 (Merkle Tree Technique [17]). The Merkle tree technique MT consists of following three algorithms (MerkleTree, MerklePath, RootReconstruct) with access to a common hash function $H:\{0,1\}^{*} \rightarrow\{0,1\}^{\lambda}$.

- MerkleTree ${ }^{H}\left(M=\left(m_{0}, \ldots, m_{2^{k}-1}\right)\right):$ A Merkle tree generation algorithm takes as an input a list of $2^{k}$ elements $M=\left(m_{0}, \ldots, m_{2^{k}-1}\right)$. It constructs a complete binary tree whose height is $k+1$ (i.e., maximum level is $k$ ).
We represent a root node as $w_{\epsilon}$ and a node in level $\ell$ as $w_{b_{1}, \ldots b_{\ell}}$ where $b_{j} \in$ $\{0,1\}$ for $j \in[\ell]$. The leaf node with an index $i \in\left\{0, \ldots, 2^{k}-1\right\}$ represents $w_{12 \mathrm{~B}(i)}$ where 12 B is a conversion function from an integer $i$ to the $k$-bit binary representation.
Each leaf node with an index $i \in\left\{0, \ldots, 2^{k}-1\right\}$ (i.e., $\left.w_{12 \mathrm{~B}(i)}\right)$ is assigned a value $h_{12 \mathrm{~B}(i)}=H\left(m_{i}\right)$. Each level $j$ internal (non-leaf) node $w_{b_{1}, \ldots b_{j}}$ is assigned a value $h_{b_{1}, \ldots b_{j}}=H\left(h_{b_{1}, \ldots b_{j}, 0} \| h_{b_{1}, \ldots b_{j}, 1}\right)$ where $h_{b_{1}, \ldots b_{j}, 0}$ and $h_{b_{1}, \ldots b_{j}, 1}$ are values assigned to the left-children node $w_{b_{1}, \ldots b_{j}, 0}$ and the rightchildren node $w_{b_{1}, \ldots b_{j}, 1}$, respectively. The root node $w_{\epsilon}$ is assigned a hash value $h_{\epsilon}=H\left(h_{0} \| h_{1}\right)$ and denote this value as root. This algorithm outputs a value root and the description tree which describes the entire tree.
- MerklePath ${ }^{H}$ (tree, $\left.i\right)$ : A Merkle path generation algorithm takes as an input a description of a tree tree and a leaf node index $i \in\left\{0, \ldots, 2^{k}-1\right\}$. Then, this algorithm computes $\left(b_{1}, \ldots b_{k}\right)=12 \mathrm{~B}(i)$ and outputs a list path $=$ $\left(h_{\overline{b_{1}}}, h_{b_{1}, \overline{b_{2}}}, \ldots, h_{b_{1}, \ldots, \overline{b_{k}}}\right)$ where $\overline{b_{j}}=1-b_{j}$ for $j \in[k]$.
- RootReconstruct ${ }^{H}$ (path, $\left.m_{i}, i\right):$ A root reconstruction algorithm takes as an input a list path $=\left(h_{\overline{b_{1}}}, h_{b_{1}, \overline{b_{2}}}, \ldots, h_{b_{1}, \ldots, \overline{b_{k}}}\right)$, an element $m_{i}$, and a leaf node index $i \in\left\{0, \ldots, 2^{k}-1\right\}$. This algorithm computes $\left(b_{1}, \ldots b_{k}\right)=\mathrm{I} 2 \mathrm{~B}(i)$ and assigns $h_{b_{1}, \ldots b_{k}}$. For $i=k-1$ to 1 , computes $h_{b_{1}, \ldots b_{j}}=H\left(h_{b_{1}, \ldots b_{j}, 0} \| h_{b_{1}, \ldots b_{j}, 1}\right)$ and outputs root $=H\left(h_{0} \| h_{1}\right)$.

Lemma 1 (Collision Extractor for Merkle Tree). There exists the following efficient collision extractor algorithms $\mathrm{Ext}_{1}$ and $\mathrm{Ext}_{2}$.

- Ext ${ }_{1}$ takes as an input a description of Merkle tree tree whose root node is assigned value root and ( $m_{i}^{\prime}$, path, $i$ ). If tree is constructed from a list $M=\left(m_{0}, \ldots, m_{2^{k}-1}\right), m_{i} \neq m_{i}^{\prime}$, and root $=$ RootReconstruct ${ }^{H}\left(\right.$ path $\left., m_{i}, i\right)$ holds, it outputs a collision of the hash function $H$.
- Ext ${ }_{2}$ takes as an input a tuple ( $m, j$, path, path'). If RootReconstruct ${ }^{H}$ (path, $m, j)=$ RootReconstruct ${ }^{H}$ (path' $\left., m, j\right)$ and path $\neq$ path $^{\prime}$ hold, it outputs a collision of the hash function $H$.


## 3 Security of Oblivious Signatures Revisited

In this section, first, we review a definition of a 1-out-of- $n$ signature scheme and security notion called ambiguity. Next, we review the security definition of the unforgeability in [19] and discuss the flaws of their security model. Then, we redefine the unforgeability security for a 1-out-of- $n$ signature scheme.

## 3.1 (1, n)-Oblivious Signature Scheme

We review a syntax of a 1-out-of- $n$ oblivious signature scheme and the security definition of ambiguity.

Definition 8 (Oblivious Signature Scheme). a 1-out-of-n oblivious signature scheme ( $1, n$ )-OS consists of following algorithms (Setup, KeyGen, $\mathrm{U}_{1}, \mathrm{~S}_{2}, \mathrm{U}_{\text {Der }}$, Verify).

- Setup $\left(1^{\lambda}\right):$ A setup algorithm takes as an input a security parameter $1^{\lambda}$. It returns the public parameter pp . In this work, we assume that pp defines a message space and represents this space by $\mathcal{M}_{\mathrm{pp}}$. We omit a public parameter pp in the input of all algorithms except for KeyGen.
- KeyGen(pp) : A key-generation algorithm takes as an input a public parameter pp. It returns a verification key vk and a signing key sk.
- $\mathrm{U}_{1}\left(\mathrm{vk}, M=\left(m_{0}, \ldots, m_{n-1}\right), j\right):$ This is a first message generation algorithm that is run by a user. It takes as input a verification key vk , a list of message $M=\left(m_{0}, \ldots, m_{n-1}\right)$, and a message index $j \in\{0, \ldots, n-1\}$. It returns a pair of a first message and a state $(\mu, \mathrm{st})$ or $\perp$.
$-\mathrm{S}_{2}\left(\mathrm{vk}, \mathrm{sk}, M=\left(m_{0}, \ldots, m_{n-1}\right), \mu\right)$ : This is a second message generation algorithm that is run by a signer. It takes as input a verification key vk , a signing key sk, a list of message $M=\left(m_{0}, \ldots, m_{n-1}\right)$, and a first message $\mu$. It returns a second message $\rho$ or $\perp$.
- $\mathrm{U}_{\mathrm{Der}}(\mathrm{vk}, \mathrm{st}, \rho)$ : This is a signature derivation algorithm that is run by a user. It takes as an input a verification key vk , a state st, and a second message $\rho$. It returns a pair of a message and its signature $(m, \sigma)$ or $\perp$.
- Verify $(\mathrm{vk}, m, \sigma):$ A verification algorithm takes as an input a verification key vk , a message $m$, and a signature $\sigma$. It returns a bit $b \in\{0,1\}$.

Correctness. ( $1, n$ )-OS satisfies correctness if for all $\lambda \in \mathbb{N}$, $n \leftarrow n(\lambda)$, $\mathrm{pp} \leftarrow$ Setup $\left(1^{\lambda}\right)$, for all message set $\mathcal{M}=\left(m_{0}, \ldots, m_{n-1}\right)$ such that $m_{i} \in \mathcal{M}_{\mathrm{pp}}$, (vk, sk) $\leftarrow$ KeyGen(pp), for all $j \in\{0, \ldots n-1\}$, $(\mu, \mathrm{st}) \leftarrow \mathrm{U}_{1}(\mathrm{vk}, M, j), \rho \leftarrow$ $\mathrm{S}_{2}(\mathrm{vk}, \mathrm{sk}, M, \mu)$, and $\left(m_{j}, \sigma\right) \leftarrow \mathrm{U}_{\mathrm{Der}}(\mathrm{vk}, \mathrm{st}, \rho)$, Verify $\left(\mathrm{vk}, m_{j}, \sigma\right)=1$ holds.

Definition 9 (Ambiguity). Let $(1, n)$-OS $=\left(\right.$ Setup, KeyGen, $\mathrm{U}_{1}, \mathrm{~S}_{2}, \mathrm{U}_{\text {Der }}$, Verify $)$ be an oblivious signature scheme and A a PPT algorithm. The ambiguity for $(1, n)$-OS is defined by the ambiguity security game $\operatorname{Game}_{(1, n)-\mathrm{OS}, \mathrm{A}}^{\mathrm{Amb}}$ between the challenger C and A in Fig. 3.

```
GAME Game \({ }_{(1, n)-\mathrm{Os}, \mathrm{A}}^{\mathrm{Amb}}\left(1^{\lambda}\right)\) :
    \(\mathrm{pp} \leftarrow \operatorname{Setup}\left(1^{\lambda}\right),(\mathrm{vk}, \mathrm{sk}) \leftarrow \operatorname{KeyGen}(\mathrm{pp})\),
    \(\left(M=\left(m_{0}, \ldots, m_{n-1}\right), i_{0}, i_{1}, \mathrm{st}_{\mathrm{A}}\right) \leftarrow \mathrm{A}(\mathrm{pp}, \mathrm{vk}, \mathrm{sk})\)
    \(b \stackrel{\$}{\leftarrow}\{0,1\},\left(\mu, \mathrm{st}_{\mathrm{s}}\right) \leftarrow \mathrm{U}_{1}\left(\mathrm{vk}, M, i_{b}\right), b^{*} \leftarrow \mathrm{~A}\left(\mu, \mathrm{st}_{\mathrm{A}}\right)\).
    If \(b^{*}=b\) return 1 . Otherwise return 0 .
```

Fig. 3. The ambiguity security game $\operatorname{Game}_{(1, n)-\mathrm{Os}, \mathrm{A}}^{\mathrm{Amb}}$.

The advantage of an adversary A for the ambiguity security game is defined by $\operatorname{Adv}_{(1, n)-\mathrm{OS}, \mathrm{A}}^{\mathrm{Amb}}:=\left|\operatorname{Pr}\left[\operatorname{Game}_{(1, n)-\mathrm{OS}, \mathrm{A}}^{\mathrm{Amb}} \Rightarrow 1\right]-\frac{1}{2}\right| \cdot(1, n)$-OS satisfies ambiguity if for all PPT adversaries $\mathrm{A}, \operatorname{Adv}_{(1, n)-\mathrm{Os}, \mathrm{A}}^{\mathrm{Amb}}\left(1^{\lambda}\right)$ is negligible in $\lambda$.

### 3.2 Definition of Unforgeability Revisited

We review the security definition of unforgeability for $(1, n)$-OS in previous works in [19]. The unforgeability for a 1-out-of- $n$ oblivious signature scheme in [19] is formalized by the following game between a challenger C and a PPT adversary A.
$-C$ runs $p p \leftarrow \operatorname{Setup}\left(1^{\lambda}\right)$ and $(v k, s k) \leftarrow \operatorname{KeyGen}(p p)$, and gives (pp, vk) to $A$.

- A is allowed to engage polynomially many signing protocol executions. In an $i$-th protocol execution,
- A makes a list $\left.M_{i}=\left(m_{i, 0}, \ldots, m_{i, n-1}\right)\right)$ and chooses $m_{i, j_{i}}$.
- A sends $\left(\mu_{i}, M_{i}=\left(m_{i, 0}, \ldots, m_{i, n-1}\right)\right)$ to C.
- C runs $\rho_{i} \leftarrow \mathrm{~S}_{2}\left(\mathrm{vk}\right.$, sk, $\left.M_{i}, \mu_{i}\right)$ and gives $\rho_{i}$ to A.
- Let $\mathbb{L}^{\text {Sign }}$ be a list of messages that A obtained signatures. A outputs a forgery $\left(m^{*}, \sigma^{*}\right)$ which satisfies $m^{*} \notin \mathbb{L}^{\text {Sign }}$. A must complete all singing executions before it outputs a forgery.

If no PPT adversary A outputs a valid forgery in negligible probability in $\lambda$, $(1, n)$-OS satisfies the unforgeability security.

We point out three problems for the above security definition.

- Problem 1: How to Store Messages in $\mathbb{L}^{\text {Sign }}$ : In the above security game, we need to store corresponding messages that the signer obtains signatures. However, by ambiguity property, we cannot identify the message $m_{i, j_{i}}$ that is chosen by the signer from a transcription of the $i$-th interaction with $M_{i}$. This security model does not explain how to record an entry of $\mathbb{L}^{\text {Sign }}$.
- Problem 2: Trivial Attack: Let us consider the following adversary A that runs signing protocol execution twice. A chooses $M=\left(m_{0}, m_{1}\right)$ where $m_{0}$ and $m_{1}$ are distinct, and sets lists as $M_{1}=M_{2}=M$. In the 1st interaction, A chooses $m_{0} \in M_{1}$, obtains a signature $\sigma_{0}$ on a message $m_{0}$, and outputs ( $m_{0}, \sigma_{0}$ ) at the end of interaction. In the 2 nd interaction, A chooses $m_{1} \in M_{2}$, obtains a signature $\sigma_{1}$ on a message $m_{1}$, and outputs ( $m_{0}, \sigma_{0}$ ) at the end of interaction. Then, A outputs a trivial forgery $\left(m^{*}, \sigma^{*}\right)=\left(m_{1}, \sigma_{1}\right)$. This attack is caused by the reuse of a signature ( $m_{0}, \sigma_{0}$ ) at the end of the signing interaction.
- Problem 3: Missing Adversary Strategy: The security game does not capture an adversary with the following strategy. Let us consider an adversary A that executes the signing protocol only once. A interacts with the signer with a message list $M$ and intends to forge a signature $\sigma^{*}$ on a message $m^{*} \notin M$, but give up outputting $(m, \sigma)$ where $m \in M$ at the end of signing interaction. Since the security game only considers the adversary that outputting $(m, \sigma)$ where $m \in M$ at the end of the signing execution, the security game cannot capture the adversary A give up outputting $(m, \sigma)$ where $m \in M$ and forge ( $m^{*}, \sigma^{*}$ ) where $m^{*} \notin M$.


### 3.3 New Unforgeability Definition

To address the problems of the unforgeability security model by Tso et al. [19], we modify their security model and redefine the unforgeability security. Here, we briefly explain how to address these problems.

- Countermeasure for Problem 1: This problem is easy to fix by forcing A to output ( $m_{i, j_{i}}, \sigma_{i}$ ) at the end of each signing interaction.
- Countermeasure for Problem 2: This attack is caused by the reuse of a signature at the end of signing interactions. That is A submits $(m, \sigma)$ twice or more at the end of signing interactions.
To address this problem, we introduce the signature reuse check. This prevents resubmission of $(m, \sigma)$ at the end of signing interactions. However, this is not enough to prevent the reuse of a signature. If a signature has a re-randomizable property (i.e., The property that a signature is refreshed without the signing key), A can easily avoid resubmission and succeed in the trivial attack.
For this reason, normal unforgeability security is not enough. We address this issue by letting strong unforgeability security be a default for the security requirement.
- Countermeasure for Problem 3: This problem is addressed by adding another winning condition for A. When A submits $\left(m^{*}, \sigma^{*}\right)$ at the end of $i$-th signing interaction, if ( $m^{*}, \sigma^{*}$ ) is valid and $m^{*} \notin M_{i}$, A wins the game where $M_{i}$ is a list of messages send by A at the beginning of $i$-th signing interaction.

By reflecting the above countermeasures to the unforgeability security model by Tso et al. [19], we redefine the unforgeability security model as the strong unforgeability under chosen message attacks under the sequential signing interaction (Seq-sEUF-CMA) security.

Definition 10 (Seq-sEUF-CMA Security). Let ( $1, n$ )-OS = (Setup, KeyGen, $\mathrm{U}_{1}, \mathrm{~S}_{2}, \mathrm{U}_{\text {Der }}$, Verify) be a 1-out-of-n oblivious signature scheme and A a PPT algorithm. The strong unforgeability under chosen message attacks under the sequential signing interaction (Seq-sEUF-CMA) security for $(1, n)$-OS is defined by the Seq-sEUF-CMA security game $\operatorname{Game}_{(1, n)-\mathrm{OS}, \mathrm{A}}^{\text {Seq-SEUCMA }}$ between the challenger C and A in Fig. 4.

The advantage of an adversary A for the Seq-sEUF-CMA security game is defined by $\operatorname{Adv}_{(1, n)-\mathrm{OS}, \mathrm{A}}^{\text {Seq-SEUFCMA }}\left(1^{\lambda}\right):=\operatorname{Pr}\left[\operatorname{Game}_{(1, n)-\mathrm{OS}, \mathrm{A}}^{\text {Seq-SEUCMA }}\left(1^{\lambda}\right) \Rightarrow 1\right]$. $(1, n)$-OS satisfies Seq-sEUF-CMA security if for all PPT adversaries A, $\operatorname{Adv}_{(1, n)-\mathrm{OS}, \mathrm{A}}^{\text {Seq-EUFCMA }}\left(1^{\lambda}\right)$ is negligible in $\lambda$.

Our security model is the sequential signing interaction model. One may think that it is natural to consider the concurrent signing interaction model. However, by extending our model to the concurrent signing setting there is a trivial attack. We discuss the security model that allows concurrent signing interaction in Section 5.

## 4 Our Construction

In this section, first, we review the generic construction by Zhou et al. [21]. Second, we propose our new generic construction based on their construction. Then, we prove the security of our proposed scheme.

```
GAME Game \({ }_{(1, n)-\mathrm{OS}, \mathrm{A}}^{\text {Seq-SELCMA }}\left(1^{\lambda}\right)\) :
    \(\mathbb{L}^{\text {Sign }} \leftarrow\{ \}, \mathbb{L}^{\text {ListM }} \stackrel{\text { Lit }}{\leftarrow} \leftarrow, q^{\text {Sign }} \leftarrow 0, q^{\text {Fin }} \leftarrow 0\)
    \(\mathrm{pp} \leftarrow \operatorname{Setup}\left(1^{\lambda}\right),(\mathrm{vk}, \mathrm{sk}) \leftarrow \operatorname{KeyGen}(\mathrm{pp}),\left(m^{*}, \sigma^{*}\right) \leftarrow \mathrm{A}^{\mathcal{O}^{\text {Sign }}(\cdot, \cdot), \mathcal{O}^{\text {Fin }}(\cdot, \cdot)}(\mathrm{pp}, \mathrm{vk})\)
    If \(q^{\text {Sign }}=q^{\text {Fin }} \wedge \operatorname{Verify}\left(\mathrm{vk}, m^{*}, \sigma^{*}\right)=1 \wedge\left(m^{*}, \sigma^{*}\right) \notin \mathbb{L}^{\text {Sign }}\), return 1 .
    Otherwise return 0 .
Oracle \(\mathcal{O}^{\text {Sign }}\left(M_{q}{ }^{\text {sign }}, \mu\right)\) :
    If \(q^{\text {Sign }} \neq q^{\text {Fin }}\), return \(\perp\).
    \(\rho \leftarrow \mathrm{S}_{2}(\mathrm{vk}, \mathrm{sk}, M, \mu)\), if \(\rho=\perp\), return \(\perp\).
    If \(\rho \neq \perp, q^{\text {Sign }} \leftarrow q^{\text {Sign }}+1, \mathbb{L}^{\text {ListM }} \leftarrow \mathbb{L}^{\text {ListM }} \cup\left\{\left(q^{\text {Sign }}, M_{q^{\text {Sign }}}\right)\right\}\), return \(\rho\).
Oracle \(\mathcal{O}^{\text {Fin }}\left(m^{*}, \sigma^{*}\right)\) :
    If \(q^{\text {Sign }} \neq q^{\text {Fin }}+1\), return \(\perp\).
    If Verify \(\left(\mathrm{vk}, m^{*}, \sigma^{*}\right)=0\), return the game output 0 and abort.
    (//Oblivious signature reuse check)
    If \(\left(m^{*}, \sigma^{*}\right) \in \mathbb{L}^{\text {Sign }}\), return the game output 0 and abort.
    Retrieve an entry \(\left(q^{\text {Sign }}, M_{q \text { Sign }}\right) \in \mathbb{L}^{\text {ListM }}\).
    If \(m^{*} \in M_{q^{\text {Sign }}}, \mathbb{L}^{\text {Sign }} \leftarrow \mathbb{L}^{\text {Sign }} \cup\left\{\left(m^{*}, \sigma^{*}\right)\right\}, q^{\text {Fin }} \leftarrow q^{\text {Fin }}+1\), return "accept".
    (//Capture adversaries that give up completing signing executions in the game.)
    If \(m^{*} \notin M_{q^{\text {sign }}}\), return the game output 1 and abort.
```

Fig. 4. The Seq-sEUF-CMA security game $\operatorname{Game}_{(1, n)-\mathrm{OS}, \mathrm{A}}^{\text {Seq-sELECMA }}$. The main modifications from previous works security game are highlighted in white box.

### 4.1 Generic Construction by Zhou et al. [21]

The generic construction of a 1 -out-of- $n$ signature scheme $(1, n)-\mathrm{OS}_{\text {zLH }}$ by Zhou et al. [21] is a combination of a commitment scheme COM and a digital signature scheme DS. Their construction $(1, n)-\mathrm{OS}_{\mathrm{ZLH}}[\mathrm{COM}, \mathrm{DS}]=($ OS.Setup, OS.KeyGen, OS. U $U_{1}$, OS. $\mathrm{S}_{2}$, OS. $\mathrm{U}_{\text {Der }}$, OS.Verify) is given in Fig. 5.

We briefly provide an overview of a signing interaction and an intuition for the security of their construction. In the signing interaction, the user chooses a message list $M=\left(m_{i}\right)_{i \in\{0, \ldots, n-1\}}$ and a specific message $m_{j_{i}}$ that the user wants to obtain the corresponding signature. To hide this choice from the signer, the signer computes the commitment $c$ on $m_{j}$ with the randomness $r$. The user sends ( $M, \mu=c$ ) to the signer.

Here, we provide an intuition for the security of their construction. From the view of the signer, by the hiding property of the commitment scheme, the signer does not identify $m_{j}$ from $(M, \mu=c)$. This guarantees the ambiguity of their construction. The signer computes a signature $\sigma_{i}^{\text {DS }}$ on a tuple ( $m_{i}, c$ ) for $i \in\{0, \ldots, n-1\}$ and sends $\rho=\left(\sigma_{i}^{\mathrm{DS}}\right)_{i \in\{0, \ldots, n-1\}}$.

If the signer honestly computes $c$ on $m_{j} \in M$, we can verify that $m_{j_{i}}$ is committed into $c$ by decommitting with $r$. An oblivious signature on $m_{j}$ is obtained as $\sigma^{\mathrm{OS}}=\left(c, r, \sigma_{j}^{\mathrm{DS}}\right)$. If a malicious user wants to obtain two signatures for two distinct messages $m, m^{\prime} \in M$ or obtain a signature on $m^{*} \notin M$ from the signing protocol execution output $\left(M=\left(m_{i}\right)_{i \in\{0, \ldots, n-1\}}, \mu, \rho\right)$, the malicious

```
OS.Setup ( \(1^{\lambda}\) ):
    \(\mathrm{ck} \leftarrow \operatorname{COM} . \operatorname{KeyGen}\left(1^{\lambda}\right), \mathrm{pp}^{\mathrm{DS}} \leftarrow \mathrm{DS}\).Setup \(\left(1^{\lambda}\right)\), return \(\mathrm{pp}^{\mathrm{OS}} \leftarrow\left(\mathrm{ck}, \mathrm{pp}^{\mathrm{DS}}\right)\).
OS. KeyGen(pp \(\left.{ }^{\text {OS }}=\left(c k, p^{\text {DS }}\right)\right)\) :
    \(\left(\mathrm{vk}^{\mathrm{DS}}, \mathrm{sk}^{\mathrm{DS}}\right) \leftarrow \mathrm{DS}\). KeyGen \(\left(\mathrm{pp}^{\mathrm{DS}}\right)\), return \(\left(\mathrm{vk}^{\mathrm{OS}}, \mathrm{sk}^{\mathrm{OS}}\right) \leftarrow\left(\mathrm{vk}^{\mathrm{DS}}, \mathrm{sk}^{\mathrm{DS}}\right)\).
OS. \(\mathrm{U}_{1}\left(\mathrm{vk}^{\mathrm{OS}}, M=\left(m_{0}, \ldots, m_{n-1}\right), j \in\{0, \ldots, n-1\}\right):\)
    \(r \stackrel{\&}{\leftarrow} \Omega_{\mathrm{ck}}, c \leftarrow \operatorname{COM}\).Commit(ck, \(\left.m ; r\right), \mu \leftarrow c\), st \(\leftarrow(M, c, r, j)\), return \((\mu\), st) .
OS. \(\mathrm{S}_{2}\left(\mathrm{vk}^{\mathrm{OS}}, \mathrm{sk}^{\mathrm{OS}}=\mathrm{sk}^{\mathrm{DS}}, M=\left(m_{0}, \ldots, m_{n-1}\right), \mu=c\right):\)
    For \(i=0\) to \(n-1, \sigma_{i}^{\mathrm{DS}} \leftarrow \mathrm{DS} . \operatorname{Sign}\left(\mathrm{sk}^{\mathrm{DS}},\left(m_{i}, c\right)\right)\).
    Return \(\rho \leftarrow\left(\sigma_{0}^{\text {DS }}, \ldots, \sigma_{n-1}^{\text {DS }}\right)\).
OS. \(U_{\text {Der }}\left(\mathrm{vk}^{\mathrm{OS}}=\mathrm{vk}^{\mathrm{DS}}\right.\), st \(\left.=\left(M=\left(m_{0}, \ldots, m_{n-1}\right), c, r, j\right), \rho=\left(\sigma_{0}^{\mathrm{DS}}, \ldots, \sigma_{n-1}^{\mathrm{DS}}\right)\right):\)
    For \(i=0\) to \(n-1\), if DS.Verify \(\left(\mathrm{vk}^{\mathrm{DS}},\left(m_{i}, c\right), \sigma_{i}^{\mathrm{DS}}\right)=0\), return \(\perp\).
    \(\sigma^{\mathrm{OS}} \leftarrow\left(c, r, \sigma_{j}^{\mathrm{DS}}\right)\), return \(\left(m_{j}, \sigma^{\mathrm{OS}}\right)\).
OS.Verify \(\left(\mathrm{vk}^{\mathrm{OS}}=\mathrm{vk}^{\mathrm{DS}}, m, \sigma^{\mathrm{OS}}=\left(c, r, \sigma^{\mathrm{DS}}\right):\right.\)
    If \(c \neq\) COM.Commit(ck, \(m ; r\) ), return 0 .
    If DS.Verify \(\left(\mathrm{vk}^{\mathrm{DS}},(m, c), \sigma^{\mathrm{DS}}\right)=0\), return 0 .
    Otherwise return 1.
```

Fig. 5. The generic construction $(1, n)-\mathrm{OS}_{\text {zLH }}[\mathrm{COM}, \mathrm{DS}]$.
user must break either the EUF-CMA security of DS or the binding property of COM. This guarantees the unforgeability security of their construction.

A drawback of their construction is the second communication cost. A second message $\rho$ consists of $n$ digital signatures. If $n$ becomes large, it will cause heavy communication traffic. It is desirable to reduce the number of signatures in $\rho$.

### 4.2 Our Generic Construction

As explained in the previous section, the drawback of the construction by Zhou et al. [21] is the size of a second message $\rho$. To circumvent this bottleneck, we improve their scheme by using a Merkle tree technique. Concretely, instead of signing on ( $m_{i}, c$ ) for each $m_{i} \in M$, we modify it to sign on (root, $c$ ) where root is a root of the Merkle tree computed from $M$. This modification allows us to reduce the number of digital signatures included in $\rho$ from $n$ to 1 .

Now, we describe our construction. Let COM be a commitment scheme, DS a digital signature scheme, $\mathcal{H}=\left\{H_{\lambda}\right\}$ a hash function family, and $\mathrm{MT}=$ (MerkleTree, MerklePath, RootReconstruct) a Merkle tree technique in Section 2.4. To simplify the discussion, we assume that $n>1$ is a power of $2 .^{1}$

Our generic construction $(1, n)-$ OS $_{\text {Ours }}[\mathcal{H}, \mathrm{COM}, \mathrm{DS}]=$ (OS.Setup, OS.KeyGen, OS. $\mathrm{U}_{1}$, OS. $\mathrm{S}_{2}$, OS. $\mathrm{U}_{\text {Der }}$, OS.Verify) is given in Fig. 6.

[^17]```
OS.Setup(1^):
    H}\stackrel{&}{\leftarrow}\mp@subsup{H}{\lambda}{},\textrm{ck}\leftarrow\operatorname{COM.KeyGen(1)
    Return pp }\mp@subsup{}{}{\textrm{OS}}\leftarrow(H,\textrm{ck},\mp@subsup{\textrm{pp}}{}{\textrm{DS}})
OS.KeyGen(pp OS = (ck, pp DS ) :
    (\mp@subsup{\textrm{vk}}{}{\textrm{DS}},\mp@subsup{\textrm{sk}}{}{\textrm{DS}})\leftarrow\textrm{DS}.KeyGen(p\mp@subsup{p}{}{\textrm{DS}}), return (\mp@subsup{\textrm{vk}}{}{\textrm{OS}},\mp@subsup{\textrm{sk}}{}{\textrm{OS}})\leftarrow(\mp@subsup{\textrm{vk}}{}{\textrm{DS}},\mp@subsup{\textrm{sk}}{}{\textrm{DS}}).
OS.U⿴囗⿱一一⿱⿴囗十丌
    If there exists }(t,\mp@subsup{t}{}{\prime})\in{0,\ldots,n-1\mp@subsup{}}{}{2}\mathrm{ s.t. }t\not=\mp@subsup{t}{}{\prime}\wedge\mp@subsup{m}{t}{}=\mp@subsup{m}{\mp@subsup{t}{}{\prime}}{\prime}\mathrm{ , return }\perp\mathrm{ .
    r&
OS.S ( (vk }\mp@subsup{}{}{\textrm{OS}},\mp@subsup{\textrm{sk}}{}{\textrm{OS}}=\mp@subsup{\textrm{sk}}{}{\textrm{DS}},M=(\mp@subsup{m}{0}{},\ldots,\mp@subsup{m}{n-1}{}),\mu=c)
    If there exists }(t,\mp@subsup{t}{}{\prime})\in{0,\ldots,n-1\mp@subsup{}}{}{2}\mathrm{ s.t. }t\not=\mp@subsup{t}{}{\prime}\wedge\mp@subsup{m}{t}{}=\mp@subsup{m}{\mp@subsup{t}{}{\prime}}{\prime}\mathrm{ , return }\perp\mathrm{ .
    (root, tree)}\leftarrow\mp@subsup{M}{\mathrm{ MerkleTree }}{}\mp@subsup{}{}{H}(M),\mp@subsup{\sigma}{}{\textrm{DS}}\leftarrow\textrm{DS}.Sign(\mp@subsup{sk}{}{\textrm{DS}},(\mathrm{ root, }c))
    Return }\rho\leftarrow\mp@subsup{\sigma}{}{\mathrm{ DS }
OS.U.Uer(vk
    (root, tree) }\leftarrow\mathrm{ MerkleTree }\mp@subsup{}{}{H}(M)\mathrm{ , path }\leftarrow\mathrm{ MerklePath }\mp@subsup{}{}{H}(\mathrm{ tree, j)
    If DS.Verify (vk }\mp@subsup{}{}{\textrm{DS}},(\mathrm{ root, }c),\mp@subsup{\sigma}{}{\textrm{DS}})=0\mathrm{ , return }\perp\mathrm{ .
    \sigma }\mp@subsup{}{}{\textrm{OS}}\leftarrow(\mathrm{ root, }c,\mp@subsup{\sigma}{}{\textrm{DS}}\mathrm{ , path, j,r), return ( }\mp@subsup{m}{j}{},\mp@subsup{\sigma}{}{\textrm{OS}})\mathrm{ .
OS.Verify ( }\mp@subsup{\textrm{kk}}{}{\textrm{OS}}=\mp@subsup{\textrm{vk}}{}{\textrm{DS}},m,\mp@subsup{\sigma}{}{\textrm{OS}}=(\mathrm{ root, }c,\mp@subsup{\sigma}{}{\textrm{DS}},\mathrm{ path, j,r):
    If root }\not=\mathrm{ RootReconstruct }\mp@subsup{}{}{H}\mathrm{ (path, m,j), return 0.
    If c\not= COM.Commit(ck, m;r), return 0.
```



```
    Otherwise return 1.
```

Fig．6．Our generic construction $(1, n)-$ OS $_{\text {ours }}[\mathcal{H}, \mathrm{COM}, \mathrm{DS}]$ ．

## 4．3 Analysis

We analyze our scheme $(1, n)$－OS Ours． ．It is easy to see that our scheme satisfies the correctness．Now，we prove that our generic construction $(1, n)$－OS Ours satisfies the ambiguity and the Seq－sEUF－CMA security．

Theorem 1．If COM is computational hiding commitment，$(1, n)$－ $\mathrm{OS}_{\text {Ours }}[\mathcal{H}, \mathrm{COM}$ ， DS］satisfies the ambiguity．

Proof．The ambiguity of our scheme can be proven in a similar way in［21］． Let A be an adversary for the ambiguity game of $(1, n)-\mathrm{OS}_{\text {Ours }}$ ．We give a re－ duction algorithm $B$ that reduces the ambiguity security of our scheme to the computational hiding property of COM in Fig． 7.

Now，we confirm that B simulates the ambiguity game of $(1, n)$－ $\mathrm{OS}_{\text {Ours }}$ ．In the case that $b=0, c^{*} \leftarrow$ COM．Commit $\left(\mathrm{ck}, m_{0}^{*}=m_{i_{0}}\right)$ holds．B simulates $\mu$ on the choice of $m_{i_{0}}$ in this case．Similary，in the case that $b=1, c^{*} \leftarrow$ COM．Commit（ck，$m_{1}^{*}=m_{i_{1}}$ ）holds．B simulates $\mu$ on the choice of $m_{i_{1}}$ in this case．Since $b$ is chosen uniformly at random from $\{0,1\}$ ，B perfectly simu－ lates the ambiguity game of $(1, n)$－OS ${ }_{\text {Ours }}$ ．We can see that $\operatorname{Adv}_{\mathrm{COM}, \mathrm{B}}^{\text {Hide }}\left(1^{\lambda}\right)=$ $\operatorname{Adv}_{(1, n)-\mathrm{OS}}^{\mathrm{Amur},}, \mathrm{A}\left(1^{\lambda}\right)$ holds．Thus，we can conclude Theorem 1.
Theorem 2．If $\mathcal{H}$ is a family of collision－resistant hash functions，DS satis－ fies the sEUF－CMA security，and COM is computational binding commitment， $(1, n)$－OS ${ }_{\text {Ours }}[\mathcal{H}, \mathrm{COM}, \mathrm{DS}]$ satisfies the Seq－sEUF－CMA security．

```
\(\mathrm{B}\left(1^{\lambda}, \mathrm{ck}\right)\) :
    \(H \stackrel{\oiint}{\leftarrow} H_{\lambda}, \mathrm{pp}^{\mathrm{DS}} \leftarrow \mathrm{DS} . \operatorname{Setup}\left(1^{\lambda}\right), \mathrm{pp}^{\mathrm{OS}} \leftarrow\left(H, \mathrm{ck}, \mathrm{pp}^{\mathrm{DS}}\right)\),
    \(\left(\mathrm{vk}^{\mathrm{DS}}, \mathrm{sk}^{\mathrm{DS}}\right) \leftarrow \mathrm{DS}\).KeyGen \(\left(\mathrm{pp}^{\mathrm{DS}}\right),\left(\mathrm{vk}^{\mathrm{OS}}, \mathrm{sk}^{\mathrm{OS}}\right) \leftarrow\left(\mathrm{vk}^{\mathrm{DS}}, \mathrm{sk}^{\mathrm{DS}}\right)\),
    \(\left(M=\left(m_{0}, \ldots, m_{n-1}\right), i_{0}, i_{1}, \mathrm{st}_{\mathrm{A}}\right) \leftarrow \mathrm{A}\left(\mathrm{pp}^{\mathrm{OS}}, \mathrm{vk}^{\mathrm{OS}}, \mathrm{sk}^{\mathrm{OS}}\right)\)
    \(m_{0}^{*} \leftarrow m_{i_{0}}, m_{1}^{*} \leftarrow m_{i_{1}}\), send \(\left(m_{0}^{*}, m_{1}^{*}\right)\) to the challenger C and obtain
        a commitment \(c^{*}\) where \(c^{*} \leftarrow \operatorname{COM}\).Commit (ck, \(m_{b}^{*}\) ) and \(b \stackrel{\&}{\leftarrow}\{0,1\}\) is chosen C.
    \(b^{\prime} \leftarrow \mathrm{A}\left(\mu=c^{*}\right.\), st \(\left.\mathrm{st}_{\mathrm{A}}\right)\), return \(b^{*} \leftarrow b^{\prime}\).
```

Fig. 7. The reduction algorithm B.

Proof. Let A be a PPT adversary for the Seq-sEUF-CMA game of $(1, n)$-OS ${ }_{\text {Ours }}$. We introduce the base game $\operatorname{Game}_{(1, n)-\mathrm{OS}}^{\mathrm{Our}, \mathrm{A}} \mathrm{Base}$ which simulates $\operatorname{Game}_{(1, n) \text {-OS }}^{\text {Baur }, \mathrm{A}}$. We provide Game $_{(1, n)-\mathrm{OS}}^{\mathrm{Bars}, \mathrm{A}} \mathrm{A}$ in Fig. 8.
 (e.g., Final, $\mathrm{DS}_{\text {reuse }}$ ) which are used for classifying forgery type and a table $\mathbb{T}$ which stores the computation of the signing oracle $\mathcal{O}^{\text {Sign }}$. More precisely, the flag Final represents that a forgery $\left(m^{*}, \sigma^{* O S}=\left(\right.\right.$ root $^{*}, c^{*}, \sigma^{* D S}$, path $\left.\left.{ }^{*}, j^{*}, r^{*}\right)\right)$ is submitted in the final output (Final $=$ true $)$ or $\mathcal{O}^{\text {Fin }}$ (Final $=$ false $)$. The flag $\mathrm{DS}_{\text {reuse }}$ represents that there is a pair $\left(\widetilde{m}, \widetilde{\sigma}^{\mathrm{OS}}\right) \neq\left(\widetilde{m}^{\prime}, \widetilde{\sigma}^{\prime O S}\right)$ in $\mathbb{L}^{\text {Sign }}$ such that the first three elements of $\sigma^{\mathrm{OS}}$ are the same. i.e., $\left(\widetilde{\text { root }}, \widetilde{c}, \widetilde{\sigma}^{\mathrm{DS}}\right)=\left(\widetilde{\text { root }}^{\prime}, \widetilde{c}^{\prime}, \widetilde{\sigma}^{\prime \mathrm{DS}}\right)$ holds. We represent that such a pair exists as $\mathrm{DS}_{\text {reuse }}=$ true. The table $\mathbb{T}$ stores a tuple ( $i, M$, root, $c, \sigma^{\mathrm{DS}}$ ) where ( $M, c$ ) is an input for an $i$-th $\mathcal{O}^{\text {Sign }}$ query, (root, tree) $\leftarrow$ MerkleTree ${ }^{H}(M), \sigma^{\mathrm{DS}} \leftarrow \mathrm{DS}$.Sign $\left(\mathrm{sk}^{\mathrm{DS}},(\right.$ root,$\left.c)\right)$. The counter $q^{\text {Sign }}$ represents the number of outputs that A received from the $\mathcal{O}^{\text {Sign }}$ oracle and $q^{\text {Fin }}$ represent the number of submitted signatures from $A$.

Now, we divide an adversary $A$ into three types $A_{1}, A_{2}, A_{3}$ according to states of flags $\mathrm{DS}_{\text {reuse }}, \mathrm{DS}_{\text {forge }}$, and $\mathrm{COM}_{\text {coll }}$ when A wins the game Game ${ }^{\text {Base }}$.
$-A_{1}$ wins the game with $\mathrm{DS}_{\text {forge }}=$ true.
$-A_{2}$ wins the game with $\mathrm{COM}_{\text {coll }}=$ true.
$-\mathrm{A}_{3}$ wins the game with $\mathrm{DS}_{\text {forge }}=\mathrm{false} \wedge \mathrm{COM}_{\text {coll }}=\mathrm{false}$.
For adversaries $\mathrm{A}_{1}, \mathrm{~A}_{2}$, and $\mathrm{A}_{3}$, we can construct a reduction for the security of DS, COM, and $H$ respectively. Now, we give reductions for these adversaries.

Reduction $\mathrm{B}^{\mathrm{DS}}$ : A reduction $\mathrm{B}^{\mathrm{DS}}$ to the sEUFCMA security game of DS is obtained by modifying $\operatorname{Game}_{(1, n)-\mathrm{OS}_{\text {ours }, \mathrm{A}}^{\text {Aase }}}$ as follows. Instead of running $\mathrm{pp}^{\mathrm{DS}} \leftarrow$ DS.Setup $\left(1^{\lambda}\right)$ and $\left(\mathrm{vk}^{\mathrm{DS}}, \mathrm{sk}^{\mathrm{DS}}\right) \leftarrow \mathrm{DS}$. KeyGen $\left(\mathrm{pp}^{\mathrm{DS}}\right)$, $\mathrm{B}^{\mathrm{DS}}$ uses $\left(\mathrm{pp}^{\mathrm{DS}}, \mathrm{vk}^{\mathrm{DS}}\right)$ given by the sEUFCMA security game of DS. For a signing query $(M, c)$ from $\mathrm{A}, \mathrm{B}^{\mathrm{DS}}$ query (root, $c$ ) to the signing oracle of the sEUFCMA security game of DS, obtains $\sigma^{\mathrm{DS}} \leftarrow \mathrm{DS} . \operatorname{Sign}\left(\mathrm{sk}^{\mathrm{DS}},(\right.$ root,$\left.c)\right)$, and returns $\sigma^{\mathrm{DS}}$. To simplify the discussion, we assume that A makes distinct $(M, c)$ to $\mathrm{B}^{\mathrm{DS}}$. (If A makes the same $(M, c)$ more than once, $\mathrm{B}^{\mathrm{DS}}$ simply outputs return $\sigma^{\mathrm{DS}} \leftarrow \mathrm{DS} . \operatorname{Sign}\left(\mathrm{sk}^{\mathrm{DS}},(\mathrm{root}, c)\right)$ which was previously obtained by the signing oracle of the sEUFCMA security game where root is computed from $M$.)

```
\(\operatorname{Game}_{\mathrm{OS}}^{\text {Saur }, \mathrm{A}}\) A \(\left(1^{\lambda}\right)\) :
    \(\mathbb{L}^{\text {Sign }} \leftarrow\{ \}, \mathbb{L}^{\text {ListM }} \leftarrow\{ \}, \mathbb{T} \leftarrow\{ \}, q^{\text {Sign }} \leftarrow 0, q^{\text {Fin }} \leftarrow 0\), Final \(\leftarrow\) false,
    \(\mathrm{DS}_{\text {reuse }} \leftarrow\) false, \(\mathrm{COM}_{\text {coll }} \leftarrow\) false, \(\mathrm{DS}_{\text {forge }} \leftarrow\) false, ck \(\leftarrow\) COM.KeyGen \(\left(1^{\lambda}\right)\),
    \(\mathrm{pp}^{\mathrm{DS}} \leftarrow \mathrm{DS} . \operatorname{Setup}\left(1^{\lambda}\right), \mathrm{pp}^{\mathrm{OS}} \leftarrow\left(H, \mathrm{ck}, \mathrm{pp}^{\mathrm{DS}}\right),\left(\mathrm{vk}^{\mathrm{DS}}, \mathrm{sk}^{\mathrm{DS}}\right) \leftarrow\) DS.KeyGen \(\left(\mathrm{pp}^{\mathrm{DS}}\right)\),
    \(\left(\mathrm{vk}^{\mathrm{OS}}, \mathrm{sk}^{\mathrm{OS}}\right) \leftarrow\left(\mathrm{vk}^{\mathrm{DS}}, \mathrm{sk}^{\mathrm{DS}}\right),\left(m^{*}, \sigma^{* \mathrm{OS}}\right) \leftarrow \mathrm{A}^{\mathcal{O}^{\text {Sign }}(\cdot, \cdot), \mathcal{O}^{\mathrm{Fin}}(\cdot, \cdot)}\left(\mathrm{pp}^{\mathrm{OS}}, \mathrm{vk}^{\mathrm{OS}}\right)\)
    If \(q^{\mathrm{Sign}} \neq q^{\mathrm{Fin}} \vee \mathrm{OS}\). Verify \(\left(\mathrm{vk}^{\mathrm{OS}}, m^{*}, \sigma^{*}\right) \neq 1 \vee\left(m^{*}, \sigma^{* \mathrm{OS}}\right) \in \mathbb{L}^{\text {Sign }}\), return 0 .
    Final \(\leftarrow\) true \(, \mathbb{L}^{\text {Sign }} \leftarrow \mathbb{L}^{\text {Sign }} \cup\left\{\left(m^{*}, \sigma^{* O S}\right)\right\}, q^{\text {Fin }} \leftarrow q^{\text {Fin }}+1\)
    Search a pair \(\left(\widetilde{m}, \widetilde{\sigma}^{\mathrm{OS}}\right) \neq\left(\widetilde{m}^{\prime}, \widetilde{\sigma}^{\prime O S}\right)\) in \(\mathbb{L}^{\text {Sign }}\) such that the first three
        elements of \(\sigma^{\mathrm{OS}}\) are the same. i.e., \(\left(\widetilde{\mathrm{root}}, \widetilde{c}, \widetilde{\sigma}^{\mathrm{DS}}\right)=\left(\widetilde{\mathrm{root}}^{\prime}, \widetilde{c}^{\prime}, \widetilde{\sigma}^{\prime \mathrm{DS}}\right)\)
    If there is no such a pair, \(\mathrm{DS}_{\text {forge }} \leftarrow\) true, return 1.
```

            \(\left(\right.\) Final \(=\) true \(\wedge D_{\text {reuse }}=\) false \(\wedge D_{\text {forge }}=\) true \(\wedge\) COM \(_{\text {coll }}=\) false \()\)
    \(\mathrm{DS}_{\text {reuse }} \leftarrow\) true.
    Parse \(\widetilde{\sigma}^{\mathrm{OS}}\) as \(\left(\right.\) root \(\left.^{*}, c^{*}, \sigma^{* \mathrm{DS}}, \widetilde{\text { path }}, \tilde{j}, \widetilde{r}\right), \widetilde{\sigma}^{\text {OS }}\) as (root \(\left.{ }^{*}, c^{*}, \sigma^{* \mathrm{DS}}, \widetilde{\text { path }}^{\prime}, \widetilde{j}^{\prime}, \widetilde{r}^{\prime}\right)\).
    If \((\widetilde{m}, \widetilde{r}) \neq\left(\widetilde{m}^{\prime}, \widetilde{r}^{\prime}\right), \mathrm{COM}_{\text {coll }} \leftarrow\) true, return 1 .
            \(\left(\right.\) Final \(=\) true \(\wedge \mathrm{DS}_{\text {reuse }}=\) true \(\wedge \mathrm{DS}_{\text {forge }}=\) false \(\wedge \mathrm{COM}_{\text {coll }}=\) true \()\)
    Otherwise, return 1.
            \(\left(\right.\) Final \(=\) true \(\wedge D S_{\text {reuse }}=\) true \(\wedge D_{\text {forge }}=\) false \(\wedge C O M_{\text {coll }}=\) false \()\)
    Oracle $\mathcal{O}^{\text {Sign }}\left(M=\left(m_{0}, \ldots, m_{n-1}\right), \mu=c\right)$ :
If $q^{\text {Sign }} \neq q^{\text {Fin }}$, return $\perp$.
If there exists a pair $\left(t \neq t^{\prime} \in\{0, \ldots, n-1\}\right)$ such that $m_{t}=m_{t^{\prime}}$, return $\perp$.
(root, tree) $\leftarrow$ MerkleTree ${ }^{H}(M), \sigma^{\mathrm{DS}} \leftarrow \mathrm{DS} . \operatorname{Sign}\left(\right.$ sk $^{\mathrm{DS}},($ root, $\left.c)\right)$,
$q^{\text {Sign }} \leftarrow q^{\text {Sign }}+1, M_{q^{\text {Sign }}} \leftarrow M, \mathbb{L}^{\text {ListM }} \leftarrow \mathbb{L}^{\text {ListM }} \cup\left\{\left(q^{\text {Sign }}, M_{q^{\text {Sign }}}\right)\right\}$,
$\mathbb{T} \leftarrow \mathbb{T} \cup\left\{\left(q^{\text {Sign }}, M_{q^{\text {sign }}}\right.\right.$, root, $\left.\left.c, \sigma^{\mathrm{DS}}\right)\right\}$,
return $\rho \leftarrow \sigma^{\mathrm{DS}}$ to A.
Oracle $\mathcal{O}^{\text {Fin }}\left(m^{*}, \sigma^{* O S}\right)$ :
If $q^{\text {Sign }} \neq q^{\text {Fin }}+1$, return $\perp$.
If OS.Verify $\left(\mathrm{vk}^{\mathrm{OS}}, m^{*}, \sigma^{* O S}\right) \neq 1$, return the game output 0 and abort.
If $\left(m^{*}, \sigma^{* O S}\right) \in \mathbb{L}^{\text {Sign }}$, return the game output 0 and abort.
$\mathbb{L}^{\text {Sign }} \leftarrow \mathbb{L}^{\text {Sign }} \cup\left\{\left(m^{*}, \sigma^{* O S}\right)\right\}, q^{\text {Fin }} \leftarrow q^{\text {Fin }}+1$, retrieve $\left(q^{\text {Sign }}, M_{q^{S i g n}}\right) \in \mathbb{L}^{\text {ListM }}$.
If $m^{*} \in M_{q}$ Sign , return "accept" to A.
Parse $\sigma^{* O S}$ as (root ${ }^{*}, c^{*}, \sigma^{* D S}$, path ${ }^{*}, j^{*}, r^{*}$ ).
If $\left(q^{\text {Sign }}, *\right.$, root $\left.^{*}, c^{*}, \sigma^{* D S}\right) \in \mathbb{T}$ return the game output 1.
(Final $=$ false $\wedge D S_{\text {reuse }}=$ false $\wedge$ DS $_{\text {forge }}=$ false $\wedge C O M_{\text {coll }}=$ false $)$
Search a pair $\left(\widetilde{m}, \widetilde{\sigma}^{\mathrm{OS}}\right) \neq\left(\widetilde{m}^{\prime}, \widetilde{\sigma}^{\prime O S}\right)$ in $\mathbb{L}^{\text {Sign }}$ such that the first three
elements of $\sigma^{\mathrm{OS}}$ are the same. i.e., $\left(\widetilde{\mathrm{root}}, \widetilde{c}, \widetilde{\sigma}^{\mathrm{DS}}\right)=\left(\widetilde{\mathrm{root}^{\prime}}, \widetilde{c}^{\prime}, \widetilde{\sigma}^{\prime \mathrm{DS}}\right)$
If there is no such a pair, $\mathrm{DS}_{\text {forge }} \leftarrow$ true, return the game output 1 .
$\left(\right.$ Final $=$ false $\wedge D S_{\text {reuse }}=$ false $\wedge D S_{\text {forge }}=$ true $\wedge C O M_{\text {coll }}=$ false $)$
$\mathrm{DS}_{\text {reuse }} \leftarrow$ true.
Parse $\widetilde{\sigma}^{\mathrm{OS}}$ as $\left(\right.$ root $\left.^{*}, c^{*}, \sigma^{* \mathrm{DS}}, \widetilde{\text { path }}, \tilde{j}, \widetilde{r}\right), \widetilde{\sigma}^{\prime \mathrm{OS}}$ as (root $\left.{ }^{*}, c^{*}, \sigma^{* \mathrm{DS}}, \widetilde{\text { path }}^{\prime}, \widetilde{j}^{\prime}, \widetilde{r}^{\prime}\right)$.
If $(\widetilde{m}, \widetilde{r}) \neq\left(\widetilde{m}^{\prime}, \widetilde{r}^{\prime}\right), \mathrm{COM}_{\text {coll }} \leftarrow$ true, return the game output 1 .
(Final $=$ false $\wedge$ DS $_{\text {reuse }}=$ true $\wedge \mathrm{DS}_{\text {forge }}=$ false $\wedge \mathrm{COM}_{\text {coll }}=$ true $)$
Otherwise, return the game output 1 .
(Final $=\mathrm{false} \wedge \mathrm{DS}_{\text {reuse }}=$ true $\wedge \mathrm{DS}_{\text {forge }}=$ false $\left.\wedge \mathrm{COM}_{\text {coll }}=\mathrm{false}\right)$

Fig. 8. The base game $\operatorname{Game}_{(1, n)-\mathrm{OS}}^{\mathrm{Ours}, \mathrm{A}} \mathrm{A} \cdot$

If $\mathrm{B}^{\mathrm{DS}}$ outputs 1 with the condition where $\mathrm{DS}_{\text {forge }}=$ true, there is the forgery $\left(\widetilde{\text { root }}, \widetilde{c}, \widetilde{\sigma}^{\mathrm{DS}}\right)$. Since $\mathrm{DS}_{\text {forge }}=$ true holds, $\mathrm{DS}_{\text {reuse }}=\mathrm{false}$ holds. This fact implies that for $\left(m, \sigma^{\mathrm{OS}}\right) \in \mathbb{L}^{\text {Sign }}$, the first three elements (root, $c, \sigma^{\mathrm{DS}}$ ) of $\sigma^{\text {OS }}$ are all distinct in $\mathbb{L}^{\text {Sign }}$ and valid signatures for DS (i.e., DS.Verify $\left(\mathrm{vk}^{\mathrm{DS}}\right.$, (root, $c$ ),$\left.\sigma^{\mathrm{DS}}\right)=1$ ). Moreover, $\mathrm{B}^{\mathrm{DS}}$ makes $q^{\text {Sign }}$ signing queries to signing oracle, $q^{\text {Sign }}<q^{\text {Fin }}$ holds where $q^{\text {Fin }}$ is the number of entry in $\mathbb{L}^{\text {Sign }}$. Hence, there is a forgery $\left((\widetilde{\text { root }}, \widetilde{c}), \widetilde{\sigma}^{\mathrm{DS}}\right)$ of DS. By modifying Game $_{(1, n)-\mathrm{OS}}^{\text {Ours }, \mathrm{A}}$ Base to output this forgery $\left((\widetilde{\text { root }}, \widetilde{c}), \widetilde{\sigma}^{\mathrm{DS}}\right)$, we can obtain $\mathrm{B}^{\mathrm{DS}}$.

Reduction $\mathrm{B}^{\mathrm{COM}}$ : A reduction $\mathrm{B}^{\mathrm{COM}}$ to the computational binding property of COM is obtained by modifying $\operatorname{Game}_{(1, n)-\mathrm{OS}_{\text {ours }}, \mathrm{A}}^{\mathrm{Base}}$ as follows. $\mathrm{B}^{\mathrm{COM}}$ uses ck given by the strong computational binding security game of COM.

If Gameos ${ }_{\text {Ours }}, \mathrm{A}$ outputs 1 with the condition where $\mathrm{COM}_{\text {coll }}=$ true, there is a collision $(\widetilde{m}, \widetilde{r}) \neq\left(\widetilde{m}^{\prime}, \widetilde{r}^{\prime}\right)$ such that COM.Commit(ck, $\left.\widetilde{m} ; \widetilde{r}\right)=$ COM.Commit(ck, $\left.\widetilde{m}^{\prime} ; \widetilde{r}^{\prime}\right)$ holds. Since if $\mathrm{COM}_{\text {coll }}=$ true holds, $\mathrm{DS}_{\text {reuse }}=$ true holds in Game ${ }_{\mathrm{OS}}^{\text {Ours }}$, A . This fact implies that there is a pair $\left(\widetilde{m}, \widetilde{\sigma}^{\text {OS }}=\left(\operatorname{root}^{*}, c^{*}, \sigma^{* D S}, \widetilde{\text { path }}, \widetilde{j}, \widetilde{r}\right)\right) \neq$ $\left(\widetilde{m}^{\prime}, \widetilde{\sigma}^{\prime \text { OS }}=\left(\right.\right.$ root $\left.\left.{ }^{*}, c^{*}, \sigma^{* \mathrm{DS}}, \widetilde{\text { path }}{ }^{\prime}, \widetilde{j}^{\prime}, \widetilde{r}^{\prime}\right)\right)$. Since $\left(\widetilde{m}, \widetilde{\sigma}^{\mathrm{OS}}\right)$ and $\left(\widetilde{m}^{\prime}, \widetilde{\sigma}^{\text {OS }}\right)$ are valid signatures, $(\widetilde{m}, \widetilde{r}) \neq\left(\widetilde{m}^{\prime}, \widetilde{r}^{\prime}\right)$ and COM.Commit(ck, $\left.\widetilde{m} ; \widetilde{r}\right)=$ COM.Commit(ck, $\left.\widetilde{m}^{\prime} ; \widetilde{r}^{\prime}\right)$ hold. By modifying $\operatorname{Game}_{(1, n)-\mathrm{OS}}^{\text {Ours }, \mathrm{A}} \mathrm{Base}$ to output this collision $\left(\left(\widetilde{m}, \widetilde{\sigma}^{\mathrm{OS}}\right),\left(\widetilde{m}^{\prime}\right.\right.$, $\left.\widetilde{\sigma}^{\prime \mathrm{OS}}\right)$ ), we can obtain $\mathrm{B}^{\mathrm{COM}}$.

Reduction $\mathrm{B}^{\text {Hash }}$ : We explain how to obtain a reduction $\mathrm{B}^{\text {Hash }}$ to the collision resistance property from $\operatorname{Game}_{(1, n)-\mathrm{OS}_{\text {ours },}, \mathrm{A}}^{\mathrm{Base}}$. If $\operatorname{Game}_{(1, n)-\mathrm{OS}_{\text {our },}, \mathrm{A}}^{\text {Base }}$ outputs 1 with the condition where Final $=$ false $\wedge D S_{\text {reuse }}=$ false $\wedge D S_{\text {forge }}=$ false, a collision a hash function can be found. Since Final $=$ false $\wedge \mathrm{DS}_{\text {reuse }}=$ false $\wedge \mathrm{DS}_{\text {forge }}=$ false holds, then $\left(q^{\text {Sign }}, *\right.$, root $\left.{ }^{*}, c^{*}, \sigma^{* \mathrm{DS}}\right) \in \mathbb{T}$ holds. Let $\left(M_{q^{\text {Sign }}}, c_{q^{\mathrm{Sign}}}\right)$ be an input for the $q^{\text {Sign }}$-th $\mathcal{O}^{\text {Sign }}$ query. Then, by the computation of $\mathcal{O}^{\text {Sign }}$ and table $\mathbb{T}, c^{*}=c_{q^{\text {sign }}},\left(\right.$ root $^{*}$, tree $\left.^{*}\right)=$ MerkleTree ${ }^{H}\left(M_{q^{\text {sign }}}\right)$, and DS.Verify $\left(\mathrm{vk}^{\mathrm{DS}},\left(\right.\right.$ root $\left.\left.^{*}, c^{*}\right), \sigma^{* \mathrm{DS}}\right)=1$ holds. Since $m^{*} \notin M_{q^{\text {sign }}}$, a collision of a hash function $H$ can be computed by $\left(x, x^{\prime}\right) \leftarrow \operatorname{Ext}_{1}\left(\operatorname{tree}^{*},\left(m^{*}\right.\right.$, path $\left.\left.^{*}, i^{*}\right)\right)$. We modify $\operatorname{Game}_{(1, n) \text {-OS Ours }, \mathrm{A}}^{\text {Base }}$ to output this collision $\left(x, x^{\prime}\right)$ in this case.

If $\mathrm{B}_{(1, n)-\mathrm{OS}}^{\text {Ours }, \mathrm{A}} \mathrm{A}$ Outputs 1 with the condition where $\mathrm{DS}_{\text {reuse }}=\operatorname{true} \wedge \mathrm{DS}_{\text {forge }}=$ false $\wedge \operatorname{COM}_{\text {coll }}=$ false (regardless of the bool value Final), a collision of a hash function can be also found. Since $\mathrm{DS}_{\text {reuse }}=$ false $\wedge \mathcal{C O M}_{\text {coll }}=$ false holds, then there is a pair $\left(\widetilde{m}, \widetilde{\sigma}^{\mathrm{OS}}=\left(\operatorname{root}^{*}, c^{*}, \sigma^{* \mathrm{DS}}, \widetilde{\text { path }}, \widetilde{j}, \widetilde{r}\right)\right) \neq\left(\widetilde{m}, \widetilde{\sigma}^{\prime \mathrm{OS}}=\right.$ $\left(\right.$ root $\left.^{*}, c^{*}, \sigma^{* \mathrm{DS}}, \widetilde{\text { path }}{ }^{\prime}, \widetilde{j}^{\prime}, \widetilde{r}\right)$ ) holds. From this fact, we can see that $(\widetilde{\text { path }}, \widetilde{j}) \neq$ ( $\widetilde{\text { path }}^{\prime}, \tilde{j}^{\prime}$ ) holds. If $j^{*} \neq \widetilde{j}$ holds, we can obtain a collision of a hash function H as $\left(x, x^{\prime}\right) \leftarrow \operatorname{Ext}_{1}\left(\operatorname{tree}^{*},\left(m^{*}\right.\right.$, path $\left.^{*}, i^{*}\right)$. If $j^{*}=\widetilde{j}$ holds, then path $=\widetilde{\text { path }}{ }^{\prime}$ holds and thus we can compute a collision of a hash function as $\left(x, x^{\prime}\right) \leftarrow$ $\operatorname{Ext}_{2}\left(m, j^{*}, \widetilde{\text { path }}, \widetilde{\text { path }}{ }^{\prime}\right)$. We modify $\operatorname{Game}_{(1, n)-\mathrm{OS}_{\text {Ours }}, \mathrm{A}}^{\text {Base }}$ to output this collision $\left(x, x^{\prime}\right)$ in these case.

By reduction algorithms $B^{D S}, B^{C O M}$, and $B^{\text {Hash }}$ described above, we can bound the advantage $\operatorname{Adv}_{(1, n)-\mathrm{OS}, \mathrm{A}}^{\text {Seq-SEUFCMA }}\left(1^{\lambda}\right)$ as

$$
\begin{aligned}
& \operatorname{Adv}_{(1, n)-\mathrm{OS}, \mathrm{~A}}^{\text {Seq-SEUFCMA }}\left(1^{\lambda}\right) \\
& =\operatorname{Pr}\left[\operatorname{Game}_{(1, n)-\text { OS }_{\text {Ours }}, \mathrm{A}}^{\text {Seq-EUFCMA }}\left(1^{\lambda}\right) \Rightarrow 1\right] \\
& =\operatorname{Pr}\left[\operatorname{Game}_{(1, n)-\mathrm{OS}_{\text {ours }}, \mathrm{A}}^{\mathrm{Base}}\left(1^{\lambda}\right) \Rightarrow 1\right] \\
& =\operatorname{Pr}\left[\operatorname{Game}_{(1, n)-\mathrm{OS}}^{\text {Our }, \mathrm{A}} \mathrm{~A}\left(1^{\lambda}\right) \Rightarrow 1 \wedge \mathrm{DS}_{\text {forge }}=\text { true }\right] \\
& +\operatorname{Pr}\left[\operatorname{Game}_{(1, n)-\mathrm{OS}_{\text {Ours }, ~}}^{\mathrm{A}} \mathrm{~A}^{\mathrm{Base}}\left(1^{\lambda}\right) \Rightarrow 1 \wedge \mathrm{COM}_{\text {coll }}=\text { true }\right] \\
& +\operatorname{Pr}\left[\operatorname{Game}_{(1, n)-\mathrm{OS}}^{\text {Ours }, \mathrm{A}} \mathrm{~A}^{\text {Base }}\left(1^{\lambda}\right) \Rightarrow 1 \wedge \mathrm{DS}_{\text {forge }}=\text { false } \wedge \operatorname{COM}_{\text {coll }}=\mathrm{false}\right] \\
& \leq \operatorname{Adv}_{\mathrm{DS}, \mathrm{~A}_{1}}^{\mathrm{sEUCR}}\left(1^{\lambda}\right)+\operatorname{Adv}_{\mathrm{COM}, \mathrm{~A}_{2}}^{\mathrm{sBind}}\left(1^{\lambda}\right)+\operatorname{Adv}_{\mathcal{H}, \mathrm{A}_{3}}^{\mathrm{Coll}}\left(1^{\lambda}\right) \text {. }
\end{aligned}
$$

By this fact, we can conclude Theorem 2.

## 5 Conclusion

Summary of Our Results. In this paper, we revisit the unforgeability security for a 1-out-of- $n$ oblivious signature scheme and point out problems. By reflecting on these problems, we define the Seq-sEUF-CMA security. We propose the improved generic construction of a 1-out-of- $n$ oblivious signature scheme $(1, n)-\mathrm{OS}_{\text {Ours }}$. Compared to the construction by Zhou et al. [21], our construction offers a smaller second message size. The sum of a second message size and a signature size is improved from $O(n)$ to $O(\log n)$.

Discussion of Our Security Model and Open Problem. We introduce the Seq-sEUF-CMA security in Definition 10. This security model restricts an adversary A to execute signing interactions only in a sequential manner. It is natural to consider a model that allows concurrent signing interactions. However, if we straightforwardly extend our security model to a concurrent setting, there is a trivial attack.

Let us consider the following adversary A that runs signing protocol executions twice concurrently. A chooses two list $M_{1}=\left(m_{1,0}, \ldots, m_{1, n-1}\right)$ and $M_{2}=\left(m_{2,0}, \ldots, m_{2, n-1}\right)$ such that $M_{1} \cap M_{2}=\emptyset$ (i.e., there is no element $m$ such that $m \in M_{1} \wedge m \in M_{2}$ ). In the 1st interaction, A chooses $m_{1,0} \in M_{1}$, obtains a signature $\sigma_{1}$ on a message $m_{1,0}$. In the 2 nd interaction A chooses $m_{2,0} \in M_{2}$, obtains a signature $\sigma_{2}$ on a message $m_{2,0}$. A finishs the 1st interaction by outputting ( $m_{2,0}, \sigma_{2}$ ). Since $m_{2,0} \notin M_{1}$, A trivially wins the unforgeability game.

Due to this trivial attack, we cannot straightforwardly extend our security model to the concurrent signing interaction setting. We leave an open problem to define the unforgeability security model for a 1-out-of- $n$ oblivious signature scheme that supports concurrent signing interactions.

## References

1. M. Abe and E. Fujisaki. How to date blind signatures. In K. Kim and T. Matsumoto, editors, Advances in Cryptology - ASIACRYPT '96, International Con-
ference on the Theory and Applications of Cryptology and Information Security, Kyongju, Korea, November 3-7, 1996, Proceedings, volume 1163 of Lecture Notes in Computer Science, pages 244-251. Springer, 1996.
2. S. Agrawal, E. Kirshanova, D. Stehlé, and A. Yadav. Practical, round-optimal lattice-based blind signatures. In H. Yin, A. Stavrou, C. Cremers, and E. Shi, editors, Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022, pages 39-53. ACM, 2022.
3. M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28-June 1, 2006, Proceedings, volume 4004 of Lecture Notes in Computer Science, pages 409-426. Springer, 2006.
4. P. Blaskiewicz, L. Hanzlik, K. Kluczniak, L. Krzywiecki, M. Kutylowski, M. Slowik, and M. Wszola. Pseudonymous signature schemes. In K. Li, X. Chen, and W. Susilo, editors, Advances in Cyber Security: Principles, Techniques, and Applications, pages 185-255. Springer, 2019.
5. D. Chaum. Blind signatures for untraceable payments. In D. Chaum, R. L. Rivest, and A. T. Sherman, editors, Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982, pages 199-203. Plenum Press, New York, 1982.
6. L. Chen. Oblivious signatures. In D. Gollmann, editor, Computer Security ESORICS 94, Third European Symposium on Research in Computer Security, Brighton, UK, November 7-9, 1994, Proceedings, volume 875 of Lecture Notes in Computer Science, pages 161-172. Springer, 1994.
7. S. Chiou and J. Chen. Design and implementation of a multiple-choice e-voting scheme on mobile system using novel t -out-of- n oblivious signature. J. Inf. Sci. Eng., 34(1):135-154, 2018.
8. R. del Pino and S. Katsumata. A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling. In Y. Dodis and T. Shrimpton, editors, Advances in Cryptology - CRYPTO 2022-42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part II, volume 13508 of Lecture Notes in Computer Science, pages 306-336. Springer, 2022.
9. G. Fuchsbauer, C. Hanser, C. Kamath, and D. Slamanig. Practical round-optimal blind signatures in the standard model from weaker assumptions. In V. Zikas and R. D. Prisco, editors, Security and Cryptography for Networks - 10th International Conference, SCN 2016, Amalf, Italy, August 31 - September 2, 2016, Proceedings, volume 9841 of Lecture Notes in Computer Science, pages 391-408. Springer, 2016.
10. G. Fuchsbauer, C. Hanser, and D. Slamanig. Practical round-optimal blind signatures in the standard model. In R. Gennaro and M. Robshaw, editors, Advances in Cryptology - CRYPTO 2015-35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II, volume 9216 of Lecture Notes in Computer Science, pages 233-253. Springer, 2015.
11. E. Ghadafi. Efficient round-optimal blind signatures in the standard model. In A. Kiayias, editor, Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers, volume 10322 of Lecture Notes in Computer Science, pages 455-473. Springer, 2017.
12. L. Hanzlik and K. Kluczniak. A short paper on blind signatures from knowledge assumptions. In J. Grossklags and B. Preneel, editors, Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers, volume 9603 of Lecture Notes in Computer Science, pages 535-543. Springer, 2016.
13. L. Hanzlik and K. Kluczniak. Two-move and setup-free blind signatures with perfect blindness. In J. Baek and R. Zhang, editors, Proceedings of the 4 th ACM International Workshop on ASIA Public-Key Cryptography, APKC@AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2, 2017, pages 1-11. ACM, 2017.
14. L. Hanzlik, J. Loss, and B. Wagner. Rai-choo! evolving blind signatures to the next level. In C. Hazay and M. Stam, editors, Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V, volume 14008 of Lecture Notes in Computer Science, pages 753-783. Springer, 2023.
15. S. Katsumata, R. Nishimaki, S. Yamada, and T. Yamakawa. Round-optimal blind signatures in the plain model from classical and quantum standard assumptions. In A. Canteaut and F. Standaert, editors, Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I, volume 12696 of Lecture Notes in Computer Science, pages 404-434. Springer, 2021.
16. V. Lyubashevsky, N. K. Nguyen, and M. Plançon. Efficient lattice-based blind signatures via gaussian one-time signatures. In G. Hanaoka, J. Shikata, and Y. Watanabe, editors, Public-Key Cryptography - PKC 2022-25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8-11, 2022, Proceedings, Part II, volume 13178 of Lecture Notes in Computer Science, pages 498-527. Springer, 2022.
17. R. C. Merkle. A digital signature based on a conventional encryption function. In C. Pomerance, editor, Advances in Cryptology-CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings, volume 293 of Lecture Notes in Computer Science, pages 369-378. Springer, 1987.
18. C. Song, X. Yin, and Y. Liu. A practical electronic voting protocol based upon oblivious signature scheme. In 2008 International Conference on Computational Intelligence and Security, CIS 2008, 13-17 December 2008, Suzhou, China, Volume 1 - Conference Papers, pages 381-384. IEEE Computer Society, 2008.
19. R. Tso, T. Okamoto, and E. Okamoto. 1-out-of-n oblivious signatures. In L. Chen, Y. Mu, and W. Susilo, editors, Information Security Practice and Experience, 4 th International Conference, ISPEC 2008, Sydney, Australia, April 21-23, 2008, Proceedings, volume 4991 of Lecture Notes in Computer Science, pages 45-55. Springer, 2008.
20. J. You, Z. Liu, R. Tso, Y. Tseng, and M. Mambo. Quantum-resistant 1-out-of-n oblivious signatures from lattices. In C. Cheng and M. Akiyama, editors, Advances in Information and Computer Security - 17th International Workshop on Security, IWSEC 2022, Tokyo, Japan, August 31 - September 2, 2022, Proceedings, volume 13504 of Lecture Notes in Computer Science, pages 166-186. Springer, 2022.
21. Y. Zhou, S. Liu, and S. Han. Generic construction of 1-out-of-n oblivious signatures. IEICE Trans. Inf. Syst., 105-D (11):1836-1844, 2022.

# Compact Identity-based Signature and Puncturable Signature from SQISign 

Surbhi Sahaw and Ratna Dutta ${ }^{1}$<br>${ }^{1}$ Indian Institute of Technology Kharagpur<br>surbhi_shaw@iitkgp.ac.in; ratna@maths.iitkgp.ac.in


#### Abstract

Puncturable signature (PS) offers a fine-grained revocation of signing ability by updating its signing key for a given message $m$ such that the resulting punctured signing key can produce signatures for all messages except for $m$. In light of the applications of PS in proof-of-stake blockchain protocols, disappearing signatures and asynchronous transaction data signing services, this paper addresses the need for designing practical and efficient PS schemes. Existing proposals pertaining to PS suffer from various limitations, including computational inefficiency, false-positive errors, vulnerability to quantum attacks and large key and signature sizes. To overcome these challenges, we aim to design a PS from isogenies. We first propose an Identity-Based Signature (IBS) by employing the Short Quaternion and Isogeny Signature (SQISign). We provide a rigorous security analysis of our IBS and prove it is secure against unforgeability under chosen identity and chosen message attacks. More interestingly, our IBS achieves the most compact key and signature size compared to existing isogeny-based IBS schemes. Leveraging our proposed IBS, we introduce the first Short Quaternion and Isogeny Puncturable Signature (SQIPS) which allows for selective revocation of signatures and is supported by a comprehensive security analysis against existential forgery under chosen message attacks with adaptive puncturing. Our PS scheme SQIPS provides resistance from quantum attacks, enjoys small signature size and is free from false-positive errors.


Keywords: Puncturable signature, Isogenies, Identity-based signature, Post-quantum cryptography.

## 1 Introduction

With the proliferation of digital technology and the widespread adoption of online transactions, ensuring the privacy and security of sensitive data has emerged as a paramount concern. Cryptographic techniques lay the foundation for secure transactions by protecting the integrity and confidentiality of digital communications. Digital signatures are of particular importance among these cryptographic techniques as they enable parties to verify the authenticity and integrity of communications over the Internet. Puncturable signature (PS) is a variant of digital signature proposed by Bellare, Stepanovs and Waters [1] at EUROCRYPT 2016. It offers a fine-grained revocation of signing ability by updating the secret key
with selective messages. In contrast to a conventional digital signature, PS includes an additional algorithm known as Puncture which enables the signer to create punctured secret key with messages chosen by itself. Precisely, with the punctured secret key that has been punctured at a specific message $m$, the signer can sign on any message except for the punctured message $m$. The security definition of a PS requires that the adversary cannot forge signatures on punctured messages even though the punctured secret key is compromised.
Applications. Puncturable signatures have been identified as a versatile cryptographic primitive with numerous applications. These include improving the resilience of proof-of-stake blockchain protocols, designing disappearing signatures and securing asynchronous transaction data signing services. We delve deeper into these applications and their significance below:

- Proof of Stake (PoS) and Proof of Work (PoW) are two consensus mechanisms used in blockchain networks to validate transactions. While PoW requires substantial computational power, PoS relies on participants' cryptocurrency stake, resulting in a more energy-efficient approach. However, the majority of existing PoS protocols are prone to long-range attacks [9] [7]. In this attack, the attacker can tweak the historical records of the blockchain which could lead to double-spending of cryptocurrency or the deletion of prior transactions. PS provide a viable solution to construct practical PoS blockchain resilient to long-range attacks by enabling the selective revocation of past signatures. By puncturing prior used signatures associated with a specific stakeholder, the potential for an attacker to leverage accumulated stakes from the past and manipulate the blockchain's history is reduced. This prevents the forging of past signatures and deter long-range attacks.
- Puncturable signatures are essential building blocks for designing disappearing signature [10] in the bounded storage model. A disappearing signature refers to a signature scheme where the signature becomes inaccessible or "disappears" once the streaming of the signature stops. In the context of bounded storage model, a disappearing signature ensures that the signature can only be verified online and cannot be retained by any malicious party.
- Asynchronous transaction data signing services involve the signing and verification of transaction data in a non-interactive manner without necessitating all parties involved to be online simultaneously [15]. In this context, messages may be delayed and participants may not be available simultaneously due to factors like connectivity issues or delivery failures. PS have applications in ensuring the integrity and authenticity of transaction data in asynchronous signing services. By using PS, the transaction session identity can serve as a prefix that is subsequently punctured after the honest user signs the transaction data. This ensures that no other signature can exist for messages with the same prefix, thereby upholding the integrity of the transaction data.

Related Works. Several studies have been carried out pertaining to PS, exploring their potential applications and security properties. The notion of PS was first proposed by Bellare et al. [1] in 2016. However, their proposed scheme was based on indistinguishability obfuscation which resulted in excessive computational
overhead, rendering the scheme impractical. In a subsequent work, Halevi et al. [11] proposed a PS by combining a statistically binding commitment scheme with non-interactive zero-knowledge proofs. Their approach differed from the conventional PS schemes as it involved updating the public key instead of the secret key during each puncture operation which posed significant challenges in practical deployment. In 2020, Li et al. [13] presented a PS using a bloom filter that surpasses prior schemes in terms of signature size and algorithm efficiency. Additionally, the authors explored the application of PS in proof-of-stake blockchain protocols, specifically addressing the issue of long-range attacks caused by secret key leakage [9] [7]. However, their proposed scheme faced a notable challenge in the form of non-negligible false-positive errors, stemming from the probabilistic nature of the bloom filter data structure. Moreover, their proposed scheme was based on the Strong Diffie-Hellman (SDH) assumption in bilinear map setting and is thus susceptible to quantum attacks due to Shor's algorithm [18]. In light of the devastating consequences that quantum computers have had on the security of classical cryptosystems, Jiang et al. [12] proposed a generic construction of PS from identity-based signatures (IBS). Moreover, they presented different instantiations of their generic construction from lattice-based, pairing-based and multivariate-based assumptions. More precisely, their lattice-based instantiation leverages the efficient IBS proposed by Tian and Huang [19] and is based on the Short Integer Solution (SIS) assumption. Their pairing-based instantiation uses the identity-based version of Paterson's signature [20] which is based on the Computational Diffie-Hellman (CDH) assumption. The instantiation over multivariate assumption relies on ID-based Rainbow signature [4].

Contributions. The existing proposals for PS are undesirable for practical applications. Some PS schemes have large key and signature sizes as they rely on heavy cryptographic structures, making them computationally expensive and inefficient. The PS based on bloom filter suffers from non-negligible false-positive errors, providing economical benefits to the attackers in blockchain. Some PS schemes are prone to quantum attacks raising significant security concerns. To address these limitations, it is imperative to develop improved and more practical approaches to PS. In this work, we identify a gap in the existing literature, noting the absence of a construction for PS from isogenies. The emergence of isogeny-based cryptography as a promising candidate for post-quantum cryptosystems, characterized by its compact key sizes compared to other alternatives, has motivated us to focus on the design of an isogeny-based PS scheme. The compactness of isogeny-based cryptography makes it particularly appealing for practical applications, where efficiency and scalability are crucial factors. To show an instantiation of the generic construction of PS proposed by Jiang et al. [12], we seek an IBS scheme from isogenies. One of the main technical challenges encountered during our research is the absence of a suitable IBS based on isogenies to instantiate the generic construction. Though there exist two constructions of IBS from isogenies in the literature, none appears to be a suitable candidate to design PS. Peng et al. [16] proposed the first construction of IBS from isogenies. Unfortunately, their IBS scheme was proven to be flawed by Shaw
and Dutta [17] who provided a viable fix and designed an IBS scheme from IDbased identification scheme. However, we find that the IBS scheme of [17] has a large key and signature size, rendering it unsuitable for blockchain applications. Furthermore, both the prior IBS schemes are based on Commutative Supersingular Isogeny Diffie-Hellman (CSIDH) based group action [2] which suffers from a subexponential attack [5] leading to poor concrete efficiency. The somewhat unsatisfactory state-of-art motivates us to first design an IBS from isogenies with compact key and signature size.

The most recent and sophisticated Short Quaternion and Isogeny Signature (SQISign) by De Feo et al. [6] is the starting point in designing our IBS. The signature scheme SQISign is derived from a one-round, high soundness, interactive identification protocol. The combined size of the signature and public key of SQISign are an order of magnitude smaller than all other post-quantum signature schemes. We then employ our proposed IBS to design our PS from isogenies.
Thus, our main contributions in this paper are two-fold, as summarized below:

- Firstly, we design an IBS scheme from SQISign which we refer to as Short Quaternion and Isogeny Identity-based Signature (SQIIBS). We provide a rigorous security reduction showing it is secure against unforgeability under chosen identity and chosen message attacks (UF-IBS-CMA). We compare our scheme with the existing IBS schemes from isogenies and show that our scheme outperforms existing schemes in terms of key size and signature size which thereby reduces the storage and communication cost.
- Secondly, we employ our identity-based signature scheme SQIIBS to construct our PS from isogenies which we refer to it as Short Quaternion and Isogeny Puncturable Signature (SQIPS). We prove our scheme to be secure against existential unforgeability under chosen message attacks with adaptive puncturing (UF-CMA-AP). We also compare the features of our scheme with the existing PS schemes. Our scheme works for a pre-determined time of key punctures since the range of prefix space is fixed in advance. The size of the punctured secret key decreases linearly as the times of key puncture increase. Our scheme involves an efficient puncture operation that only contain a conversion from a bit string to a decimal integer and the deletion of a part in the current secret key. More positively, our scheme provides quantum security, enjoys small signature size and is free from false-positive errors.


## 2 Preliminaries

Let $\lambda \in \mathbb{N}$ denotes the security parameter. By $i \in[T]$, we mean $i$ belongs to the set $\{1,2, \ldots, T\}$. The symbol $\# S$ denotes the cardinality of $S$. By bin $(x)$, we mean the binary representation of $x$. A function $\epsilon(\cdot)$ is negligible if for every positive integer $c$, there exists an integer $k$ such that for all $\lambda>k,|\epsilon(\lambda)|<1 / \lambda^{c}$.

### 2.1 Quaternion Algebras, Orders and Ideals

Quaternion Algebras. For $a, b \in \mathbb{Q}^{*}=\mathbb{Q} \backslash\{0\}$, the quaternion algebra over $\mathbb{Q}$, denoted by $H(a, b)=\mathbb{Q}+i \mathbb{Q}+j \mathbb{Q}+k \mathbb{Q}$, is defined as a four-dimensional non-commutative vector space with basis $\{1, i, j, k\}$ such that $i^{2}=a, j^{2}=b$ and $k=i j=-j i$. Every quaternion algebra $H(a, b)$ is associated by a standard convolution $g: H(a, b) \rightarrow H(a, b)$ given by $g: \alpha=a_{1}+a_{2} i+a_{3} j+a_{4} k \rightarrow$ $a_{1}-a_{2} i-a_{3} j-a_{4} k=\bar{\alpha}$. The reduced norm $\mathrm{nr}: H(a, b) \rightarrow \mathbb{Q}$ of a standard convolution $g$ is the map $\mathrm{nr}: \alpha \rightarrow \alpha g(\alpha)$. In this work, we are interested in the quaternion algebra $\mathcal{B}_{p, \infty}=H(-1,-p)$ for some prime $p$.

Ideals and Orders. A fractional ideal $I=\alpha_{1} \mathbb{Z}+\alpha_{2} \mathbb{Z}+\alpha_{3} \mathbb{Z}+\alpha_{4} \mathbb{Z}$ is a $\mathbb{Z}$-lattice of rank four with $\left\{\alpha_{1}, \alpha_{2}, \alpha_{3}, \alpha_{4}\right\}$ a basis of $\mathcal{B}_{p, \infty}$. The norm of $I$, denoted by $\mathrm{nr}(I)$, is defined as the largest rational number such that $\operatorname{nr}(\alpha) \in \operatorname{nr}(I) \mathbb{Z}$ for any $\alpha \in I$. The conjugate ideal $\bar{I}$ of $I$ is given by $\bar{I}=\{\bar{\alpha} \mid \alpha \in I\}$. An order is a subring of $\mathcal{B}_{p, \infty}$ that is also a fractional ideal. A maximal order $\mathcal{O}$ is an order that is not properly contained in any other order. The left order of a fractional ideal $I$, denoted by $\mathcal{O}_{L}(I)$, is defined as $\mathcal{O}_{L}(I)=\left\{\alpha \in \mathcal{B}_{p, \infty} \mid \alpha I \subseteq I\right\}$. Similarly, right order of a fractional ideal $I$, denoted by $\mathcal{O}_{R}(I)$, is defined as $\mathcal{O}_{R}(I)=\left\{\alpha \in \mathcal{B}_{p, \infty} \mid I \alpha \subseteq I\right\}$. Here $I$ is said to be a left $\mathcal{O}_{L}(I)$-ideal or a right $\mathcal{O}_{R}(I)$-ideal or an $\left(\mathcal{O}_{L}(I), \mathcal{O}_{R}(I)\right)$-ideal. An Eichler order is the intersection of two maximal orders inside $\mathcal{B}_{p, \infty}$. A fractional ideal $I$ is called integral if $I \subseteq \mathcal{O}_{L}(I)$ or $I \subseteq \mathcal{O}_{R}(I)$. Two left $\mathcal{O}$-ideals $I$ and $J$ are equivalent if there exists $\beta \in \mathcal{B}_{p, \infty} \backslash\{0\}$ such that $I=J \beta$ and is denoted by $I \sim J$. A special extremal order is an order $\mathcal{O}$ in $\mathcal{B}_{p, \infty}$ which contains a suborder of the form $R+j R$ where $R=\mathbb{Z}[\omega] \subset \mathbb{Q}[i]$ is a quadratic order and $\omega$ has smallest norm in $\mathcal{O}$.

### 2.2 Elliptic curves, isogenies and Deuring's correspondence

Isogenies. Let $E_{1}$ and $E_{2}$ be two elliptic curves over a finite field $F$. An isogeny from $E_{1}$ to $E_{2}$ is a non-constant morphism $\varphi: E_{1} \rightarrow E_{2}$ over $F$ satisfying $\varphi\left(\Theta_{E_{1}}\right)=\Theta_{E_{2}}$ where $\Theta_{E_{i}}$ is the point at infinity of the curve $E_{i}$ for $i=1,2$. The degree of the isogeny $\varphi$, denoted by $\operatorname{deg}(\varphi)$ is its degree as a rational map. A non-zero isogeny $\varphi: E_{1} \rightarrow E_{2}$ is called separable if and only if $\operatorname{deg}(\varphi)=\# \operatorname{ker}(\varphi)$ where $\operatorname{ker}(\varphi)=\varphi^{-1}\left(\Theta_{E_{2}}\right)$ is the kernel of $\varphi$. An isogeny $\varphi$ is said to be cyclic (non-backtracking) if its kernel is a cyclic group. For any isogeny $\varphi: E_{1} \rightarrow E_{2}$, there exists a unique dual isogeny $\hat{\varphi}: E_{2} \rightarrow E_{1}$ satisfying $\varphi \circ \hat{\varphi}=[\operatorname{deg}(\varphi)]$, the multiplication-by- $\operatorname{deg}(\varphi)$ map on $E_{2}$. An isogeny from an elliptic curve $E$ to itself is called an endomorphism. The set of all endomorphisms of $E$ forms a ring under pointwise addition and composition, called the endomorphism ring of $E$ and is denoted by $\operatorname{End}(E)$. For a supersingular elliptic curve $E$, the endomorphism ring $\operatorname{End}(E)$ is isomorphic to an order in a quaternion algebra. The $j$-invariant of an elliptic curve $E: y^{2}=x^{3}+A x+B$ over $F$ is given by $j(E)=1728 \frac{4 A^{3}}{4 A^{3}+27 B^{2}}$.

Theorem 2.21. [2] Given a finite subgroup $G$ of an elliptic curve $E_{1}$, there exists a unique (up to $F$-isomorphism) elliptic curve $E_{2}$ and a separable isogeny $\varphi: E_{1} \rightarrow E_{2}$ such that $\operatorname{ker}(\varphi)=G$ and $E_{2}:=E_{1} / G$ with $\operatorname{deg}(\varphi)=\# \operatorname{ker}(\varphi)$.

Throughout this work, we focus on supersingular curves over $F=\mathbb{F}_{p^{2}}$. We fix the curve $E_{0}: y^{2}=x^{3}+x$ over $\mathbb{F}_{p^{2}}$ which has special extremal endomorphism ring $\operatorname{End}\left(E_{0}\right)=\mathcal{O}_{0}=\left\langle 1, i, \frac{i+j}{2}, \frac{1+k}{2}\right\rangle$ where $i^{2}=-1, j^{2}=-p$ and $k=i j$.

Deuring's correspondence: Deuring's correspondence [8] establishes a one-to-one correspondence between the set of isomorphism classes of supersingular curves over $\mathbb{F}_{p^{2}}$ and the set of ideal classes of a given maximal order. Under this correspondence, we look into the connection between ideals in maximal orders of quaternions and separable isogenies between supersingular curves over $\mathbb{F}_{p^{2}}$.

Theorem 2.22. Let $\varphi: E_{0} \rightarrow E_{1}$ be a separable isogeny and $\mathcal{O}_{0}=\operatorname{End}\left(E_{0}\right)$ and $\mathcal{O}_{1}=\operatorname{End}\left(E_{1}\right)$ are the maximal orders corresponding to the endomorphism rings of $E_{0}$ and $E_{1}$. Then we define the corresponding left $\mathcal{O}_{0}$-ideal $I_{\varphi}=\{\alpha \in$ $\mathcal{O}_{0} \mid \alpha(P)=\Theta_{E_{0}}$ for all $\left.P \in \operatorname{ker}(\varphi)\right\}$. Conversely, given a left $\mathcal{O}_{0}$-ideal I, we can define the kernel $E_{0}[I]=\cap_{\alpha \in I} E_{0}[\alpha]=\left\{P \in E_{0} \mid \alpha(P)=\Theta_{E_{0}}\right.$ for all $\left.\alpha \in I\right\}$ and compute the separable isogeny $\varphi_{I}: E_{0} \rightarrow E_{0} / E_{0}[I]$ that corresponds to $I$.

Lemma 2.23. [6]. Let $\mathcal{O}$ be a maximal order, I be a left $\mathcal{O}$-ideal and $\beta \in I \backslash\{0\}$. Then $\chi_{I}(\beta)=I \frac{\bar{\beta}}{n r(I)}$ is a left $\mathcal{O}$-ideal equivalent to $I$ and has norm $\frac{\operatorname{nr}(\beta)}{n r(I)}$.

Pushforward and pullback isogeny. Consider three elliptic curves $E_{0}, E_{1}$, $E_{2}$ over $\mathbb{F}_{p^{2}}$ and two separable isogenies $\varphi_{1}: E_{0} \rightarrow E_{1}$ and $\varphi_{2}: E_{0} \rightarrow E_{2}$ of coprime degrees $N_{1}$ and $N_{2}$ respectively. The pushforward of $\varphi_{1}$ by $\varphi_{2}$ is denoted by $\left[\varphi_{2}\right]_{*} \varphi_{1}$ and is defined as the separable isogeny $\left[\varphi_{2}\right]_{*} \varphi_{1}$ from $E_{2}$ to some new curve $E_{3}$ such that $\operatorname{ker}\left(\left[\varphi_{2}\right]_{*} \varphi_{1}\right)=\varphi_{2}\left(\operatorname{ker}\left(\varphi_{1}\right)\right)$ and $\operatorname{deg}\left(\left[\varphi_{2}\right]_{*} \varphi_{1}\right)=N_{1}$. Similarly, the pushforward of $\varphi_{2}$ by $\varphi_{1}$ is denoted by $\left[\varphi_{1}\right]_{*} \varphi_{2}$ and is defined as the separable isogeny $\left[\varphi_{1}\right]_{*} \varphi_{2}: E_{1} \rightarrow E_{3}$ such that $\operatorname{ker}\left(\left[\varphi_{1}\right]_{*} \varphi_{2}\right)=\varphi_{1}\left(\operatorname{ker}\left(\varphi_{2}\right)\right)$ and $\operatorname{deg}\left(\left[\varphi_{1}\right]_{*} \varphi_{2}\right)=N_{2}$. Pullback isogeny is the dual notion of pushforward isogeny. Consider two separable isogeniers $\varphi_{1}: E_{0} \rightarrow E_{1}$ and $\rho_{2}: E_{1} \rightarrow E_{3}$ of coprime degrees. The pullback of $\rho_{2}$ by $\varphi_{1}$ is denoted by $\left[\varphi_{1}\right]^{*} \rho_{2}$ and is defined as the separable isogeny $\left[\varphi_{1}\right]^{*} \rho_{2}$ from $E_{0}$ to a new curve $E_{4}$ satisfying $\left[\varphi_{1}\right]^{*} \rho_{2}=\left[\hat{\varphi}_{1}\right]_{*} \rho_{2}$.

The pushforward and pullback terms can be extended to ideals as well. Consider a $\left(\mathcal{O}_{0}, \mathcal{O}_{1}\right)$-ideal $J$ and a $\left(\mathcal{O}_{0}, \mathcal{O}_{2}\right)$-ideal $K$ where $\mathcal{O}_{0}=\operatorname{End}\left(E_{0}\right)$, $\mathcal{O}_{1}=\operatorname{End}\left(E_{1}\right)$ and $\mathcal{O}_{2}=\operatorname{End}\left(E_{2}\right)$. The pushforward of $J$ by $K$, denoted by $[K]_{*} J$ is the ideal $I_{\left[\varphi_{K}\right] * \varphi_{J}}$ corresponding to the pushforward isogeny $\left[\varphi_{K}\right]_{*} \varphi_{J}$. Consider a $\left(\mathcal{O}_{1}, \mathcal{O}_{3}\right)$-ideal $L$ where $\mathcal{O}_{1}=\operatorname{End}\left(E_{1}\right), \mathcal{O}_{3}=\operatorname{End}\left(E_{3}\right)$, then the pullback of $L$ by $J$, denoted by $[J]^{*} L$ is defined as $[J]^{*} L=[\bar{J}]_{*} L$.

Lemma 2.24. [6] Let $I$ is an ideal with left order $\mathcal{O}_{0}$ and right order $\mathcal{O}$ and $J_{1}, J_{2}$ be $\mathcal{O}_{0}$-ideals with $J_{1} \sim J_{2}$ and $\operatorname{gcd}\left(\operatorname{nr}\left(J_{1}\right), \operatorname{nr}\left(J_{2}\right), \operatorname{nr}(I)\right)=1$. Suppose that $J_{1}=\chi_{J_{2}}(\beta)$ and $\beta \in J_{2} \cap \mathcal{O}_{0} \cap \mathcal{O}$. Then $[I]_{*} J_{1} \sim[I]_{*} J_{2}$ and $[I]_{*} J_{1}=\chi_{[I]_{*} J_{2}}(\beta)$.

### 2.3 SigningKLPT algorithm

We briefly review below the sub-algorithms invoked by the algorithm SigningKLPT. The details of which can be found in the work of De Feo et al. [6].

Cornacchia $(M) \rightarrow(x, y)$ : This algorithm on input $M \in \mathbb{Z}$ either outputs $\perp$ if $M$ cannot be represented as $f(x, y)$ or returns a solution $(x, y)$ to $f(x, y)=M$.
EquivalentPrimeldeal $(I) \rightarrow L \sim I$ : This algorithm takes as input a left $\mathcal{O}_{0}$-ideal $I$ represented by Minkowski reduced basis [14] $\left(\delta_{1}, \delta_{2}, \delta_{3}, \delta_{4}\right)$. It chooses an integer $m$, generates a random element $\delta=\Sigma_{i} x_{i} \delta_{i}$ with $x_{i} \in[-m, m]$ and checks if $\frac{\operatorname{nr}(\delta)}{\operatorname{nr}(I)}$ is a prime number. If not, it continues to generate random $\delta$ until it finds a $\delta \in I$ for which $\frac{\operatorname{nr}(\delta)}{\operatorname{nr}(I)}$ is a prime number. The algorithm outputs the ideal $L=\chi_{I}(\delta)=I \frac{\bar{\delta}}{\operatorname{nr}(I)}$ equivalent to $I$ and of prime norm.
EquivalentRandomEichlerldeal $(I, N) \rightarrow L \sim I$ : This algorithm takes as input a left $\mathcal{O}_{0}$-ideal $I$ and an integer $N$ and finds a random equivalent left $\mathcal{O}_{0}$-ideal $L$ of norm coprime to $N$.
FullRepresentInteger $\mathcal{O}_{0}(M) \rightarrow \gamma$ : This algorithm takes input an integer $M \in \mathbb{Z}$ with $M>p$ and outputs an element $\gamma \in \mathcal{O}_{0}$ with $\operatorname{nr}(\gamma)=M$ as follows.
i. Sets $m^{\prime}=\left\lfloor\sqrt{\frac{4 M}{p}}\right\rfloor$ and samples a random integer $z^{\prime} \in\left[-m^{\prime}, m^{\prime}\right]$.
ii. Sets $m^{\prime \prime}=\left\lfloor\sqrt{\frac{4 M}{p}-\left(z^{\prime}\right)^{2}}\right\rfloor$ and samples a random integer $t^{\prime} \in\left[-m^{\prime \prime}, m^{\prime \prime}\right]$.
iii. Sets $M^{\prime}=4 M-p\left(\left(z^{\prime}\right)^{2}+\left(t^{\prime}\right)^{2}\right)$ and runs Cornacchia $\left(M^{\prime}\right)$ until Cornacchia returns a solution $\left(x^{\prime}, y^{\prime}\right)$ to $f\left(x^{\prime}, y^{\prime}\right)=M^{\prime}$.
iv. If $x^{\prime} \neq t^{\prime}(\bmod 2)$ or $z^{\prime} \neq y^{\prime}(\bmod 2)$ then go back to Step (i).
v. The algorithm outputs $\gamma=x+y i+z \frac{i+j}{2}+t \frac{1+k}{2} \in \mathcal{O}_{0}$ of norm $M$ where $x=\frac{x^{\prime}-t}{2}, y=\frac{y^{\prime}-z}{2}, z=z^{\prime}$ and $t=t^{\prime}$.
IdealModConstraint $(I, \gamma) \rightarrow\left(C_{0}: D_{0}\right)$ : On input a left $\mathcal{O}_{0}$-ideal $I$ of norm $N$ and an element $\gamma \in \mathcal{O}_{0}$ of norm $N n$, this algorithm outputs a projective point $\left(C_{0}: D_{0}\right) \in \mathbb{P}^{1}(\mathbb{Z} / N \mathbb{Z})$ satisfying $\gamma \mu_{0} \in I$ with $\mu_{0}=\left(C_{0}+\omega D_{0}\right) j \in R j$.
EichlerModConstraint $(I, \gamma, \delta) \rightarrow\left(C_{0}: D_{0}\right)$ : This algorithm takes input a left $\mathcal{O}_{0}$-ideal $I$ of norm $N$, elements $\gamma, \delta \in \mathcal{O}_{0}$ of norms coprime to $N$ and outputs a projective point $\left(C_{0}: D_{0}\right) \in \mathbb{P}^{1}(\mathbb{Z} / N \mathbb{Z})$ satisfying $\gamma \mu_{0} \delta \in I$ where $\mu_{0}=\left(C_{0}+\omega D_{0}\right) j \in R j$.
FullStrongApproximation $_{\mathcal{S}}(N, C, D) \rightarrow \mu$ : Taking as input a prime $N$, integers $C, D$ and a subset $\mathcal{S} \subset \mathbb{N}$, this algorithm outputs $\mu \in \mathcal{O}_{0}$ of norm in $\mathcal{S}$ satisfying $2 \mu=\lambda \mu_{0}+N \mu_{1}$ where $\mu_{0}=\left(C_{0}+\omega D_{0}\right) j \in R j, \lambda \in \mathbb{Z}$ and $\mu_{1} \in \mathcal{O}_{0}$. When $\mathcal{S}=\{d \in \mathbb{N}: d \mid D\}$ for some $D \in \mathbb{N}$, we simply write FullStrongApproximation ${ }_{D}$.
$\operatorname{CRT}_{M, N}(x, y) \rightarrow z$ : This is the algorithm for Chinese Remainder Theorem which takes as input $x \in \mathbb{Z}_{M}, y \in \mathbb{Z}_{N}$ and returns $z \in \mathbb{Z}_{M N}$ satisfying $z \equiv x$ $(\bmod M)$ and $z \equiv y(\bmod N)$ where $M$ and $N$ are coprime to each other.

We now describe the algorithm SigningKLPT ${ }_{\ell^{e}}\left(I_{\tau}, I\right)[6]$ which takes as input a prime $l$, a fixed $e \in \mathbb{N}$, a left $\mathcal{O}_{0}$ and a right $\mathcal{O}$-ideal $I_{\tau}$ of norm $N_{\tau}$ and a left $\mathcal{O}$-ideal $I$ and outputs an ideal $J \sim I$ of norm $\ell^{e}$. The steps involved in the algorithm SigningKLPT are illustrated in Fig. 1 and explicitly described below.

1. Runs the algorithm EquivalentRandomEichlerldeal $\left(I, N_{\tau}\right)$ to generate a random ideal $K \sim I$ with $\operatorname{gcd}\left(\operatorname{nr}(K), N_{\tau}\right)=1$. We denote the right order of the ideal $K$ (or $I$ ) by $\mathcal{O}_{2}$.
2. Performs the pullback of the $\left(\mathcal{O}, \mathcal{O}_{2}\right)$ - ideal $K$ by the $\left(\mathcal{O}_{0}, \mathcal{O}\right)$-ideal $I_{\tau}$ to obtain a $\left(\mathcal{O}_{0}, \mathcal{O}^{\prime}\right)$-ideal $K^{\prime}=\left[I_{\tau}\right]^{*} K$ where $\mathcal{O}^{\prime}=\operatorname{End}\left(E^{\prime}\right)$ for some curve $E^{\prime}$.
3. Computes an ideal $L=K^{\prime} \frac{\overline{\delta^{\prime}}}{\operatorname{nr}\left(K^{\prime}\right)}=\chi_{K^{\prime}}\left(\delta^{\prime}\right) \leftarrow$ EquivalentPrimeldeal $\left(K^{\prime}\right)$ equivalent to $K^{\prime}$ but of prime norm $N$ for some $\delta^{\prime} \in K^{\prime}$. (See Lemma 2.23)
4. Chooses $e_{0} \in \mathbb{N}$ and runs the algorithm FullRepresentInteger $\mathcal{O}_{0}\left(N \ell^{e_{0}}\right)$ to obtain an element $\gamma \in \mathcal{O}_{0}$ such that $\operatorname{nr}(\gamma)=N \ell^{e_{0}}$. Sets $e_{1}=e-e_{0} \in \mathbb{N}$.
5. Finds the projective point $\left(C_{0}: D_{0}\right) \in \mathbb{P}^{1}(\mathbb{Z} / N \mathbb{Z}) \leftarrow$ IdealModConstraint $(L, \gamma)$ satisfying $\gamma \mu_{0} \in L$ where $\mu_{0}=\left(C_{0}+\omega D_{0}\right) j \in R j$.
6. Chooses $\delta \in \mathcal{O}_{0}$ with $\operatorname{gcd}\left(\operatorname{nr}(\delta), N_{\tau}\right)=1$ and runs the algorithm EichlerModConstraint $\left(\mathbb{Z}+I_{\tau}, \gamma, \delta\right)$ on input the ideal $\mathbb{Z}+I_{\tau}$ of norm $N_{\tau}$ and elements $\gamma, \delta \in \mathcal{O}_{0}$ of norms coprime to $N_{\tau}$ to find the projective point $\left(C_{1}\right.$ : $\left.D_{1}\right) \in \mathbb{P}^{1}\left(\mathbb{Z} / N_{\tau} \mathbb{Z}\right)$ satisfying $\gamma \mu_{1} \delta \in \mathbb{Z}+I_{\tau}$ where $\mu_{1}=\left(C_{1}+\omega D_{1}\right) j \in R j$.
7. Computes $C \leftarrow \operatorname{CRT}_{N, N_{\tau}}\left(C_{0}, C_{1}\right)$ where $C$ is the solution modulo $N N_{\tau}$ to the system of congruences $C \equiv C_{0}(\bmod N)$ and $C \equiv C_{1}\left(\bmod N_{\tau}\right)$ and $D \leftarrow \operatorname{CRT}_{N, N_{\tau}}\left(D_{0}, D_{1}\right)$ where $D$ is the solution modulo $N N_{\tau}$ to the system of congruences $D \equiv D_{0}(\bmod N)$ and $D \equiv D_{1}\left(\bmod N_{\tau}\right)$. If $\ell^{e} p\left(C^{2}+D^{2}\right)$ is not a quadratic residue, go back to Step 4 and repeat the process.
8. Executes the algorithm FullStrongApproximation $\ell_{\star}\left(N N_{\tau}, C, D\right)$ to generate $\mu \in \mathcal{O}_{0}$ of norm $\ell^{e_{1}}$ where $\ell^{\star}=\left\{\ell^{\alpha}: \alpha \in \mathbb{N}\right\}$.
9. Sets $\beta=\gamma \mu$, obtains the $\left(\mathcal{O}_{0}, \mathcal{O}^{\prime}\right)$-ideal $\chi_{L}(\beta)=L \frac{\bar{\beta}}{\operatorname{nr}(L)}$ (See Lemma 2.23) and computes the $\left(\mathcal{O}, \mathcal{O}_{2}\right)$ - ideal $J=\left[I_{\tau}\right]_{*} \chi_{L}(\beta)$ by using pushforward of the ideal $\chi_{L}(\beta)$ by the $\left(\mathcal{O}_{0}, \mathcal{O}\right)$-ideal $I_{\tau}$. (See Lemma 2.24)
10. The algorithm then returns the ideal $J \sim I$.


Fig. 1. Pictorial description of SigningKLPT algorithm
Correctness. Step 5 and Step 8 ensure $\beta \in L$ whereas Step 6 ensures $\beta \in$ $\mathbb{Z}+I_{\tau}$. Also we have, $\operatorname{nr}(\beta)=\operatorname{nr}(\gamma) \operatorname{nr}(\mu)=N \ell^{e_{0}} \dot{\ell}^{e_{1}}=N \ell^{e}$ which implies $\operatorname{nr}(J)$ $=\operatorname{nr}\left(\left[I_{\tau}\right]_{*} \chi_{L}(\beta)\right)=\frac{\operatorname{nr}(\beta)}{\operatorname{nr}(L)}=\frac{N \ell^{e}}{N}=\ell^{e}$. Also, Lemma 2.24 applied to $\chi_{L}(\beta)=$ $L \frac{\bar{\beta}}{\operatorname{nr}(L)}=\chi_{K^{\prime}}\left(\delta^{\prime}\right) \frac{\bar{\beta}}{\operatorname{rr}(L)}=K^{\prime} \frac{\bar{\delta}^{\prime}}{\operatorname{nr}\left(K^{\prime}\right)} \frac{\bar{\beta}}{\operatorname{nr}(L)}=\chi_{K^{\prime}}\left(\frac{\overline{\beta \delta^{\prime}}}{\operatorname{nr}(L)}\right)$ implies that $\left[I_{\tau}\right]_{*} \chi_{L}(\beta) \sim$ $\left[I_{\tau}\right]_{*} K^{\prime}$. This proves $J \sim K$ and we also have $K \sim I$, which implies $J \sim I$.

### 2.4 Signature Scheme

Definition 2.41. A signature scheme associated with a message space $\mathcal{M}$ is a tuple of probabilistic polynomial-time (PPT) algorithms Sig = (Setup, KeyGen, Sign, Verify) with the following syntax:
Sig.Setup $\left(1^{\lambda}\right) \rightarrow \mathrm{pp}:$ A trusted party taking input $1^{\lambda}$ outputs the public parameter pp and makes it publicly available.
Sig. KeyGen $(\mathrm{pp}) \rightarrow(\mathrm{sk}, \mathrm{pk})$ : On input pp , the user runs this algorithm to generate a signing and verification key pair (sk, pk).
Sig.Sign $(\mathrm{pp}, \mathrm{sk}, m) \rightarrow \sigma:$ Taking input pp, sk and a message $m \in \mathcal{M}$, the signer executes this algorithm to generate a signature $\sigma$ on the message $m$.
Sig.Verify $(\mathrm{pp}, \mathrm{pk}, m, \sigma) \rightarrow$ Valid/Invalid : On input pp, pk, $m \in \mathcal{M}$ and a signature $\sigma$, the verifier checks the validity of the signature $\sigma$ on $m$.

Correctness. For all pp $\leftarrow \operatorname{Sig} . \operatorname{Setup}\left(1^{\lambda}\right)$, all (sk, pk) $\leftarrow \operatorname{Sig}$.KeyGen $(\mathrm{pp})$ and all signature $\sigma \leftarrow \operatorname{Sig} . \operatorname{Sign}(\mathrm{pp}, \mathrm{sk}, m)$, it holds that

$$
\text { Sig.Verify }(\mathrm{pp}, \mathrm{pk}, m, \sigma)=\text { Valid }
$$

Definition 2.42. A signature scheme Sig is secure against existential unforgeability under chosen-message attacks (UF-CMA) if for all PPT adversary $\mathcal{A}$, there exists a negligible function $\epsilon$ such that

$$
\operatorname{Adv} \operatorname{Adig}, \mathcal{A}_{\mathrm{UF}}^{\mathrm{SMA}}(\lambda)=\operatorname{Pr}\left[\mathcal{A} \text { wins in } \operatorname{Exp}_{\mathrm{Sig}, \mathcal{A}}^{\mathrm{UF}-\mathrm{A} A}(\lambda)\right]<\epsilon
$$

where the experiment $\operatorname{Exp}_{\mathrm{Sig}, \mathcal{A}}^{\mathrm{UF}-\mathrm{CMA}}(\lambda)$ is depicted in Fig.2.

> Setup: The challenger $\mathcal{C}$ generates the public parameter $\mathrm{pp} \leftarrow \operatorname{Sig}$.Setup ( $1^{\lambda}$ ) and secret-public key pair $(\mathrm{sk}, \mathrm{pk}) \leftarrow$ Sig.KeyGen $(\mathrm{pp})$. It forwards pp and pk to the adversary $\mathcal{A}$ while keeps sk secret to itself. It also maintains a list SList and initializes SList to $\emptyset$.
> Query Phase: $\mathcal{A}$ issues polynomially many adaptive signature queries to the following oracle:
> - $\mathcal{O}_{\mathrm{S}}(\mathrm{sk}, \cdot)$ : On receiving a signature query on a message $m$, the challenger $\mathcal{C}$ checks if $m \notin \mathcal{M}$. If the check succeeds, it returns $\perp$. Otherwise, it computes a signature $\sigma \leftarrow$ Sig.Sign (pp, sk, $m$ ) on the message $m$ under the secret key sk and updates SList $\leftarrow$ SList $\cup$ $\{m\}$. It returns the computed signature $\sigma$ to the adversary $\mathcal{A}$.
> Forgery: The adversary $\mathcal{A}$ eventually submits a forgery ( $m^{*}, \sigma^{*}$ ). The adversary $\mathcal{A}$ wins the game if $m^{*} \notin$ SList and Valid $\leftarrow \operatorname{Sig} . \operatorname{Verify}\left(\mathrm{pp}, \mathrm{pk}, m^{*}, \sigma^{*}\right)$.

Fig. 2. $\operatorname{Exp}_{\mathrm{Si}_{\mathrm{I}}, \mathcal{A}}^{\mathrm{UF}-\mathrm{CM}}(\lambda)$ : Existential unforgeability under chosen-message attack

### 2.5 SQISign: an isogeny-based signature scheme

The signature scheme SQISign [6] comprises of four PPT algorithms (Setup, KeyGen, Sign, Verify) having the following interface:
SQISign.Setup $\left(1^{\lambda}\right) \rightarrow \mathrm{pp}_{\mathrm{sgn}}$ : A trusted authority runs this algorithm on input a security parameter $1^{\lambda}$ and performs the following steps:
i. Chooses a prime $p$ and fixes the supersingular curve $E_{0}: y^{2}=x^{3}+x$ over $\mathbb{F}_{p^{2}}$ with special extremal endomorphism ring $\operatorname{End}\left(E_{0}\right)=\mathcal{O}_{0}=\left\langle 1, i, \frac{i+j}{2}, \frac{1+k}{2}\right\rangle$.
ii. Picks a smooth number $D=2^{e}$ where $2^{e}>p^{3}$.
iii. Picks an odd smooth number $D_{c}=\ell^{e}$ where $\ell$ is a prime and $e \in \mathbb{N}$ and computes $\mu\left(D_{c}\right)=(\ell+1) \cdot \ell^{e-1}$.
iv. Samples a cryptographic hash function $\mathcal{H}_{1}: \mathbb{F}_{p^{2}} \times\{0,1\}^{*} \rightarrow\left[\mu\left(D_{c}\right)\right]$.
v. Samples an arbitrary function $\Phi_{D_{c}}(E, s)$ that maps a curve $E$ and an integer $s \in\left[\mu\left(D_{c}\right)\right]$ to a non-backtracking isogeny of degree $D_{c}$ from $E[3]$.
vi. Sets the public parameter $\mathrm{pp}_{\mathrm{sgn}}=\left(p, E_{0}, D_{c}, D, \mathcal{H}_{1}, \Phi_{D_{c}}\right)$.

SQISign.KeyGen $\left(\mathrm{pp}_{\mathrm{sgn}}\right) \rightarrow(\mathrm{sk}, \mathrm{pk})$ : On input $\mathrm{pp}_{\mathrm{sgn}}$, the key generation algorithm run by a user generates a signing-verification key pair (sk, pk) as follows:
i. Picks a random isogeny $\tau: E_{0} \rightarrow E_{A}$ of degree $N_{\tau}$.
ii. Sets the signing key $\mathrm{sk}=\tau$ and verification key $\mathrm{pk}=E_{A}$.

SQISign.Sign $\left(\mathrm{pp}_{\mathrm{sgn}}\right.$, $\left.\mathrm{sk}, m\right) \rightarrow \sigma$ : Taking input $\mathrm{pp}_{\mathrm{sgn}}$, signing key $\mathrm{sk}=\tau$ and a message $m \in\{0,1\}^{*}$, the signer generates a signature $\sigma$ on $m$ as follows:
i. Picks a random commitment isogeny $\psi: E_{0} \rightarrow E_{1}$.
ii. Computes $s=\mathcal{H}_{1}\left(j\left(E_{1}\right), m\right)$ and sets the challenge isogeny $\Phi_{D_{c}}\left(E_{1}, s\right)=\varphi$ where $\varphi: E_{1} \rightarrow E_{2}$ is a non-backtracking isogeny of degree $D_{c}$.
iii. Computes $\bar{I}_{\tau}, I_{\tau}, I_{\psi}$ and $I_{\varphi}$ corresponding to $\hat{\tau}, \tau, \psi$ and $\varphi$ respectively.
iv. The signer having the knowledge of $\mathcal{O}=\operatorname{End}\left(E_{A}\right)$ through sk $=\tau$ and $\mathcal{O}_{2}=$ $\operatorname{End}\left(E_{2}\right)$ through $\varphi \circ \psi: E_{0} \rightarrow E_{2}$, executes the algorithm SigningKLPT $2^{e}\left(I_{\tau}, I\right)$ described in Section 2.3 on input the $\left(\mathcal{O}_{0}, \mathcal{O}\right)$-ideal $I_{\tau}$ and the left $\mathcal{O}$-ideal $I=I_{\varphi} I_{\psi} \bar{I}_{\tau}$ to obtain a $\left(\mathcal{O}, \mathcal{O}_{2}\right)$-ideal $J \sim I$ of norm $D=2^{e}$.
v. Constructs a cyclic isogeny $\eta: E_{A} \rightarrow E_{2}$ of degree $D$ corresponding to the ideal $J$ such that $\hat{\varphi} \circ \eta$ is cyclic. The signature is the pair $\sigma=\left(E_{1}, \eta\right)$.
SQISign.Verify $\left(\mathrm{pp}_{\mathrm{sgn}}, \mathrm{pk}, m, \sigma\right) \rightarrow$ Valid/Invalid: The verifier verifies the validity of the signature $\sigma=\left(E_{1}, \eta\right)$ on the message $m$ as follows:
i. Computes $s=\mathcal{H}_{1}\left(j\left(E_{1}\right), m\right)$ and then recovers the isogeny $\Phi_{D_{c}}\left(E_{1}, s\right)=\varphi$.
ii. Checks if $\eta$ is an isogeny of degree $D$ from $E_{A}$ to $E_{2}$ and that $\hat{\varphi} \circ \eta: E_{A} \rightarrow E_{1}$ is cyclic.
iii. If all the checks succeed returns Valid, otherwise returns Invalid.

Correctness. It follows from the correctness of SigningKLPT algorithm.

## 3 Security Aspect of SQISign

To prove the security of the signature scheme SQISign, the authors resort to a computational assumption that formalises the idea that the isogeny $\eta$ corresponding to the ideal $J$ returned by the algorithm SigningKLPT is indistinguishable from a random isogeny of the same degree. Before defining the problem formally, we analyze the structure of $\eta$.

Lemma 3.01. [6] Consider the ideal $L$ and element $\beta \in L$ computed as in steps 3, 9 respectively of the algorithm SigningKLPT described in Section 2.3. The isogeny $\eta$ corresponding to the output $J$ of SigningKLPT algorithm is equal to $\eta=[\tau]_{* \iota}$ where $\iota$ is an isogeny of degree $\ell^{e}$ satisfying $\beta=\hat{\iota} \circ \varphi_{L}$.

We recall the following notations before defining the (computationally) indistinguishable problem underlying the security of SQISign.
$\mathcal{U}_{L, N_{\tau}}$ : For a given ideal $L$ of norm $N, \mathcal{U}_{L, N_{\tau}}$ denotes the set of all isogenies $\iota$ computed in Lemma 3.01 from elements $\beta=\gamma \mu \in L$ where $\gamma$ is any
possible output of the algorithm FullRepresentInteger ${ }_{\mathcal{O}_{0}}$ and $\mu$ is computed by algorithm FullStrongApproximation in Step 8 of SigningKLPT.
$\mathcal{P}_{N_{\tau}}$ : We define $\mathcal{P}_{N_{\tau}}=\bigcup_{\mathcal{C} \in \mathrm{CI}(\mathcal{O})} \mathcal{U}_{\mathcal{C}, N_{\tau}}$ where we write $\mathcal{U}_{\mathcal{C}, N_{\tau}}$ for $\mathcal{U}_{L, N_{\tau}}$ where $L \leftarrow$ EquivalentPrimeldeal $(\mathcal{C})$ for an equivalence class $\mathcal{C}$ in the ideal class group $\mathrm{Cl}\left(\mathcal{O}_{0}\right)$ of $\mathcal{O}_{0}$.
Iso $_{D, j(E)}$ : Denotes the set of cyclic isogenies of degree $D$ whose domain is a curve inside the isomorphism class of $E$.
$[\tau]_{*} \mathcal{P}$ : Denotes the subset $\left\{[\tau]_{*} \mid \varphi \in \mathcal{P}\right\}$ of Iso $_{D, j\left(E_{0}\right)}$ where $\mathcal{P}$ is a subset of $\mathrm{Iso}_{D, j(E)}$ and $\tau: E \rightarrow E_{0}$ is an isogeny with $\operatorname{gcd}(\operatorname{deg}(\tau), D)=1$.
$\mathcal{K}$ : a probability distribution on the set of cyclic isogenies whose domain is $E_{0}$, representing the distribution of SQISign private keys.

Definition 3.02. [6] Let $p$ be a prime and $D$ be a smooth integer. Let $\tau: E_{0} \rightarrow$ $E_{A}$ be a random isogeny drawn from $\mathcal{K}$ and let $N_{\tau}$ be its degree. Let Oracle ${ }_{\tau}$ be an oracle sampling random elements in $[\tau] * \mathcal{P}_{N_{\tau}}$. Let $\eta$ be an isogeny of degree $D$ whose domain curve is $E$. Given $p, D, \mathcal{K}, E_{A}, \eta$ and a polynomial number of queries to $\mathrm{Oracle}_{\tau}$, the Real or Random Isogeny problem is to determine where

1. whether $\eta$ is uniformly random in $\operatorname{Iso}_{D, j\left(E_{A}\right)}$
2. or $\eta$ is uniformly random in $[\tau]_{*} \mathcal{P}_{N_{\tau}}$.

Informally speaking, the problem states that the ideals output by the algorithm SigningKLPT are indistinguishable from uniformly random ideals of the same norm. The hardness assumption underlying the security of SQISign is the Real or Random Isogeny problem defined in Definition 3.02.
Theorem 3.03. [6] The scheme SQISign is UF-CMA secure under the hardness of Real or Random Isogeny Problem defined in Definition 3.02.

### 3.1 Identity-based signature

Definition 3.11. An identity-based signature is a tuple IBS = (Setup, Extract, Sign, Verify) of four PPT algorithms with the following syntax:
IBS.Setup $\left(1^{\lambda}\right) \rightarrow\left(\mathrm{pp}_{\mathrm{ibs}}\right.$, msk): The key generation centre (KGC) on input $1^{\lambda}$ generates a public parameter $\mathrm{pp}_{\mathrm{ibs}}$ and a master secret key msk.
IBS.Extract $\left(\mathrm{pp}_{\mathrm{ibs}}\right.$, msk, id $) \rightarrow$ uskid: The KGC runs this key extract algorithm on input the public parameter $\mathrm{pp}_{\mathrm{ibs}}$, the master secret key msk and user identity id. It generates the user secret key usk ${ }_{i d}$ for the given identity id.
IBS.Sign $\left(\mathrm{pp}_{\mathrm{ibs}}\right.$, usk $\left.\mathrm{id}_{\mathrm{id}}, m\right) \rightarrow \sigma$ : Taking input the public parameter $\mathrm{pp}_{\mathrm{ibs}}$, user secret key usk ${ }_{i d}$ and a message $m$, the signer executes this randomized algorithm and outputs a signature $\sigma$ on the message $m$.
IBS.Verify $\left(\mathrm{pp}_{\mathrm{ibs}}\right.$, id, $\left.m, \sigma\right) \rightarrow$ Valid/Invalid: The verifier runs this deterministic algorithm on input the public parameter $\mathrm{pp}_{\mathrm{ibs}}$, an identity id, a message $m$ and a signature $\sigma$ to verify the validity of the signature $\sigma$.
Correctness. For all $\left(\mathrm{pp}_{\mathrm{ibs}}, \mathrm{msk}\right) \leftarrow \mathrm{IBS}$. Setup $\left(1^{\lambda}\right)$, all usk $\mathrm{id}_{\mathrm{id}} \leftarrow \operatorname{IBS}$. Extract $\left(\mathrm{pp}_{\mathrm{ibs}}\right.$, msk, id), all $m$ and all id, it holds that

$$
\operatorname{IBS} . V e r i f y\left(\mathrm{pp}_{\mathrm{ibs}}, \mathrm{id}, m, \mathrm{IBS} . \operatorname{Sign}\left(\mathrm{pp}_{\mathrm{ibs}}, \text { usk }_{\mathrm{id}}, m\right)\right) \rightarrow \text { Valid. }
$$

Definition 3.12. An IBS scheme is said to be secure against unforgeability under chosen identity and chosen message attacks (UF-IBS-CMA) if for all PPT adversary $\mathcal{A}$, there exists a negligible function $\epsilon$ such that

$$
\operatorname{Adv}_{\mathrm{IBS}}^{\mathrm{UF}, \mathcal{A}} \mathrm{~A}-\mathrm{CMA}(\lambda)=\operatorname{Pr}\left[\mathcal{A} \text { wins in } \operatorname{Exp}_{\mathrm{IBS}, \mathcal{A}}^{\mathrm{UF}-\mathrm{ABS}-\mathrm{CMA}}(\lambda)\right]<\epsilon
$$

where the experiment $\operatorname{Exp} \operatorname{IFSS}, \mathcal{A}$-IBS-CMA $(\lambda)$ that formalizes the unforgeability game is described in Fig.3.

Setup: The challenger $\mathcal{C}$ takes input the security parameter $1^{\lambda}$ and generate $\left(\mathrm{pp}_{\mathrm{ibs}}, \mathrm{msk}\right) \leftarrow$ IBS.Setup $\left(1^{\lambda}\right)$. It gives the public parameter $\mathrm{pp}_{\mathrm{ibs}}$ to $\mathcal{A}$ while keeps the master secret key msk secret to itself. Also it maintains three lists Klist, Clist and Mlist and initializes each to $\emptyset$.
Query Phase: $\mathcal{C}$ responds to polynomially many adaptive queries made by $\mathcal{A}$ as follows:

- Oracle $\mathcal{O}_{\text {Extract }}(\cdot):$ On receiving a query on a user identity id from $\mathcal{A}, \mathcal{C}$ checks whether (id, usk ${ }_{\text {id }}$ ) $\in$ Kist. If so, it returns usk ${ }_{\text {id }}$ and appends id to CList. Otherwise, it generates usk ${ }_{\text {id }} \leftarrow$ IBS.Extract $\left(\mathrm{pp}_{\mathrm{ibs}}\right.$, msk, id), returns usk $\mathrm{id}_{\mathrm{id}}$ and appends (id, usk id ) to Klist and id to Clist.
- Oracle $\mathcal{O}_{\text {Sign }}(\cdot)$ : On receiving a query on a message $m$ and a user identity id from $\mathcal{A}, \mathcal{C}$ computes usk id as in the extraction query, except for appending identity id to Clist. It then computes a signature $\sigma \leftarrow \operatorname{IBS} . \operatorname{Sign}\left(\mathrm{pp}_{\mathrm{ibs}}\right.$, usk $\left.\mathrm{id}, m\right)$ and appends ( $m$, id, $\sigma$ ) to Mlist.
Forgery: The adversary $\mathcal{A}$ eventually outputs a message $m^{*}$, user identity id* and a forge signature $\sigma^{*}$. The adversary $\mathcal{A}$ wins the game if IBS.Verify $\left(\mathrm{pp}_{\mathrm{ibs}}, \mathrm{id}^{*}, m^{*}, \sigma^{*}\right) \rightarrow$ Valid with the restriction that id ${ }^{*} \notin$ Clist and $\left(m^{*}, \mathrm{id}^{*}, \cdot\right) \notin$ Mlist.

Fig. 3. $\operatorname{Exp}_{\mathrm{IBS}, \mathcal{A}}^{\mathrm{UF}-\mathrm{IBS}-\mathrm{CMA}}(\lambda)$ : Unforgeability under chosen identity and chosen message attacks

### 3.2 Puncturable Signature Scheme

Definition 3.21. A puncturable signature is a tuple $P S=(P S . S e t u p, P S . P u n c t u r e$, PS.Sign, PS.Verify) of PPT algorithms associated with a message space $\mathcal{M}$ and prefix space $\mathcal{P}$ that satisfy the following requirements. Note that, if $x \in \mathcal{P}$, then there exists some $m \in M$ with prefix $x$ and every message $m$ has a unique prefix.
PS.Setup $\left(1^{\lambda}\right) \rightarrow\left(\mathrm{pp}_{\mathrm{ps}}, \mathrm{sk}_{0}\right)$ : On input $1^{\lambda}$, the signer executes this algorithm to generate the public parameter $\mathrm{pk}_{\mathrm{ps}}$ and initial secret key $\mathrm{sk}_{0}$.
PS.Puncture(sk, $\left.x^{\prime}\right) \rightarrow \mathrm{sk}^{\prime}$ : The signer takes as input its secret key sk and a prefix $x^{\prime} \in \mathcal{P}$ and runs this randomized algorithm to output an updated secret key $s k^{\prime}$. We say the prefix $x^{\prime}$ has been punctured and refer the updated secret key $\mathrm{sk}^{\prime}$ as a punctured secret key.
PS.Sign $\left(\mathrm{pp}_{\mathrm{ps}}\right.$, sk, $\left.m\right) \rightarrow \Sigma / \perp$ : Taking input $\mathrm{pp}_{\mathrm{ps}}$, secret key sk and a message $m \in \mathcal{M}$, the signer runs this randomized algorithm to generate a signature $\Sigma$ if the prefix $x^{\prime} \in \mathcal{P}$ has not been punctured. Otherwise, it returns $\perp$.
PS.Verify $\left(\mathrm{pp}_{\mathrm{ps}}, m, \Sigma\right) \rightarrow$ Valid/Invalid: This is a deterministic algorithm that takes as input the public parameter $\mathrm{pp}_{\mathrm{ps}}$, a message $m$ and a signature $\Sigma$. It outputs Valid if $\Sigma$ is a valid signature on $m$ and Invalid otherwise.
Correctness. The scheme PS is correct if it satisfies the following conditions:
i. For any message $m \in \mathcal{M}$, any prefix $x^{\prime} \in \mathcal{P}$ and any $\left(\mathrm{pp}_{\mathrm{ps}}, \mathrm{sk}_{0}\right) \leftarrow \operatorname{PS} . \operatorname{Setup}\left(1^{\lambda}\right)$, it holds that PS.Verify $\left(\mathrm{pp}_{\mathrm{ps}}, m, \operatorname{PS} . \operatorname{Sign}\left(\mathrm{pp}_{\mathrm{ps}}, \mathrm{sk}_{0}, m\right)\right) \rightarrow$ Valid where $\mathrm{sk}_{0}$ is the initial non-punctured secret key.
ii. For any message $m \in \mathcal{M}$ with prefix $x^{\prime} \in \mathcal{P}$ which has been punctured with secret key sk, it holds that PS.Verify $\left(\mathrm{pp}_{\mathrm{ps}}, m, \operatorname{PS} . \operatorname{Sign}\left(\mathrm{pp}_{\mathrm{ps}}, \mathrm{sk}^{\prime}, m\right)\right) \rightarrow$ Invalid where $s k^{\prime} \leftarrow \mathrm{PS}$.Puncture(sk, $x^{\prime}$ ) is the punctured secret key corresponding to the prefix $x^{\prime}$.
iii. For any message $m \in \mathcal{M}$ with prefix $x \in \mathcal{P}$ which has not been punctured, we have PS.Verify $\left(\mathrm{pp}_{\mathrm{ps}}, m\right.$, PS. $\left.\operatorname{Sign}\left(\mathrm{pp}_{\mathrm{ps}}, \mathrm{sk}^{\prime}, m\right)\right) \rightarrow$ Valid where $\mathrm{sk}^{\prime} \leftarrow$ PS.Puncture(sk, $x^{\prime}$ ) is the punctured secret key corresponding to the prefix $x^{\prime} \neq x$ of a message $m^{\prime}$ with $m^{\prime} \neq m$.

Definition 3.22. A puncturable signature scheme PS is secure against existential unforgeability under chosen-message attacks with adaptive puncturing (UF-CMA-AP) if for all PPT adversary $\mathcal{A}$, there exists a negligible function $\epsilon$ such that

$$
\operatorname{Adv}_{\mathrm{PS}, \mathcal{A}}^{\mathrm{UF}-\mathrm{CMA}-\mathrm{AP}}(\lambda)=\operatorname{Pr}\left[\mathcal{A} \text { wins in } \operatorname{Exp}_{\mathrm{PS}, \mathcal{A}}^{\mathrm{UF}-\mathrm{A}} \mathrm{AP}(\lambda)\right]<\epsilon
$$

where the experiment $\operatorname{Exp} \mathrm{EPS}, \mathcal{A}_{\mathrm{UF}} \mathrm{CMA}-\mathrm{AP}(\lambda)$ is described in Fig. 4.

Setup: The challenger $\mathcal{C}$ takes input the security parameter $1^{\lambda}$ and generates $\left(\mathrm{pp}_{\mathrm{ps}}, \mathrm{sk}_{0}\right) \leftarrow$ PS.Setup $\left(1^{\lambda}\right)$. It forwards $\mathrm{pp}_{\mathrm{ps}}$ to $\mathcal{A}$ while keeps sk secret to itself. It also maintains the set $\mathcal{Q}_{\text {sig }}$ for signed messages and the set $\mathcal{Q}_{\text {pun }}$ for punctured prefixes and initializes each to $\emptyset$.
Query Phase: The adversary $\mathcal{A}$ issues polynomially many adaptive queries to the oracles $\mathcal{O}_{\text {Puncture }}(\mathrm{sk}, \cdot)$ and $\mathcal{O}_{\mathrm{sign}}\left(\mathrm{pp}_{\mathrm{ps}}\right.$, sk, $\left.\cdot\right)$ as follows:

- $\mathcal{O}_{\text {Puncture }}(\mathrm{sk}, \cdot)$ : Upon receiving a query on prefix $x^{\prime}$, the challenger $\mathcal{C}$ generates a punctured secret key sk ${ }^{\prime} \leftarrow$ Puncture $\left(\right.$ sk, $\left.x^{\prime}\right)$ and updates $\mathcal{Q}_{\text {pun }} \leftarrow \mathcal{Q}_{\text {pun }} \cup\left\{x^{\prime}\right\}$.
$-\mathcal{O}_{\mathrm{Sgn}}(\mathrm{sk}, \cdot)$ : On receiving a signature query on a message $m$ with prefix $x^{\prime} \in \mathcal{P}$, the challenger $\mathcal{C}$ checks if $x^{\prime} \in \mathcal{Q}_{\text {pun }}$. If the check succeeds, it returns $\perp$. Otherwise, it computes the signature $\Sigma \leftarrow$ PS.Sign $\left(\mathrm{pp}_{\mathrm{ps}}\right.$, sk, $\left.m\right)$ on the message $m$ and updates $\mathcal{Q}_{\text {sig }} \leftarrow$ $\mathcal{Q}_{\text {sig }} \cup\{m\}$. It returns the computed signature $\Sigma$ to the adversary $\mathcal{A}$.
Challenge: The adversary $\mathcal{A}$ sends a target prefix $x^{*}$ to the challenger $\mathcal{C}$ and issues additional puncture and signature queries as described in the Query phase.
Corruption Query: $\mathcal{C}$ returns the current secret key sk* if $x^{*} \in \mathcal{Q}_{\text {pun }}$ and $\perp$ otherwise.
Forgery: The adversary $\mathcal{A}$ eventually submits a forgery $\left(m^{*}, \Sigma^{*}, x^{*}\right)$ where $x^{*}$ is the prefix of $m^{*} . \mathcal{A}$ wins the game if $m^{*} \notin \mathcal{Q}_{\text {sig }}, x^{*} \in \mathcal{Q}_{\text {pun }}$ and Valid $\leftarrow \operatorname{PS} . V e r i f y\left(p_{\mathrm{ps}}, m^{*}, \Sigma^{*}\right)$.

Fig. 4. $\operatorname{Exp}_{\mathrm{PS}, \mathcal{A}}^{\mathrm{UF}-\mathrm{CMA}-\mathrm{AP}}(\lambda)$ : Existential unforgeability under chosen-message attacks with adaptive puncturing

## 4 Our Identity-based Signature from SQISign

In this section, we propose our identity-based signature from SQISign. We refer to our scheme as Short Quaternion and Isogeny Identity-based Signatures (SQIIBS). SQIIBS.Setup $\left(1^{\lambda}\right) \rightarrow\left(\mathrm{pp}_{\mathrm{ibs}}\right.$, msk): A KGC on input the security parameter $1^{\lambda}$ generates the public parameter $\mathrm{pp}_{\mathrm{ibs}}$ and a master secret key msk as follows:
i. Same as the algorithm SQISign.Setup described in Section 2.5. Additionally, it picks a random isogeny $\tau_{1}: E_{0} \rightarrow E_{A}^{(1)}$.
ii. Publishes the public parameter $\mathrm{pp}_{\mathrm{ibs}}=\left(p, E_{0}, D_{c}, D, \mathcal{H}_{1}, \Phi_{D_{c}}, E_{A}^{(1)}\right)$ and keeps the master secret key msk $=\tau_{1}$ secret to itself.
SQIIBS.Extract $\left(\mathrm{pp}_{\mathrm{ibs}}, \mathrm{msk}, \mathrm{id}\right) \rightarrow$ usk id : On input the public parameter $\mathrm{pp}_{\mathrm{ibs}}=$ $\left(p, E_{0}, D_{c}, D, \mathcal{H}_{1}, \Phi_{D_{c}}, E_{A}^{(1)}\right)$, master secret key msk $=\tau_{1}$ and an identity id, the KGC executes this algorithm to generate the user secret key usk ${ }_{\text {id }}$ as follows:
i. Picks a random isogeny $\tau_{2}: E_{0} \rightarrow E_{A}^{(2)}$.
ii. Selects a random commitment isogeny $\psi_{1}: E_{0} \rightarrow E_{1}^{(1)}$.
iii. Computes $s_{1}=\mathcal{H}_{1}\left(j\left(E_{1}^{(1)}\right), \operatorname{bin}\left(j\left(E_{A}^{(2)}\right)\right) \|\right.$ id) and sets $\Phi_{D_{c}}\left(E_{1}^{(1)}, s_{1}\right)=\varphi_{1}$ where $\varphi_{1}: E_{1}^{(1)} \rightarrow E_{2}^{(1)}$ is a non-backtracking isogeny of degree $D_{c}$.
iv. Computes the ideals $\bar{I}_{\tau_{1}}, I_{\tau_{1}}, I_{\psi_{1}}$ and $I_{\varphi_{1}}$ corresponding to the isogenies $\hat{\tau_{1}}$, $\tau_{1}, \psi_{1}$ and $\varphi_{1}$ respectively.
v. The KGC having the knowledge of $\mathcal{O}^{(1)}=\operatorname{End}\left(E_{A}^{(1)}\right)$ through $\tau_{1}$ and $\mathcal{O}_{2}^{(1)}=$ $\operatorname{End}\left(E_{2}^{(1)}\right)$ through $\varphi_{1} \circ \psi_{1}: E_{0} \rightarrow E_{2}^{(1)}$, executes the SigningKLPT $2_{2^{e}}\left(I_{\tau_{1}}, I_{1}\right)$ algorithm (Section 2.3) on input the $\left(\mathcal{O}_{0}, \mathcal{O}^{(1)}\right)$-ideal $I_{\tau_{1}}$ and a left $\mathcal{O}^{(1)}$-ideal $I_{1}=I_{\varphi_{1}} I_{\psi_{1}} \bar{I}_{\tau_{1}}$ to obtain a $\left(\mathcal{O}^{(1)}, \mathcal{O}_{2}^{(1)}\right)$-ideal $J_{1} \sim I_{1}$ of norm $D=2^{e}$.
vi. Constructs the isogeny $\eta_{1}: E_{A}^{(1)} \rightarrow E_{2}^{(1)}$ of degree $D$ corresponding to the ideal $J_{1}$ such that $\hat{\varphi}_{1} \circ \eta_{1}: E_{A}^{(1)} \rightarrow E_{1}^{(1)}$ is cyclic and sets cert ${ }_{i d}=\left(E_{1}^{(1)}, \eta_{1}\right)$. vii. Issues the user secret key usk ${ }_{\text {id }}=\left(\tau_{2}\right.$, cert $\left._{\text {id }}=\left(E_{1}^{(1)}, \eta_{1}\right)\right)$.

SQIIBS. $\operatorname{Sign}\left(\mathrm{pp}_{\mathrm{ibs}}\right.$, usk $\left.\mathrm{id}_{\mathrm{id}}, m\right) \rightarrow \sigma$ : On input $\mathrm{pp}_{\mathrm{ibs}}=\left(p, E_{0}, D_{c}, D, \mathcal{H}_{1}, \Phi_{D_{c}}, E_{A}^{(1)}\right)$, user secret key usk ${ }_{\text {id }}=\left(\tau_{2}\right.$, cert $\left._{\text {id }}\right)$ and a message $m \in\{0,1\}^{*}$, the signer generates a signature $\sigma$ on $m$ as follows:
i. Picks a random commitment isogeny $\psi_{2}: E_{0} \rightarrow E_{1}^{(2)}$.
ii. Computes $s_{2}=\mathcal{H}_{1}\left(j\left(E_{1}^{(2)}\right), m\right)$ and sets the challenge isogeny $\Phi_{D_{c}}\left(E_{1}^{(2)}, s_{2}\right)=$ $\varphi_{2}$ where $\varphi_{2}: E_{1}^{(2)} \rightarrow E_{2}^{(2)}$ is a non-backtracking isogeny of degree $D_{c}$.
iii. Computes the ideal $\bar{I}_{\tau_{2}}, I_{\tau_{2}}, I_{\psi_{2}}$ and $I_{\varphi_{2}}$ corresponding to the isogenies $\hat{\tau_{2}}$, $\tau_{2}, \psi_{2}$ and $\varphi_{2}$ respectively.
iv. The signer having the knowledge of $\mathcal{O}^{(2)}=\operatorname{End}\left(E_{A}^{(2)}\right)$ through $\tau_{2}$ and $\mathcal{O}_{2}^{(2)}=$ $\operatorname{End}\left(E_{2}^{(2)}\right)$ through $\varphi_{2} \circ \psi_{2}$, executes the algorithm $\operatorname{SigningKLPT}_{2^{e}}\left(I_{\tau_{2}}, I_{2}\right)$ described in Section 2.3 on input the $\left(\mathcal{O}_{0}, \mathcal{O}^{(2)}\right)$-ideal $I_{\tau_{2}}$ and a left $\mathcal{O}^{(2)}$-ideal $I_{2}=I_{\varphi_{2}} I_{\psi_{2}} \bar{I}_{\tau_{2}}$ to obtain a $\left(\mathcal{O}^{(2)}, \mathcal{O}_{2}^{(2)}\right)$-ideal $J_{2} \sim I_{2}$ of norm $D$.
v. Constructs the isogeny $\eta_{2}: E_{A}^{(2)} \rightarrow E_{2}^{(2)}$ of degree $D$ corresponding to the ideal $J_{2}$ such that $\hat{\varphi}_{2} \circ \eta_{2}: E_{A}^{(2)} \rightarrow E_{1}^{2}$ is cyclic and sets $\bar{\sigma}=\left(E_{1}^{(2)}, \eta_{2}\right)$.
vi. Extracts cert $\mathrm{id}_{\mathrm{d}}$ from usk $\mathrm{id}_{\text {id }}$ and sets the signature $\sigma=\left(\bar{\sigma}, E_{A}^{(2)}\right.$, cert $\left._{\mathrm{id}}\right)$.

SQIIBS.Verify $\left(\mathrm{pp}_{\mathrm{ibs}}, \mathrm{id}, m, \sigma\right) \rightarrow$ Valid/Invalid: The verifier employing $\mathrm{pp}_{\mathrm{ibs}}=$ $\left(p, E_{0}, D_{c}, D, \mathcal{H}_{1}, \Phi_{D_{c}}, E_{A}^{(1)}\right)$ verifies the validity of signature $\sigma=\left(\bar{\sigma}, E_{A}^{(2)}\right.$, cert $\left._{\text {id }}\right)$ on $m \in\{0,1\}^{*}$ as follows:
i. Parses $\sigma=\left(\bar{\sigma}=\left(E_{1}^{(2)}, \eta_{2}\right), E_{A}^{(2)}\right.$, $\left.\operatorname{cert}_{\text {id }}=\left(E_{1}^{(1)}, \eta_{1}\right)\right)$.
ii. Computes $s_{1}=\mathcal{H}_{1}\left(j\left(E_{1}^{(1)}\right), \operatorname{bin}\left(j\left(E_{A}^{(2)}\right)\right) \|\right.$ id $)$ and $s_{2}=\mathcal{H}_{1}\left(j\left(E_{1}^{(2)}\right), m\right)$.
iii. Recovers the isogenies $\Phi_{D_{c}}\left(E_{1}^{(1)}, s_{1}\right)=\varphi_{1}$ and $\Phi_{D_{c}}\left(E_{1}^{(2)}, s_{2}\right)=\varphi_{2}$.
iv. Checks if $\eta_{1}$ is an isogeny of degree $D$ from $E_{A}^{(1)}$ to $E_{2}^{(1)}$ and that $\hat{\varphi_{1}} \circ \eta_{1}$ : $E_{A}^{(1)} \rightarrow E_{1}^{(1)}$ is cyclic.
v. Checks if $\eta_{2}$ is an isogeny of degree $D$ from $E_{A}^{(2)}$ to $E_{2}^{(2)}$ and that $\hat{\varphi_{2}} \circ \eta_{2}$ : $E_{A}^{(2)} \rightarrow E_{1}^{(2)}$ is cyclic.
vi. If all the checks succeed returns Valid, otherwise returns Invalid.

Correctness. The correctness of our proposed scheme SQIIBS follows immediately from the correctness of SQISign signature described in Section 2.5.

### 4.1 Efficiency

A theoretical comparison of our scheme SQIIBS with the existing works on IBS from isogenies is provided in Table 1 and Table 2. We compare our scheme with the CSIDH-based IBS scheme by Peng et al. [16] as well as the recently proposed IBS scheme by Shaw et al. [17]. Table 2 depicts that the secret key size and signature size of the existing IBS scheme grows with the value of $S_{1}$. The exponential size of $S_{1}=2^{\eta_{1}-1}$ leads to large key and signatures, making them impractical for real-life applications. The user secret key in our scheme comprises of an elliptic curve over the field $\mathbb{F}_{p^{2}}$ and two isogenies of degree $2^{e}$. The elliptic curve is represented by its $j$-invariant and thus it is of size $2 \log p$. As discussed in [6], an isogeny of degree $2^{e}$ can be compressed to $e$ bits where $e=\frac{15}{4} \log p$. Thus the user secret key is of size $2 \log p+2\left(\frac{15}{4}\right) \log p=2 \log p+\frac{15}{2} \log p$. The signature in our scheme comprises of three elliptic curves over $\mathbb{F}_{p^{2}}$ and two isogenies of degree $2^{e}$. Thus, the signature in our scheme is of size $3(2 \log p)+2\left(\frac{15}{4}\right) \log p=$ $6 \log p+\frac{15}{2} \log p$. Our scheme enjoys improved efficiency in terms of key and signature sizes which thereby reduces the storage and communication cost.

Table 1. Comparison of our SQIIBS with existing IBS schemes

| Scheme | Security Analysis | Rejection Sampling | Security |
| :---: | :---: | :---: | :---: |
| Peng et al.'s IBS[16] | $\boldsymbol{x}$ | $\boldsymbol{\jmath}$ | CSIDH |
| Shaw et al.'s IBS [17] | $\boldsymbol{\checkmark}$ | $\boldsymbol{x}$ | CSI-FiSh |
| Our Work | $\boldsymbol{\checkmark}$ | $\boldsymbol{x}$ | SQISign |

CSIDH = Commutative Supersingular Isogeny Diffie-Hellman, CSI-FiSh = Commutative Supersingular Isogeny based Fiat-Shamir signature, SQISign $=$ Short Quaternion and Isogeny Signature.

Table 2. Comparison of secret and signature size of our SQIIBS with existing IBS schemes from isogenies

| Scheme | $\mid$ usk ${ }_{\text {id }} \mid$ | $\|\sigma\|$ |
| :---: | :---: | :---: |
| Peng et al.'s IBS[16] | $n T_{1} S_{1} \log \left(2 I_{1}+1\right)+T_{1} S_{1} \log p$ | $T_{1} T_{2}\left[n \log \left(2 I_{2}+1\right)+\log S_{1}\right]+T_{1} S_{1} \log p$ |
| Shaw et al.'s IBS [17] | $T_{1} S_{1}\left[\log S_{0}+\log N\right]$ | $T_{1} T_{2}\left[\log N+\log S_{1}\right]+T_{1} S_{1} \log p$ |
| Our Work | $2 \log p+\frac{15}{2} \log p$ | $6 \log p+\frac{15}{2} \log p$ |

Here $n \in \mathbb{N}, p$ is a prime, $I_{0}, I_{1}=\delta_{0} I_{0}, I_{2}=\delta_{1} I_{1}, T_{1}, T_{2}, S_{0}=2^{\eta_{0}}-1$ and $S_{1}=2^{\eta_{1}}-1$ are integers with $T_{1}<S_{0}$ and $T_{2}<S_{1} . N$ is the size of ideal class group for CSIDH- 512 parameter set.

### 4.2 Security Analysis

Theorem 4.21. Our proposed scheme SQIIBS is UF-IBS-CMA secure as the underlying signature scheme SQISign is UF-CMA secure.

Proof. Let us assume that there exists an adversary $\mathcal{A}$ that wins the UF-IBS-CMA game with non-negligible probability. At the end of the game, $\mathcal{A}$ outputs a valid forgery $\left(m^{*}, \mathrm{id}^{*}, \sigma^{*}\right)$ where $\sigma^{*}=\left(\bar{\sigma}^{*},\left(E_{A}^{(2)}\right)^{*}\right.$, cert $\left._{\text {id }}{ }^{*}\right)$. We employ the adversary $\mathcal{A}$ as a subroutine to design an adversary $\mathcal{B}$ that breaks the UF-CMA security of the signature scheme SQISign. To complete the security reduction, $\mathcal{B}$ simulating the IBS security game with $\mathcal{A}$ must embed the public key given to $\mathcal{B}$ by its UF-CMA challenger $\mathcal{C}$ into some part of the "target" which $\mathcal{A}$ takes as a target of forgery.

There are two attack points in our construction. The adversary $\mathcal{A}$ may either take the public parameter $\mathrm{pp}_{\mathrm{ibs}}$ provided by $\mathcal{B}$ or it reuses the components cert $\mathrm{id}^{*}$ and $\left(E_{A}^{(2)}\right)^{*}$ of the answer of the signing oracle on id ${ }^{*}$ and message $m \neq m^{*}$ for its forgery. We denote the later event as "REUSE". Then the advantage of $\mathcal{A}$ is given by $\operatorname{Pr}[$ Success $]=\operatorname{Pr}[$ Success $\mid \neg$ REUSE $]+\operatorname{Pr}[$ Success $\mid$ REUSE $]$ where Success is the event that $\mathcal{A}$ wins in $\operatorname{Exp}_{\text {IBS }, \mathcal{A}}^{\mathrm{UF}-\operatorname{ABS}}$-CMA $(\lambda)$. For each of the two cases $\neg$ REUSE and REUSE, we give reductions as follows:

Case $1 \operatorname{Pr}[$ Success $\mid \neg$ REUSE]: We describe below how the UF-CMA adversary $\mathcal{B}$ plays the role of the challenger and simulates the experiment $\operatorname{Exp} \operatorname{UF-IBS}, \mathcal{A}-C M A(\lambda)$.

Setup: The UF-CMA challenger $\mathcal{C}$ generates the public parameter $\mathrm{pp}_{\mathrm{sgn}}=$ $\left(p, E_{0}, D_{c}, D, \mathcal{H}_{1}, \Phi_{D_{c}}\right)$ by executing the algorithm SQISign.Setup $\left(1^{\lambda}\right)$ and computes a secret-public key pair (sk, pk) $\leftarrow$ SQISign. KeyGen $\left(\mathrm{pp}_{\mathrm{sgn}}\right)$ where $\mathrm{sk}=\tau_{1}$ and $\mathrm{pk}=E_{A}^{(1)}$ and forwards $\mathrm{pp}_{\mathrm{sgn}}$ and pk to the adversary $\mathcal{B}$. It keeps sk secret to itself. The challenger $\mathcal{C}$ also maintains a list SList and initializes SList to $\emptyset$. Upon receiving $\mathrm{pp}_{\mathrm{sgn}}=\left(p, E_{0}, D_{c}, D, \mathcal{H}_{1}, \Phi_{D_{c}}\right)$ and $\mathrm{pk}=E_{A}^{(1)}$ from $\mathcal{C}, \mathcal{B}$ sets $\mathrm{pp}_{\mathrm{ibs}}=\left(p, E_{0}, D_{c}, D, \mathcal{H}_{1}, \Phi_{D_{c}}, E_{A}^{(1)}\right)$ and sends it to $\mathcal{A}$. It also initializes the lists Klist, Clist, Mlist to $\emptyset$.
Query Phase: The adversary $\mathcal{B}$ responds to polynomially many adaptive queries made by $\mathcal{A}$ to the oracles $\mathcal{O}_{\text {Extract }}$ and $\mathcal{O}_{\text {sign }}$ as follows:

- Oracle $\mathcal{O}_{\text {Extract }}(\cdot)$ : On receiving a query on a user identity id from $\mathcal{A}, \mathcal{B}$ checks whether (id, usk ${ }_{i d}$ ) $\in$ Kist. If there exists such a pair in Klist, it returns usk $_{\text {id }}$ to $\mathcal{A}$ and appends id to CList. If (id, usk $\mathrm{k}_{\mathrm{id}}$ ) $\notin \mathrm{Kist}, \mathcal{B}$ picks a random isogeny $\tau_{2}: E_{0} \rightarrow E_{A}^{(2)}$ and queries its signing oracle $\mathcal{O}_{\text {Sign }}\left(\mathrm{sk}=\tau_{1}, \cdot\right)$ simulated by $\mathcal{C}$ on the message $\operatorname{bin}\left(j\left(E_{A}^{(2)}\right)\right) \|$ id. Upon receiving the signature $\operatorname{cert}_{\text {id }}=\left(E_{1}^{(1)}, \eta_{1}\right)$ from $\mathcal{C}$, the adversary $\mathcal{B}$ sets usk ${ }_{\text {id }}=\left(\tau_{2}\right.$, cert $\left._{\text {id }}\right)$ and returns it to $\mathcal{A}$. The adversary $\mathcal{B}$ also appends (id, usk ${ }_{\text {id }}$ ) to Klist and id to Clist. The challenger $\mathcal{C}$ appends $\operatorname{bin}\left(j\left(E_{A}^{(2)}\right)\right) \|$ id in Slist.
- Oracle $\mathcal{O}_{\mathrm{Sign}}(\cdot)$ : On receiving a query on a message $m \in\{0,1\}^{*}$ and a user identity id from $\mathcal{A}, \mathcal{B}$ retrieves the pair (id, usk ${ }_{i d}$ ) from Klist where usk $\mathrm{k}_{\mathrm{id}}=$ $\left(\tau_{2}\right.$, cert $_{\mathrm{id}}$ ) is the user secret key corresponding to id. If (id, usk $\mathrm{id}_{\mathrm{id}}$ ) $\notin$ Kist, $\mathcal{B}$ picks a random isogeny $\tau_{2}: E_{0} \rightarrow E_{A}^{(2)}$ and queries its signing oracle $\mathcal{O}_{\mathrm{S}}\left(\mathrm{sk}=\tau_{1}, \cdot\right)$ on the message $\operatorname{bin}\left(j\left(E_{A}^{(2)}\right)\right) \|$ id. Upon receiving the signature cert $_{\text {id }}=\left(E_{1}^{(1)}, \eta_{1}\right)$ under sk $=\tau_{1}$ from $\mathcal{C}, \mathcal{B}$ sets usk ${ }_{\text {id }}=\left(\tau_{2}\right.$, cert $\left._{\text {id }}\right)$. It then executes $\bar{\sigma}=\left(E_{1}^{(2)}, \eta_{2}\right) \leftarrow$ SQISign.Sign $\left(\mathrm{pp}_{\mathrm{sgn}}, \tau_{2}, m\right)$, sets the signature $\sigma=$ $\left(\bar{\sigma}, E_{A}^{(2)}\right.$, cert $\left._{\text {id }}\right)$ and sends it to $\mathcal{A}$. It also appends ( $m$, id, $\sigma$ ) to Mlist.
Forgery: The adversary $\mathcal{A}$ eventually outputs a message $m^{*}$, user identity $\mathrm{id}^{*}$ and a forge signature $\sigma^{*}$ where $\sigma^{*}=\left(\bar{\sigma}^{*},\left(E_{A}^{(2)}\right)^{*}\right.$, cert $\left._{\mathrm{id}}{ }^{*}\right)$. If $\mathcal{A}$ wins the UF-IBS-CMA game with non-negligible probability then ( $m^{*}$, $\mathrm{id}^{*}, \sigma^{*}$ ) must be a valid forgery. Thus, IBS.Verify $\left(\mathrm{pp}_{\mathrm{ibs}}, \mathrm{id}^{*}, m^{*}, \sigma^{*}\right) \rightarrow$ Valid where $\mathrm{id}{ }^{*} \notin$ Clist and $\left(m^{*}, \mathrm{id}^{*}, \cdot\right) \notin$ Mlist. The adversary $\mathcal{B}$ submits $\operatorname{bin}\left(j\left(\left(E_{A}^{(2)}\right)^{*}\right)\right) \| \mathrm{id}^{*}$, cert $_{\mathrm{id}}{ }^{*}$ as a forgery to its own challenger $\mathcal{C}$.

The event $\neg$ REUSE means ( id $^{*}$, usk $_{\mathrm{id}^{*}}$ ) $\notin$ Klist where $^{\text {usk }}{ }_{\mathrm{id}}{ }^{*}=\left(\tau_{2}^{*}\right.$, cert $\left._{\mathrm{id}^{*}}\right)$. This implies that $\left(\operatorname{bin}\left(j\left(\left(E_{A}^{(2)}\right)^{*}\right)\right) \|\right.$ id $\left.\mathrm{d}^{*}\right) \notin$ Slist. Hence, the adversary $\mathcal{B}$ has output the valid forgery $\left(\operatorname{bin}\left(j\left(\left(E_{A}^{(2)}\right)^{*}\right)\right) \|\right.$ id $^{*}$, certid $\left.{ }_{\text {id }}^{*}\right)$ such that SQISign.Veriy $\left(\mathrm{pp}_{\mathrm{sgn}}\right.$, $E_{A}^{(1)}, \operatorname{bin}\left(j\left(\left(E_{A}^{(2)}\right)^{*}\right)\right) \|$ id*, cert id $) \rightarrow$ Valid. From the security of SQISign, it follows that $\operatorname{Pr}[$ Success $\mid \neg$ REUSE $]$ is negligible.

Case $2 \operatorname{Pr}[$ Success $\mid$ REUSE]: In this case the adversary $\mathcal{A}$ reuses the components cert $_{\mathrm{id}}{ }^{*}$ and $\left(E_{A}^{(2)}\right)^{*}$ of the answer of the signing oracle query on identity id* ${ }^{*}$ and message $m \neq m^{*}$ for its forgery.

Setup: The UF-CMA challenger $\mathcal{C}$ generates the public parameter $\mathrm{pp}_{\mathrm{sgn}}=$ $\left(p, E_{0}, D_{c}, D, \mathcal{H}_{1}, \Phi_{D_{c}}\right)$ by executing the algorithm SQISign.Setup( $1^{\lambda}$ ) as in Case 1 and computes a secret-public key pair (sk, pk) $\leftarrow$ SQISign. KeyGen $\left(\mathrm{pp}_{\mathrm{sgn}}\right)$ where $\mathrm{sk}=\tau_{2}$ and $\mathrm{pk}=E_{A}^{(2)}$ and forwards $\mathrm{pp}_{\mathrm{sgn}}$ and pk to the adversary $\mathcal{B}$. It keeps sk secret to itself. The challenger $\mathcal{C}$ maintains a list SList and initializes SList to $\emptyset$. Upon receiving $\mathrm{pp}_{\mathrm{sgn}}=\left(p, E_{0}, D_{c}, D, \mathcal{H}_{1}, \Phi_{D_{c}}\right)$ and $\mathrm{pk}=E_{A}^{(2)}$ from the challenger $\mathcal{C}$, the adversary $\mathcal{B}$ picks a random isogeny $\tau_{1}: E_{0} \rightarrow E_{A}^{(1)}$, sets $\mathrm{pp}_{\mathrm{ibs}}=\left(p, E_{0}, D_{c}, D, \mathcal{H}_{1}, \Phi_{D_{c}}, E_{A}^{(1)}\right)$, msk $=\tau_{1}$ and sends $\mathrm{pp}_{\mathrm{ibs}}$ to $\mathcal{A}$. It initializes the lists Klist, Clist, Mlist to $\emptyset$ and chooses $r \leftarrow\{1,2, \ldots, q(\lambda)\}$ where $q(\lambda)$ is the maximum number of queries by $\mathcal{A}$.
Query Phase: The adversary $\mathcal{B}$ responds to polynomially many adaptive queries to the oracles $\mathcal{O}_{\text {Extract }}$ and $\mathcal{O}_{\text {sign }}$ made by $\mathcal{A}$. Let id' be the identity for which the $r^{\text {th }}$ signing query of $\mathcal{A}$ was made.

- Oracle $\mathcal{O}_{\text {Extract }}(\cdot):$ If $\mathcal{A}$ ever makes an extract query for the identity id', the experiment is aborted. On receiving a query on a user identity id $\neq \mathrm{id}^{\prime}$ from $\mathcal{A}, \mathcal{B}$ checks whether (id, usk $\mathrm{id}_{\text {}}$ ) $\in$ Kist. If there exists such a pair in Klist, it returns usk ${ }_{\text {id }}$ and appends id to CList. If (id, usk $\mathrm{id}_{\mathrm{id}}$ ) $\notin$ Kist, it picks a random isogeny $\bar{\tau}_{2}: E_{0} \rightarrow \bar{E}_{A}^{(2)}$ and uses msk $=\tau_{1}$ to compute $\overline{\operatorname{cert}}_{\mathrm{id}}=\left(\bar{E}_{1}^{(1)}, \bar{\eta}_{1}\right)$ $\leftarrow$ SQISign. $\operatorname{Sign}\left(\mathrm{pp}_{\mathrm{sgn}}, \tau_{1}, \operatorname{bin}\left(j\left(\bar{E}_{A}^{(2)}\right)\right) \|\right.$ id $)$. It then sets $\overline{u s k}_{\mathrm{id}}=\left(\bar{\tau}_{2}, \overline{\operatorname{cert}}_{\mathrm{id}}\right)$ and returns it to $\mathcal{A}$. It appends (id, $\overline{u s k}_{\mathrm{id}}$ ) to Klist and id to Clist.
- Oracle $\mathcal{O}_{\text {Sign }}(\cdot)$ : The adversary $\mathcal{B}$ receives signing queries on pairs ( $m$, id) from the adversary $\mathcal{A}$. For the $r^{\text {th }}$ signing query on ( $\mathrm{id}^{\prime}, m$ ) by $\mathcal{A}, \mathcal{B}$ first checks whether $\left(m, \mathrm{id}^{\prime}, \sigma\right) \in$ Mlist. If there exists such a tuple, the adversary $\mathcal{B}$ aborts the experiment. Otherwise, $\mathcal{B}$ computes cert $\mathrm{id}^{\prime}=\left(\left(E_{1}^{(1)}\right)^{\prime}, \eta_{1}^{\prime}\right) \leftarrow$ SQISign. $\operatorname{Sign}\left(\mathrm{pp}_{\mathrm{sgn}}, \tau_{1}, \operatorname{bin}\left(j\left(E_{A}^{(2)}\right)\right) \|\right.$ id' $)$ using msk $=\tau_{1}$ and queries its signing oracle $\mathcal{O}_{\mathbf{S}}\left(\mathrm{sk}=\tau_{2}, \cdot\right)$ on $m$. Upon receiving the signature $\bar{\sigma}=\left(E_{1}^{(2)}, \eta_{2}\right)$ on $m$ from $\mathcal{C}$ under secret key sk $=\tau_{2}, \mathcal{B}$ sets $\sigma^{\prime}=\left(\bar{\sigma}, E_{A}^{(2)}\right.$, cert id $\left.^{\prime}\right)$ and sends it to $\mathcal{A}$. The adversary $\mathcal{B}$ updates the Mlist with $\left(m, \mathrm{id}^{\prime}, \sigma^{\prime}\right)$ and the challenger $\mathcal{C}$ updates Slist with $m$. For the $i^{\text {th }}$ query where $i \in\{r+1, \ldots, q(\lambda)\}$, on identity id' and a message $m^{\prime}$ by $\mathcal{A}$, the adversary $\mathcal{B}$ checks whether $\left(m^{\prime}, \mathrm{id}^{\prime}, \sigma^{\prime}\right) \in \mathrm{Mlist}$. If such a tuple exists, $\mathcal{B}$ answers the query from the Mlist, otherwise it proceeds as in the $r^{\text {th }}$ signing query.

Upon receiving a query on a message $m$ and identity id $\neq \mathrm{id}^{\prime}, \mathcal{B}$ retrieves the pair (id,usk ${ }_{\text {id }}$ ) from Klist where usk $_{\text {id }}=\left(\bar{\tau}_{2}, \overline{\mathrm{cert}}_{\mathrm{id}}\right)$ is the user secret key corresponding to id. If (id, uskid) $\notin$ Kist, it picks a random isogeny $\bar{\tau}_{2}: E_{0} \rightarrow$ $\bar{E}_{A}^{(2)}$ and uses its master secret key msk $=\tau_{1}$ to compute $\overline{\operatorname{cert}}_{\mathrm{id}}=\left(\bar{E}_{1}^{(1)}, \bar{\eta}_{1}\right)$ $\leftarrow \operatorname{SQISign} . \operatorname{Sign}\left(\mathrm{pp}_{\mathrm{sgn}}, \tau_{1}, \operatorname{bin}\left(j\left(\bar{E}_{A}^{(2)}\right)\right) \|\right.$ id $)$ and sets $\overline{u s k}_{\mathrm{id}}=\left(\bar{\tau}_{2}, \overline{\operatorname{cert}}_{\mathrm{id}}\right)$. It then computes the signature $\bar{\sigma}=\left(\bar{E}_{1}^{(2)}, \bar{\eta}_{2}\right) \leftarrow$ SQISign.Sign $\left(\mathrm{pp}_{\mathrm{ibs}}, \bar{\tau}_{2}, m\right)$ on $m$ and sets $\sigma=\left(\bar{\sigma}, \bar{E}_{A}^{(2)},{\overline{\text { cert }}{ }_{\mathrm{id}}}\right)$ and sends it to $\mathcal{A}$. It appends ( $m$, id, $\sigma$ ) to Mlist.
Forgery: If $\mathcal{A}$ eventually outputs a message $m^{*}$, user identity $\mathrm{id}^{*}$ and a forge signature $\sigma^{*}$ where $\sigma^{*}=\left(\bar{\sigma}^{*},\left(E_{A}^{(2)}\right)^{*}\right.$, cert $\left._{\mathrm{id}^{*}}\right)$ and the experiment was never aborted, $\mathcal{B}$ submits ( $m^{*}, \sigma^{*}$ ) as a forgery to its own challenger $\mathcal{C}$. If $\mathcal{A}$ wins the UF-IBS-CMA game with non-negligible probability then ( $m^{*}$, $\mathrm{id}{ }^{*}, \sigma^{*}$ ) must be a valid forgery. Thus, we have IBS.Verify $\left(\mathrm{pp}_{\mathrm{ibs}}, \mathrm{id}^{*}, m^{*}, \sigma^{*}\right)=$ Valid, $\mathrm{id}^{*} \notin$ Clist and ( $\left.m^{*}, \mathrm{id}^{*}, \cdot\right) \notin$ Mlist. Note that the condition $\left(m^{*}, \mathrm{id}^{*}, \cdot\right) \notin$ Mlist means that the adversary $\mathcal{B}$ never queried its signing oracle $\mathcal{O}_{\mathrm{S}}\left(\tau_{2}, \cdot\right)$ on $m^{*}$. With probability at least $1 / q(\lambda)$, the experiment is not aborted and $\mathrm{id}^{\prime}=$ $\mathrm{id}^{*}$. The success probability of $\mathcal{B}$ in forging a signature for SQISign is thus at least $\operatorname{Pr}[$ Success $\mid$ REUSE $] / q(\lambda)$. From the security of SQISign, it follows that this quantity must be negligible. Since $q$ is polynomial in $\lambda$, we must have $\operatorname{Pr}[$ Success $\mid$ REUSE $]$ is negligible as well.

## 5 Puncturable Signature : Concrete Construction

We now describe our Short Quaternion and Isogeny Puncturable Signature (SQIPS) leveraging our scheme SQIIBS described in Section 4. Let $\mathcal{M}=\{0,1\}^{*}$ denotes the message space and $\mathcal{P}=\{0,1\}^{l} \subseteq \mathcal{M}$ be the prefix space of our PS.

SQIPS.Setup $\left(1^{\lambda}\right) \rightarrow\left(\mathrm{pp}_{\mathrm{ps}}, \mathrm{sk}_{0}\right)$ : On input $1^{\lambda}$, the signer executes this algorithm to generate the public parameter $\mathrm{pk}_{\mathrm{ps}}$ and initial secret key sk as follows:
i. Invokes the algorithm SQIIBS.Setup $\left(1^{\lambda}\right)$ to compute the key pair $\left(\mathrm{pp}_{\mathrm{ibs}}, \mathrm{msk}\right)$ as follows:

- Chooses a prime $p$ and fixes the supersingular curve $E_{0}: y^{2}=x^{3}+x$ over $\mathbb{F}_{p^{2}}$ with special extremal endomorphism ring $\mathcal{O}_{0}=\left\langle 1, i, \frac{i+j}{2}, \frac{1+k}{2}\right\rangle$.
- Picks a smooth number $D=2^{e}$ where $2^{e}>p^{3}$.
- Picks an odd smooth number $D_{c}=\ell^{e}$ where $\ell$ is a prime and computes $\mu\left(D_{c}\right)=(\ell+1) \cdot \ell^{e-1}$.
- Samples a cryptographic hash function $\mathcal{H}_{1}: \mathbb{F}_{p^{2}} \times\{0,1\}^{*} \rightarrow\left[1, \mu\left(D_{c}\right)\right]$.
- Samples an arbitrary function $\Phi_{D_{c}}(E, s)$ that maps a pair $(E, s)$ of an elliptic curve $E$ and an integer $s \in\left[1, \mu\left(D_{c}\right)\right]$ to a non-backtracking isogeny of degree $D_{c}$ from $E[3]$.
- Picks a random isogeny $\tau_{1}: E_{0} \rightarrow E_{A}^{(1)}$.
- Sets $\mathrm{pp}_{\mathrm{ibs}}=\left(p, E_{0}, D_{c}, D, \mathcal{H}_{1}, \Phi_{D_{c}}, E_{A}^{(1)}\right)$ and msk $=\tau_{1}$.
ii. For each prefix $x^{\prime} \in\{0,1\}^{l}$, executes the algorithm SQIIBS.Extract(pp ibs , msk $=$ $\left.\tau_{1}, x^{\prime}\right)$ to compute the key usk ${ }_{x^{\prime}}$ and stores it in an array $T$ of size $2^{l}$.
- Picks a random isogeny $\tau_{2}: E_{0} \rightarrow E_{A}^{(2)}$.
- Selects a random commitment isogeny $\psi_{1}: E_{0} \rightarrow E_{1}^{(1)}$.
- Computes $s_{1}=\mathcal{H}_{1}\left(j\left(E_{1}^{(1)}\right), \operatorname{bin}\left(j\left(E_{A}^{(2)}\right)\right) \| x^{\prime}\right)$ and sets the challenge isogeny $\Phi_{D_{c}}\left(E_{1}^{(1)}, s_{1}\right)=\varphi_{1}$ where $\varphi_{1}: E_{1}^{(1)} \rightarrow E_{2}^{(1)}$ is a non-backtracking isogeny of degree $D_{c}$.
- Computes the ideals $\bar{I}_{\tau_{1}}, I_{\tau_{1}}, I_{\psi_{1}}$ and $I_{\varphi_{1}}$ corresponding to the isogenies $\hat{\tau_{1}}, \tau_{1}, \psi_{1}$ and $\varphi_{1}$ respectively.
- The signer having the knowledge of $\mathcal{O}^{(1)}=\operatorname{End}\left(E_{A}^{(1)}\right)$ through $\tau_{1}$ and $\mathcal{O}_{2}^{(1)}=\operatorname{End}\left(E_{2}^{(1)}\right)$ through $\varphi_{1} \circ \psi_{1}$, runs the SigningKLPT $2^{e}\left(I_{\tau_{1}}, I_{1}\right)$ algorithm (Section 2.3) on input the $\left(\mathcal{O}_{0}, \mathcal{O}^{(1)}\right)$-ideal $I_{\tau_{1}}$ and a left $\mathcal{O}^{(1)}$-ideal $I_{1}=I_{\varphi_{1}} I_{\psi_{1}} \bar{I}_{\tau_{1}}$ to obtain a $\left(\mathcal{O}^{(1)}, \mathcal{O}_{2}^{(1)}\right)$-ideal $J_{1} \sim I_{1}$ of norm $D=2^{e}$.
- Constructs the isogeny $\eta_{1}: E_{A}^{(1)} \rightarrow E_{2}^{(1)}$ of degree $D$ corresponding to the ideal $J_{1}$ such that $\hat{\varphi}_{1} \circ \eta_{1}$ is cyclic and $\operatorname{cert}_{x^{\prime}}=\left(E_{1}^{(1)}, \eta_{1}\right)$.
- Issues the user secret key usk ${ }_{x^{\prime}}=\left(\tau_{2}, \operatorname{cert}_{x^{\prime}}=\left(E_{1}^{(1)}, \eta_{1}\right)\right)$.
iii. Sets $T\left[\right.$ ind $\left._{x^{\prime}}\right]=$ usk $_{x^{\prime}}$ where ind $_{x^{\prime}}=\left(x^{\prime}\right)_{10} \in\left\{0,1, \ldots, 2^{l}-1\right\}$ is the decimal representation of the binary string $x^{\prime}$.
iv. Sets the public parameter $\mathrm{pp}_{\mathrm{ps}}=\mathrm{pp}_{\mathrm{ibs}}$ and secret key $\mathrm{sk}=T$.

SQIPS.Puncture(sk, $x^{\prime}$ ) $\rightarrow \mathbf{s k}^{\prime}$ : The signer on input the secret key sk $=T$ and a prefix $x^{\prime} \in\{0,1\}^{l}$, computes $\operatorname{ind}_{x^{\prime}}=\left(x^{\prime}\right)_{10}$ and sets $T[$ ind $]=0$. It returns the updated punctured secret key $\mathrm{sk}^{\prime}=T$ where the value corresponding to the index ind of the array $T$ is made 0 .
SQIPS. $\operatorname{Sign}\left(\mathrm{pp}_{\mathrm{ps}}, \mathrm{sk}, m\right) \rightarrow \Sigma / \perp$ : Taking input $\mathrm{pp}_{\mathrm{ps}}=\left(p, E_{0}, D_{c}, D, \mathcal{H}_{1}, \Phi_{D_{c}}, E_{A}^{(1)}\right)$, secret key sk $=T$ and a message $m \in\{0,1\}^{*}$, the signer either generates a signature $\Sigma$ if the prefix $x^{\prime}$ of $m$ has not been punctured or it returns $\perp$.
i. Returns $\perp$ if $T\left[\right.$ ind $\left._{x^{\prime}}\right]=0$.
ii. If $T\left[\right.$ ind $\left._{x^{\prime}}\right] \neq 0$, it retrieves the value usk ${ }_{x^{\prime}}=\left(\tau_{2}, \operatorname{cert}_{x^{\prime}}=\left(E_{1}^{(1)}, \eta_{1}\right)\right)=$ $T\left[\right.$ ind $\left._{x^{\prime}}\right]$ from the array and executes the algorithm SQIIBS.Sign $\left(\mathrm{pp}_{\mathrm{ibs}}\right.$, usk $\left._{x^{\prime}}, m\right)$ as follows to generate a signature on $m$.

- Picks a random commitment isogeny $\psi_{2}: E_{0} \rightarrow E_{1}^{(2)}$.
- Computes $s_{2}=\mathcal{H}_{1}\left(j\left(E_{1}^{(2)}\right), m\right)$ and $\Phi_{D_{c}}\left(E_{1}^{(2)}, s_{2}\right)=\varphi_{2}$ where $\varphi_{2}$ : $E_{1}^{(2)} \rightarrow E_{2}^{(2)}$ is a non-backtracking challenge isogeny of degree $D_{c}$.
- Computes the ideal $\bar{I}_{\tau_{2}}, I_{\tau_{2}}, I_{\psi_{2}}$ and $I_{\varphi_{2}}$ corresponding to the isogenies $\hat{\tau_{2}}, \tau_{2}, \psi_{2}$ and $\varphi_{2}$ respectively.
- The signer having the knowledge of $\mathcal{O}^{(2)}=\operatorname{End}\left(E_{A}^{(2)}\right)$ through $\tau_{2}$ and $\mathcal{O}_{2}^{(2)}=\operatorname{End}\left(E_{2}^{(2)}\right)$ through $\varphi_{2} \circ \psi_{2}$, runs the SigningKLPT $2^{e}\left(I_{\tau_{2}}, I_{2}\right)$ algorithm (Section 2.3) on input the $\left(\mathcal{O}_{0}, \mathcal{O}^{(2)}\right)$-ideal $I_{\tau_{2}}$ and a left $\mathcal{O}^{(2)}$-ideal $I_{2}=I_{\varphi_{2}} I_{\psi_{2}} \bar{I}_{\tau_{2}}$ to obtain a $\left(\mathcal{O}^{(2)}, \mathcal{O}_{2}^{(2)}\right)$-ideal $J_{2} \sim I_{2}$ of norm $D=2^{e}$.
- Constructs the isogeny $\eta_{2}: E_{A}^{(2)} \rightarrow E_{2}^{(2)}$ of degree $D$ corresponding to the ideal $J_{2}$ such that $\hat{\varphi_{2}} \circ \eta_{2}: E_{A}^{(2)} \rightarrow E_{1}^{(2)}$ is cyclic. It sets $\bar{\sigma}=\left(E_{1}^{(2)}, \eta_{2}\right)$.
- Extract cert ${ }_{x^{\prime}}$ from usk $x_{x^{\prime}}$ and sets the signature $\sigma=\left(\bar{\sigma}, E_{A}^{(2)}\right.$, cert $\left._{x^{\prime}}\right)$.
iii. Returns the puncturable signature $\Sigma=\sigma$.

SQIPS.Verify $\left(\mathrm{pp}_{\mathrm{ps}}, m, \Sigma\right) \rightarrow$ Valid/Invalid: This algorithm takes as input $\mathrm{pp}_{\mathrm{ps}}=$ $\left(p, E_{0}, D_{c}, D, \mathcal{H}_{1}, \Phi_{D_{c}}, E_{A}^{(1)}\right)$, a message $m \in\{0,1\}^{*}$ and a signature $\Sigma=\sigma=$ $\left(\bar{\sigma}, E_{A}^{(2)}, \operatorname{cert}_{x^{\prime}}\right)$ where $x^{\prime} \in\{0,1\}^{l}$ is the prefix of the message $m \in\{0,1\}^{*}$. It outputs Valid if $\Sigma$ is a valid signature on $m$ and Invalid otherwise.
i. Executes the algorithm SQIIBS.Verify as follows to check the validity of the signature $\Sigma=\sigma=\left(\bar{\sigma}, E_{A}^{(2)}, \operatorname{cert}_{x^{\prime}}\right)$ on $m$.

- Parses $\sigma=\left(\bar{\sigma}=\left(E_{1}^{(2)}, \eta_{2}\right), E_{A}^{(2)}, \operatorname{cert}_{x^{\prime}}=\left(E_{1}^{(1)}, \eta_{1}\right)\right)$.
- Computes $s_{1}=\mathcal{H}_{1}\left(j\left(E_{1}^{(1)}\right), \operatorname{bin}\left(j\left(E_{A}^{(2)}\right)\right) \| x^{\prime}\right)$ and $s_{2}=\mathcal{H}_{1}\left(j\left(E_{1}^{(2)}\right), m\right)$.
- Recovers the isogenies $\Phi_{D_{c}}\left(E_{1}^{(1)}, s_{1}\right)=\varphi_{1}$ and $\Phi_{D_{c}}\left(E_{1}^{(2)}, s_{2}\right)=\varphi_{2}$.
- Checks if $\eta_{1}$ is an isogeny of degree $D$ from $E_{A}^{(1)}$ to $E_{2}^{(1)}$ and that $\hat{\varphi}_{1} \circ \eta_{1}$ : $E_{A}^{(1)} \rightarrow E_{1}^{(1)}$ is cyclic.
- Checks if $\eta_{2}$ is an isogeny of degree $D$ from $E_{A}^{(2)}$ to $E_{2}^{(2)}$ and that $\hat{\varphi_{2}} \circ \eta_{2}$ : $E_{A}^{(2)} \rightarrow E_{1}^{(2)}$ is cyclic.
- If all the checks succeed returns Valid, otherwise returns Invalid.

Correctness. The correctness of our puncturable signature scheme SQIPS from isogenies follows from the correctness of our identity-based signature SQIIBS.

Theorem 5.01. Our proposed puncturable signature SQIPS is UF-CMA-AP secure as the underlying identity-based signature SQIIBS is UF-IBS-CMA secure.

Proof. Let us assume that there exists a PPT adversary $\mathcal{A}$ that wins the experiment $\operatorname{Exp}_{\mathrm{SQ} \text { IPSS }, \mathcal{A}}^{\mathrm{A}}(\lambda)$ depicted in Fig 4 with a non-negligible advantage. We design an adversary $\mathcal{B}$ who simulates the PS security experiment $\operatorname{Exp} \operatorname{UFQ}$ UCMA-AP $(\lambda)$, $\operatorname{exploits} \mathcal{A}$ as a subroutine and wins the IBS security experiment $\operatorname{Exp}_{S Q \in I I B S, \mathcal{B}}^{U G F-C M A}(\lambda)$ with the same advantage. Let $\mathcal{C}$ denotes the challenger for the security experiment $\operatorname{Exp}_{\mathrm{SQ}}^{\mathrm{UF}-\mathrm{IBSS}, \mathcal{B}} \mathrm{B}(\lambda)$.

Setup: The challenger $\mathcal{C}$ on input the security parameter $1^{\lambda}$, computes ( $\mathrm{pp}_{\mathrm{ibs}}$, msk) $\leftarrow \operatorname{SQIIBS} . \operatorname{Setup}\left(1^{\lambda}\right)$ and sends $\mathrm{pp}_{\text {ibs }}$ to $\mathcal{B}$. Additionally, $\mathcal{C}$ executes the algorithm SQIIBS.Extract $\left(\mathrm{pp}_{\mathrm{ibs}}\right.$, msk, $\left.x^{\prime}\right)$ to compute the key usk ${ }_{x^{\prime}}$ for each prefix $x^{\prime} \in\{0,1\}^{l}$ and forms the array $T\left[\right.$ ind $\left._{x^{\prime}}\right]=$ usk $_{x^{\prime}}$. Also it initiates three lists Klist, Clist and Mlist to $\emptyset$. Upon receiving the public parameter $\mathrm{pp}_{\mathrm{ibs}}$ from its own challenger $\mathcal{C}$, the adversary $\mathcal{B}$ sets $\mathrm{pp}_{\mathrm{ps}}=\mathrm{pp}_{\mathrm{ibs}}$ and forwards it to $\mathcal{A}$. It also initializes the sets $\mathcal{Q}_{\text {sig }}$ for signed messages and $\mathcal{Q}_{\text {pun }}$ for punctured prefixes to $\phi$.
Query Phase: The adversary $\mathcal{A}$ issues polynomially many adaptive queries to the following oracles $\mathcal{O}_{\text {Puncture }}(\mathrm{sk}, \cdot)$ and $\mathcal{O}_{\mathrm{Sgn}}(\mathrm{sk}, \cdot)$.

- $\mathcal{O}_{\text {Puncture }}(\mathrm{sk}=T, \cdot):$ Upon receiving a query on prefix $x^{\prime}$, the challenger $\mathcal{C}$ updates $\mathcal{Q}_{\text {pun }} \leftarrow \mathcal{Q}_{\text {pun }} \cup\left\{x^{\prime}\right\}$.
$-\mathcal{O}_{\mathrm{Sgn}}(\mathrm{sk}=T, \cdot):$ On receiving a signature query on a message $m \in$ $\{0,1\}^{*}$, the adversary $\mathcal{B}$ checks if $x^{\prime} \in \mathcal{Q}_{\text {pun }}$ where $x^{\prime}$ is the prefix of $m$. If the check succeeds, it returns $\perp$. Otherwise, it issues a signature query on ( $m, x^{\prime}$ ) for a with message $m$ and identity $x^{\prime}$ to $\mathcal{C}$. The challenger $\mathcal{C}$ extracts $T\left[\right.$ ind $\left._{x^{\prime}}\right]=$ usk $_{x^{\prime}}$ from sk $=T$, computes the signature $\Sigma \leftarrow$

SQIIBS. $\operatorname{Sign}\left(\mathrm{pp}_{\mathrm{ibs}^{\prime}}\right.$, usk $\left._{x^{\prime}}, m\right)$ and sends it to $\mathcal{B}$ who forwards it to $\mathcal{A}$. The adversary $\mathcal{B}$ updates $\mathcal{Q}_{\text {sig }} \leftarrow \mathcal{Q}_{\text {sig }} \cup\{m\}$.
Challenge: The adversary $\mathcal{A}$ sends a target prefix $x^{*} \in\{0,1\}^{l}$ to the adversary $\mathcal{B}$ which $\mathcal{B}$ forwards to $\mathcal{C}$ as the target identity. The adversary $\mathcal{A}$ can issue additional puncture and signature queries as described in the Query phase.
Corruption Query: Upon receiving a corruption query on $x^{*} \in\{0,1\}^{l}$, the adversary $\mathcal{B}$ returns $\perp$ if $x^{*} \notin \mathcal{Q}_{\text {pun }}$. Otherwise, $\mathcal{B}$ queries its extract oracle $\mathcal{O}_{\text {Extract }}(\cdot)$ for each prefix $x^{\prime} \in\{0,1\}^{l} \backslash\left\{x^{*}\right\}$ and updates the array $T$ with the response usk $x^{\prime} \leftarrow$ SQIIBS. Extract $\left(\mathrm{pp}_{\mathrm{ibs}}\right.$, msk, $\left.x^{\prime}\right)$ from $\mathcal{C}$ by setting $T\left[\right.$ ind $\left._{x^{\prime}}\right]=$ usk $_{x^{\prime}}$. For each $x^{\prime} \in \mathcal{Q}_{\text {pun }}$, the adversary $\mathcal{B}$ deletes the related key by setting $T\left[\right.$ ind $\left._{x^{\prime}}\right]=0$ and returns the current secret key sk $=T$ to $\mathcal{A}$.
Forgery: $\mathcal{A}$ eventually submits a forgery $\left(m^{*}, \Sigma^{*}, x^{*}\right)$ where $x^{*}$ is the prefix of $m^{*}$. $\mathcal{B}$ uses the forgery of $\mathcal{A}$ to frame its own forgery $\left(m^{*}, x^{*}, \Sigma^{*}\right)$.

If the adversary $\mathcal{A}$ wins the game then we have $m^{*} \notin \mathcal{Q}_{\text {sig }}, x^{*} \in \mathcal{Q}_{\text {pun }}$ and Valid $\leftarrow$ SQIPS.Verify $\left(\mathrm{pp}_{\mathrm{ps}}, m^{*}, \Sigma^{*}\right)$. The condition $m^{*} \notin \mathcal{Q}_{\text {sig }}$ means that $\left(m^{*}, x^{*}, \cdot\right) \notin$ Mlist. Also note that the adversary $\mathcal{B}$ has not made any extraction query on $x^{*}$, thus $x^{*} \notin$ Clist. Moreover, Valid $\leftarrow \operatorname{SQIPS} . V e r i f y\left(p_{p s}, m^{*}, \Sigma^{*}\right)$ implies that Valid $\leftarrow$ SQIIBS.Verify $\left(\mathrm{pp}_{\mathrm{ibs}}, m^{*}, \Sigma^{*}\right)$.

### 5.1 Comparison of our scheme SQIPS with the existing puncturable signatures

In Table 3, we compare our scheme with the existing schemes on PS. The PS scheme by Li et al. [13] is based on the $\tau$-Strong Diffie-Hellman assumption ( $\tau$-SDH) in bilinear map setting and is proven secure in the random oracle model (ROM). Their scheme employs the probabilistic bloom filter data structure and suffers from non-negligible false-positive errors. Jiang et al. [12] designed a pairing-based PS which is free from false positive errors and is secure under the hardness of the Computational Diffie-Hellman (CDH) assumption in the standard model (SDM). However, none of these schemes are resistant to quantum attacks. The PS schemes from lattices and MPKC proposed by Jiang et al. [12] enjoy post-quantum security and are based on the hardness of Short Integer Solution (SIS) and Multivariate Quadratic polynomial (MQ) assumptions respectively. Our isogeny-based PS is post-quantum secure as it is based on SQISign cryptosystem and is also free from false-positive errors.

Table 3. Comparison of the existing puncturable signature schemes

| Instantiation | Assumption | Security Model | Post-quantum | False-positive errors |
| :--- | :--- | :--- | :--- | :--- |
| Li et al. [13] | $\tau$-SDH | ROM | $\boldsymbol{x}$ | $\boldsymbol{\checkmark}$ |
| Pairing Inst. [12] | CDH | SDM | $\boldsymbol{x}$ | $\boldsymbol{x}$ |
| Lattice Inst. [12] | SIS | ROM | $\boldsymbol{\checkmark}$ | $\boldsymbol{x}$ |
| Multivariate Inst. [12] | MQ |  | $\boldsymbol{x}$ |  |
| Our Isogeny Inst. | SQISign | ROM | $\boldsymbol{\checkmark}$ | $\boldsymbol{x}$ |

## References

1. Bellare, M., Stepanovs, I., Waters, B.: New negative results on differing-inputs obfuscation. In: Advances in Cryptology-EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35. pp. 792-821. Springer (2016)
2. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: an efficient post-quantum commutative group action. In: Advances in CryptologyASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III 24. pp. 395-427. Springer (2018)
3. Charles, D.X., Lauter, K.E., Goren, E.Z.: Cryptographic hash functions from expander graphs. Journal of CRYPTOLOGY 22(1), 93-113 (2009)
4. Chen, J., Ling, J., Ning, J., Ding, J.: Identity-based signature schemes for multivariate public key cryptosystems. The Computer Journal 62(8), 1132-1147 (2019)
5. Childs, A., Jao, D., Soukharev, V.: Constructing elliptic curve isogenies in quantum subexponential time. Journal of Mathematical Cryptology 8(1), 1-29 (2014)
6. De Feo, L., Kohel, D., Leroux, A., Petit, C., Wesolowski, B.: SQISign: compact post-quantum signatures from quaternions and isogenies. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 64-93. Springer (2020)
7. Deirmentzoglou, E., Papakyriakopoulos, G., Patsakis, C.: A survey on long-range attacks for proof of stake protocols. IEEE Access 7, 28712-28725 (2019)
8. Deuring, M.: Die typen der multiplikatorenringe elliptischer funktionenk'orper. In: Abhandlungen aus dem mathematischen Seminar der Universit'at Hamburg. vol. 14, pp. 197-272. Springer (1941)
9. Gaži, P., Kiayias, A., Russell, A.: Stake-bleeding attacks on proof-of-stake blockchains. In: 2018 Crypto Valley Conference on Blockchain Technology (CVCBT). pp. 85-92. IEEE (2018)
10. Guan, J., Zhandry, M.: Disappearing cryptography in the bounded storage model. In: Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part II 19. pp. 365-396. Springer (2021)
11. Halevi, S., Ishai, Y., Jain, A., Komargodski, I., Sahai, A., Yogev, E.: Noninteractive multiparty computation without correlated randomness. In: Advances in Cryptology-ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part III. pp. 181-211. Springer (2017)
12. Jiang, M., Duong, D.H., Susilo, W.: Puncturable Signature: A Generic Construction and Instantiations. In: Computer Security-ESORICS 2022: 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26-30, 2022, Proceedings, Part II. pp. 507-527. Springer (2022)
13. Li, X., Xu, J., Fan, X., Wang, Y., Zhang, Z.: Puncturable signatures and applications in proof-of-stake blockchain protocols. IEEE Transactions on Information Forensics and Security 15, 3872-3885 (2020)
14. Minkowski, H.: Uber die positiven quadratischen Formen und iiber Kettenbruchanliche Algorithmen. Ges. Abh. I pp. 243-260
15. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. In: Advances in Cryptology-EUROCRYPT 2017: 36th Annual

International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30-May 4, 2017, Proceedings, Part II. pp. 643-673. Springer (2017)
16. Peng, C., Chen, J., Zhou, L., Choo, K.K.R., He, D.: CsiIBS: A post-quantum identity-based signature scheme based on isogenies. Journal of information Security and Applications 54, 102504 (2020)
17. Shaw, S., Dutta, R.: Identification scheme and forward-secure signature in identitybased setting from isogenies. In: Provable and Practical Security: 15th International Conference, ProvSec 2021, Guangzhou, China, November 5-8, 2021, Proceedings 15. pp. 309-326. Springer (2021)
18. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review 41(2), 303-332 (1999)
19. Tian, M., Huang, L.: Identity-based signatures from lattices: simpler, faster, shorter. Fundamenta Informaticae 145(2), 171-187 (2016)
20. Yi, P., Li, J., Liu, C., Han, J., Wang, H., Zhang, Y., Chen, Y.: An efficient identitybased signature scheme with provable security. Information Sciences 576, 790-799 (2021)

# High Weight Code-based Signature Scheme from QC-LDPC Codes 

Chik How Tan ${ }^{1[0000-0001-7550-3890]}$ and Theo Fanuela<br>Prabowo ${ }^{1[0000-0002-4225-9112]}$<br>Temasek Laboratories, National University of Singapore, Singapore, Singapore<br>\{tsltch, tsltfp\}@nus.edu.sg


#### Abstract

We propose a new Hamming metric code-based signature scheme (called HWQCS) based on quasi-cyclic low density parity-check (QC-LDPC) codes. We propose the use of high error on QC-LDPC codes for constructing this signature and analyse its complexity. We show that HWQCS signature scheme achieves EUF-CMA security in the classical random oracle model, assuming the hardness of the syndrome decoding problem and the codeword finding problem for QC-LDPC codes. Furthermore, we also give a detailed security analysis of the HWQCS signature scheme. Based on the complexities of solving the underlying problems, the public key size and signature size of the HWQCS signature scheme are 1568 bytes and 4759 bytes respectively at 128-bit security level.


Keywords: code-based cryptography $\cdot$ signature • QC-LDPC codes

## 1 Introduction

Code-based cryptography is based on the problem of decoding random linear codes, which is referred to as the syndrome decoding problem and is known to be NP-hard [11]. The most common code-based cryptosystems are the McEliece cryptosystem [30] and the Niederreiter cryptosystem [33], which are equivalent in terms of their security. Solving the NP-hard syndrome decoding problem is believed to be hard even for quantum computers. Over the years, a number of code-based cryptographic schemes have been proposed. These include some promising key encapsulation mechanisms called BIKE [4], Classic McEliece[12] and HQC [1], which become fourth-round candidates in the NIST call for postquantum cryptography standardization.

Unlike encryption and key encapsulation mechanisms, the construction of code-based digital signature schemes seems to be more challenging. This is indicated by the absence of code-based signature scheme in the second round onwards of the NIST PQC standardization. The most common techniques to construct signatures are based on two generic frameworks, which are, hash-andsign constructions and Fiat-Shamir framework [23] constructions. The hash-andsign construction requires some trapdoor functions, such as CFS [17] and Wave [19]. On the other hand, Fiat-Shamir framework construction does not necessarily use trapdoor functions in general, such as Stern [41], CVA [15], MPT [31],

CVE [8], cRVDC [9], etc. However, most of them are inefficient or have large key or signature sizes. Furthermore, some of the proposed code-based signatures were even found to be insecure. For example, the KKS [24], RZW [37], CVE [8], SHMWW [39] and MPT [31] are shown to be insecure in [34], [18], [25], [5] and [35] respectively.

Recently, there is a new technique to construct signature schemes, which is called MPC (multiparty computation) in the head paradigm. This approach combines secret key sharing scheme and identification scheme in the multi-party computations setting, for example, CCJ signature [14], FJR signature [22], etc. The purpose of this construction is to reduce the signature size. But most of the signature size is still around eight thousand bytes. Therefore, it is still a challenge to construct signature schemes with practical signature size and public key size.

In this paper, we proposed a new signature scheme (called HWQCS) based on quasi-cyclic low density parity-check (QC-LDPC) codes. The proposed signature scheme is based on the Fiat-Shamir transformation and introduces high weight error on QC-LDPC codes. HWQCS signature scheme resists PrabowoTan's attack [35] on MPT-like signature scheme [31]. This is achieved by signing a message depending on a new ephemeral secret key for each signature rather than relying only on a fixed secret key. So, each signature can be viewed as a onetime signature. Furthermore, this signature is also secure against Bit-Flipping algorithm attack and statistical attack.

The organization of this paper is as follows. In Section 2, we provide a brief review of the properties of linear codes, quasi-cyclic codes and also define the syndrome decoding problem, etc. In Section 3, we propose a new high weight signature scheme (called HWQCS) which is based on 2-quasi-cyclic codes. We also provide security proof of the proposed HWQCS signature scheme under the random oracle model. In Section 4, we give a detailed analysis of various possible attacks on the proposed signature scheme HWQCS. In Section 5, we examine the public/secret key size and signature size for various security levels. Finally, the paper is concluded in Section 6.

## 2 Preliminaries

In this paper, let $n, k$ be integers, denote by $\mathbb{F}_{2}$ the finite field of two elements, let $\mathbf{a}=\left(a_{1}, \ldots, a_{n}\right) \in \mathbb{F}_{2}^{n}$ be a vector in $\mathbb{F}_{2}^{n}$.

### 2.1 Linear Codes

Definition 1 Let $\mathbf{a}=\left(a_{1}, \ldots, a_{n}\right) \in \mathbb{F}_{2}^{n}$. The support of $\mathbf{a}$ is the set consisting of all indices $i \in\{1, \ldots, n\}$ such that $a_{i} \neq 0$. The Hamming weight of $\mathbf{a}$, denoted by $\mathrm{wt}(\mathbf{a})$ is the cardinality of its support. The Hamming distance between $\mathbf{a}$ and $\mathbf{b}$, denoted by $\mathrm{d}(\mathbf{a}, \mathbf{b})$ is defined as $\mathrm{wt}(\mathbf{a}-\mathbf{b})$, i.e., the number of coordinates $\mathbf{a}$ and $\mathbf{b}$ differs on.

Definition 2 Let $k$ and $n$ be two positive integers with $k \leq n$. An $[n, k]$-linear code $\mathcal{C}$ of length $n$ and dimension $k$ is a linear subspace of dimension $k$ of the vector space $\mathbb{F}_{2}^{n}$. The rate of the code $\mathcal{C}$ is $R=\frac{k}{n}$.

Definition 3 Let $\mathcal{C}$ be an $[n, k]$-linear code of length $n$ and dimension $k$. We call its minimum distance $\delta$ the minimum Hamming weight of a non-zero codeword in $\mathcal{C}$, i.e.,

$$
\begin{aligned}
\delta & =\min \{\mathrm{wt}(\mathbf{a}) \mid \mathbf{a} \in \mathcal{C}, \mathbf{a} \neq \mathbf{0}\} \\
& =\min \{\mathrm{wt}(\mathbf{a}-\mathbf{b}) \mid \mathbf{a}, \mathbf{b} \in \mathcal{C}, \mathbf{a} \neq \mathbf{b}\} .
\end{aligned}
$$

We sometimes refer to $\mathcal{C}$ as an $[n, k, \delta]$-code if $\delta$ is known.
Definition $4 A$ matrix $G \in \mathbb{F}_{2}^{k \times n}$ is said to be a generator matrix of an $[n, k]$ linear code $\mathcal{C}$ if its rows form a basis of $\mathcal{C}$. Then, $\mathcal{C}=\left\{\mathbf{u} G \mid \mathbf{u} \in \mathbb{F}_{2}^{k}\right\}$. The parity-check matrix of $\mathcal{C}$ is $H \in \mathbb{F}_{2}^{(n-k) \times n}$ such that $G H^{T}=0$ or $\mathbf{c} H^{T}=0$ for all $\mathbf{c} \in \mathcal{C}$. Furthermore, $G$ and $H$ are said to be in systematic form if they are written as

$$
G=\left[\begin{array}{ll}
I_{k} & A
\end{array}\right] \quad \text { resp. } \quad H=\left[\begin{array}{ll}
I_{n-k} & B
\end{array}\right],
$$

for some $A \in \mathbb{F}_{2}^{k \times(n-k)}$ and $B \in \mathbb{F}_{2}^{(n-k) \times k}$.
Problem 1 (Syndrome Decoding Problem (SDP)). Given a matrix $H \in$ $\mathbb{F}_{2}^{(n-k) \times n}$, a vector $\mathbf{s} \in \mathbb{F}_{2}^{n-k}$ and an integer $w>0$ as input. The Syndrome Decoding problem is to determine a vector $\mathbf{e} \in \mathbb{F}_{2}^{n}$ such that $\mathrm{wt}(\mathbf{e}) \leq w$ and $\mathbf{s}=\mathbf{e} H^{T}$.

Problem 2 (Codeword Finding Problem (CFP)). Given a matrix $H \in$ $\mathbb{F}_{2}^{(n-k) \times n}$, and an integer $w>0$ as input. The Codeword Finding problem is to determine a vector $\mathbf{e} \in \mathbb{F}_{2}^{n}$ such that $\mathrm{wt}(\mathbf{e})=w$ and $\mathbf{e} H^{T}=0$.

The SDP problem and CFP problem are well known and was proved to be NP-complete by Berlekamp, McEliece and van Tilborg in [11]. Moreover, it is proved that there exists a unique solution to SDP if the weight $w$ is below the so-called GV Distance.

Definition 5 Let $\mathcal{C}$ be an $[n, k]$ linear code over $\mathbb{F}_{2}$. The Gilbert-Varshamov (GV) Distance is the largest integer d such that

$$
\sum_{i=0}^{d-1}\binom{n}{i} \leq 2^{n-k}
$$

The first generic decoding method to solve SDP is called the Information Set Decoding (ISD) method, introduced by Prange [36] (denoted as Pra62) in 1962. It is the best known algorithm for decoding a general linear code. Since then, several improvements of the ISD method have been proposed for codes over the binary field, such as LB88 [26], Leon88 [27], Stern88 [40], Dum91 [20],
and more recently by BLP11 [13], MMT11 [28], BJMM12 [7], MO15 [29]. The computational complexity of solving the syndrome decoding problem is quantified by the work factor $\mathcal{W} \mathcal{F}_{\mathcal{A}}(n, k, w)$, which is defined as the average cost in binary operations of algorithm $\mathcal{A}$ to solve it. The work factor of Pra62 is given as follows.

$$
\mathcal{W} \mathcal{F}_{\text {Pra62 }}(n, k, w)=\frac{\min \left\{\binom{n}{w}, 2^{n-k}\right\}}{\binom{n-k}{w}} .
$$

When $w=o(n)$, then $\mathcal{W} \mathcal{F}_{\operatorname{Pra62}}(n, k, w)=\frac{\binom{n}{w}}{\binom{n-k}{w}}$ and $\frac{1}{w} \log _{2} \frac{\binom{n}{w}}{\binom{n-k}{w}} \approx c$, where $c:=-\log _{2}\left(1-\frac{k}{n}\right)$. Therefore, we have $\mathcal{W} \mathcal{F}_{\operatorname{Pra62}}(n, k, w) \approx 2^{c w(1+o(1))}$.

Among the variants of solving algorithms for the syndrome decoding problem, the following result from [42] shows that their work factors are asymptotically the same.

Proposition 1 [42] Let $k$ and $w$ be two functions of $n$ such that $\lim _{n \rightarrow \infty} \frac{k}{n}=R$, $0<R<1$, and $\lim _{n \rightarrow \infty} \frac{w}{n}=0$. For any algorithm $\mathcal{A}$ among the variants of Pra62, Stern88, Dum91, MMT11, BJMM12 and MO15, their work factors are asymptotically the same as

$$
\mathcal{W} \mathcal{F}_{\mathcal{A}}(n, k, w)=2^{c w(1+o(1))}, \quad \text { where } c=-\log _{2}(1-R)
$$

when $n$ tends to infinity.

### 2.2 Quasi-Cyclic Linear Codes

Let $\mathbb{F}_{2}$ be the finite field of two elements and let $\mathcal{R}:=\mathbb{F}_{2}[x] /\left(x^{k}-1\right)$ be the quotient ring of polynomials over $\mathbb{F}_{2}$ of degree less than $k$. Given $a=a_{0}+a_{1} x+$ $\ldots+a_{k-1} x^{k-1} \in \mathcal{R}$, we denote $\mathbf{a}:=\left(a_{0}, a_{1}, \ldots, a_{k-1}\right) \in \mathbb{F}_{2}^{k}$. Let $\mathcal{R}^{*}=\{a \in$ $\mathcal{R} \mid a$ is invertible in $\mathcal{R}\}$. Let $\mathcal{V}$ be a vector space of dimension $k$ over $\mathbb{F}_{2}$. Denote $\mathcal{V}_{k, w}:=\left\{a \in \mathcal{R}=\mathbb{F}_{2}[x] /\left(x^{k}-1\right) \mid \operatorname{wt}(\mathbf{a})=w\right\}$. We sometimes abuse the notation by interchanging a with $a \in \mathcal{R}$.

Definition 6 (Circulant Matrix) Let $\mathbf{v}=\left(v_{0}, \cdots, v_{k-1}\right) \in \mathcal{V}$, a circulant matrix defined by $\mathbf{v}$ is

$$
V:=\left[\begin{array}{cccc}
v_{0} & v_{1} & \ldots & v_{k-1} \\
v_{k-1} & v_{0} & \ldots & v_{k-2} \\
\vdots & \vdots & \ddots & \vdots \\
v_{1} & v_{2} & \ldots & v_{0}
\end{array}\right] \in \mathbb{F}_{2}^{k \times k} .
$$

For $\mathbf{u}, \mathbf{v} \in \mathcal{R}$, the product $\mathbf{w}=\mathbf{u v}$ can be computed as $\mathbf{w}=\mathbf{u} V=\mathbf{v} U$, and $w_{l}=\sum_{i+j=l \bmod k} u_{i} v_{j}$ for $l=0, \cdots, k-1$, where $\mathbf{w}=\left(w_{0}, \cdots, w_{k-1}\right)$. To find the weight of $\mathbf{u v}$, we first compute the probability that $w_{i}=1$, say $p^{\prime}$, then $\mathrm{wt}(\mathbf{w})=p^{\prime} * k$. Now, we compute the probability that $w_{i}=1$ as follows.

Lemma 1 [35] Let $\mathbf{u} \in \mathcal{V}_{k, \omega_{u}}, \mathbf{v} \in \mathcal{V}_{k, \omega_{v}}$ and $\mathbf{w}=\mathbf{u v}=\left(w_{0}, \cdots, w_{k-1}\right)$. Denote the probability that $w_{i}=1$, for $i \in\{0, \cdots, k-1\}$, as $P\left(k, \omega_{u}, \omega_{v}\right)$. Then

$$
P\left(k, \omega_{u}, \omega_{v}\right)=\frac{1}{\binom{k}{\omega_{v}}} \sum_{\substack{\leq l \leq \min \left(\omega_{u}, \omega_{v}\right) \\ l o d d}}\binom{\omega_{u}}{l}\binom{k-\omega_{u}}{\omega_{v}-l} .
$$

Definition 7 (Quasi-Cyclic Codes) A linear block code $\mathcal{C}$ of length lk over $\mathbb{F}_{2}$ is called a quasi-cyclic code of index $l$ if for any $\mathbf{c}=\left(\mathbf{c}_{0}, \cdots, \mathbf{c}_{l-1}\right) \in \mathcal{C}$, the vector obtained after applying a simultaneous circular shift to every block $\mathbf{c}_{0}, \cdots, \mathbf{c}_{l-1}$ is also a codeword.

Definition 8 (Systematic 2-Quasi-Cyclic Codes, 2-QC Codes) A systematic 2-quasi-cyclic $[2 k, k]$-code has generator matrix of the form $\left[\begin{array}{ll}H & I_{k}\end{array}\right] \in \mathbb{F}_{2}^{k \times 2 k}$ and parity check matrix $\left[I_{k} H^{T}\right] \in \mathbb{F}_{2}^{k \times 2 k}$.

Due to the quasi-cyclic structure of a code, any blockwise circular shift of a codeword is also a codeword. So, any circular shift of a syndrome will correspond to a blockwise circular shift of the error pattern. It has been shown in [38] that the work factor of the ISD algorithm for solving the syndrome decoding problem and the codeword finding problem for 2-quasi-cyclic codes for $n=2 k$ are

$$
\mathcal{W} \mathcal{F}_{\mathcal{A}, 2 \mathrm{QCSD}}(n, k, w):=\frac{\mathcal{W} \mathcal{F}_{\mathcal{A}}(n, k, w)}{\sqrt{n-k}}=2^{c[1 / 2+w(1+o(1))]-\left(\log _{2} n\right) / 2}
$$

and

$$
\mathcal{W}_{\mathcal{A}, 2 \mathrm{QCCF}}(n, k, w):=\frac{\mathcal{W} \mathcal{F}_{\mathcal{A}}(n, k, w)}{n-k}=2^{c[1+w(1+o(1))]-\log _{2} n}
$$

respectively. Since the methods and the work factors for solving the syndrome decoding problem and the codeword finding problem for 2-quasi-cyclic codes require exponential time, therefore, we assume that the syndrome decoding problem and the codeword finding problem on quasi-cyclic codes are hard problems. We define the decisional codeword finding problem for 2-quasi-cyclic codes as follows.

Problem 3 (Decisional Codeword Finding Problem for 2-Quasi-Cyclic Codes (2QC-DCFP)). Given a matrix $\left[I_{k} \mathbf{h}\right] \in \mathbf{F}_{2}^{2 k \times k}$, and an even integer $w>0$ as input, decide if there exists $\mathbf{h}_{0}, \mathbf{h}_{1} \in \mathcal{R}$ such that $\operatorname{wt}\left(\mathbf{h}_{0}\right)=\operatorname{wt}\left(\mathbf{h}_{1}\right)=w / 2$ and $\left(\mathbf{h}_{0}, \mathbf{h}_{1}\right)\left[\begin{array}{c}\mathbf{I}_{k} \\ \mathbf{h}\end{array}\right]=0$.

In the special case of 2-quasi-cyclic codes with parity check matrix $H=$ $\left[\mathbf{h}_{0} \mathbf{h}_{1}\right] \in \mathbb{F}_{2}^{k \times 2 k}$, where $\left(\mathbf{h}_{0}, \mathbf{h}_{1}\right)$ and $\mathbf{e}$ are of low weight approximate to $\sqrt{2 k}$, we have what is called the quasi-cyclic low density parity check (QC-LDPC) codes. These codes are commonly used in the construction of key encapsulation mechanisms and signatures, such as BIKE [4] and HQC [1]. The Bit-Flipping algorithm [43] is used to decode an error $\mathbf{e}$ in BIKE.

On the other hand, for our signature (proposed in Section 3), we have $n=2 k$, $H=\left[\mathbf{I}_{k} \mathbf{c}\right]$ and $\mathbf{e}=\left(\mathbf{e}_{1}, \mathbf{e}_{2}\right)$ is of high weight such that $\mathrm{wt}(\mathbf{e}) \gg \sqrt{n}, \frac{\mathrm{wt}(\mathbf{e})}{n}<\frac{1}{2}$, $\frac{\mathrm{wt}\left(\mathbf{e}_{1}\right)}{k}+\frac{\mathrm{wt}\left(\mathbf{c e}_{2}\right)}{k}>\frac{1}{2}$ and $\mathrm{wt}(\mathbf{c})<\sqrt{k}$. Experimental results show that the BitFlipping algorithm [43] is unable to obtain e correctly in this case (many bits are decoded incorrectly). Up to our knowledge, there is no efficient decoding algorithm for high weight error. Therefore, we define the following problem and assume that it is a hard problem.

Problem 4 (Syndrome Decoding Problem for High Weight on QC-LDPC Codes (HWQC-LDPC-SDP)) Let $\omega$ be integer, $n=2 k, H=\left[\begin{array}{ll}\mathbf{I}_{k} & \mathbf{c}\end{array}\right]$ and $\mathbf{e}=\left(\mathbf{e}_{1}, \mathbf{e}_{2}\right)$ is of high weight such that $\omega=\mathrm{wt}(\mathbf{e}) \gg \sqrt{n}, \frac{\mathrm{wt}(\mathbf{e})}{n}<\frac{1}{2}, \frac{\mathrm{wt}\left(\mathbf{e}_{1}\right)}{k}+\frac{\mathrm{wt}\left(\mathbf{c e}_{2}\right)}{k}>\frac{1}{2}$ and $\mathrm{wt}(\mathbf{c})<\sqrt{k}$. Given $H \in \mathbb{F}_{2}^{k \times 2 k}, s \in \mathbb{F}_{2}^{k}$ and $\omega$ as input. The syndrome decoding problem for high weight on $Q C-L D P C$ code is to determine e such that $\mathrm{wt}(\mathbf{e})=\omega$ and $\mathbf{s}=\mathbf{e} H^{T}$.

## 3 HWQCS Signature Scheme

In this section, we present the Hamming-metric code-based digital signature scheme from QC-LDPC codes with high weight errors, which we call the HWQCS signature scheme. The HWQCS signature scheme is based on the hardness of the syndrome decoding problem and the codeword finding problem on quasicyclic codes. Furthermore, the HWQCS signature scheme is different from the MPT signature scheme [31] and is resistant to Prabowo-Tan's attack [35] as each signature can be thought of as a one-time signature with a new ephemeral secret key, while the MPT signature is based on a fixed secret key.

A signature scheme consists of three algorithms: KeyGen, Sign and Verify.

- KeyGen: Given a security parameter $\lambda$, the key generation algorithm returns a key pair ( $\mathrm{pk}, \mathrm{sk}$ ) where pk and sk are the public key and the secret key respectively.
- Sign: The algorithm, on input a message $m$ and the secret key sk, returns a signature $\sigma$.
- Verify: Given a message m, a public key pk and a signature $\sigma$ as input, the algorithm returns either 0 or 1 depending on whether the signature $\sigma$ is valid or not.

Before we describe a HWQCS signature scheme, we first define the required parameters. Let $k, \omega_{f}, \omega_{u}, \omega_{e}, \omega_{c}, \omega_{s}, \omega_{t}$ be integers as public parameters. The HWQCS signature scheme is described as follows.

```
Algorithm 1: Key Generation of HWQCS Signature Scheme
    Input : \(k, \omega_{f}\), security parameter \(\lambda\)
    Output: \(p k=(\mathbf{h})\)
    1 Choose random \(\mathbf{f}_{1}, \mathbf{f}_{2} \in \mathcal{V}_{k, \omega_{f}}\) and both are invertible
    Compute \(\mathbf{h}:=\mathbf{f}_{1}^{-1} \mathbf{f}_{2}\) in \(\mathcal{R}^{*}\)
    3 The public key is \(p k=(\mathbf{h})\) and the secret key is \(s k=\left(\mathbf{f}_{1}, \mathbf{f}_{2}\right)\)
```

```
Algorithm 2: Signing of HWQCS Signature Scheme
    Input : \(k, \omega_{f}, \omega_{u}, \omega_{e}, \omega_{c}, \omega_{s}, \omega_{t}\), message \(m, p k=(\mathbf{h})\) and \(s k=\left(\mathbf{f}_{1}, \mathbf{f}_{2}\right)\)
    Output: signature \(\sigma\)
    Choose random \(\mathbf{e}_{1}, \mathbf{e}_{2} \in \mathcal{V}_{k, \omega_{e}}\) and \(\mathbf{u}_{1}, \mathbf{u}_{2} \in \mathcal{V}_{k, \omega_{u}}\)
    Compute \(\mathbf{b}:=\left(\mathbf{e}_{1}, \mathbf{e}_{2}\right)\left[\begin{array}{c}\mathbf{h} \\ \mathbf{h}^{-1}\end{array}\right]\) in \(\mathcal{R}\)
    Compute \(\mathbf{c}:=\mathcal{H}\left(m\|\mathbf{b}\|\left(\mathbf{u}_{1} \mathbf{f}_{2}+\mathbf{u}_{2} \mathbf{f}_{1}\right) \| p k\right) \in \mathcal{V}_{k, \omega_{c}}\)
    Compute \(\mathbf{s}_{i}:=\mathbf{u}_{i} \mathbf{f}_{i}+\mathbf{c e}_{i}\) in \(\mathcal{R}\) for \(i=1,2\)
    if \(\operatorname{wt}\left(\mathbf{s}_{1}\right)>\omega_{s}\) or \(\operatorname{wt}\left(\mathbf{s}_{2}\right)>\omega_{s}\) or \(\operatorname{wt}\left(\mathbf{u}_{1} \mathbf{f}_{2}+\mathbf{u}_{2} \mathbf{f}_{1}\right)>\omega_{t}\) then
        repeat from Step 1
    else
        the signature is \(\sigma=\left(\mathbf{c}, \mathbf{b}, \mathbf{s}_{1}, \mathbf{s}_{2}\right)\)
    end if
```

```
Algorithm 3: Verification of HWQCS Signature Scheme
    Input : message \(m, p k\), signature \(\sigma=\left(\mathbf{c}, \mathbf{b}, \mathbf{s}_{1}, \mathbf{s}_{2}\right)\)
    Output: validity of the signature
    1 Compute \(\mathbf{t}:=\left(\mathbf{s}_{1}, \mathbf{s}_{2}\right)\left[\begin{array}{c}\mathbf{h} \\ \mathbf{h}^{-1}\end{array}\right]-\mathbf{c b}\) in \(\mathcal{R}\)
    2 Compute \(\mathbf{c}^{\prime}:=\mathcal{H}(m\|\mathbf{b}\| \mathbf{t} \| p k) \in \mathcal{V}_{k, \omega_{c}}\)
    \(\mathbf{3}\) if \(\mathbf{c}^{\prime}=\mathbf{c}\) and \(\mathrm{wt}(\mathbf{t}) \leq \omega_{t}\) and \(\mathbf{t} \neq 0\) in \(\mathcal{R}\) then
    4 the signature is valid
    else
        the signature is invalid
    end if
```


## Correctness:

$$
\begin{aligned}
\mathbf{t} & =\left(\mathbf{s}_{1}, \mathbf{s}_{2}\right)\left[\begin{array}{c}
\mathbf{h} \\
\mathbf{h}^{-1}
\end{array}\right]-\mathbf{c b} \\
& =\left(\mathbf{u}_{1} \mathbf{f}_{2}+\mathbf{\mathbf { e } _ { 1 }} \mathbf{h}\right)+\left(\mathbf{u}_{2} \mathbf{f}_{1}+\mathbf{c e}_{2} \mathbf{h}^{-1}\right)-\mathbf{c}\left(\mathbf{e}_{1} \mathbf{h}+\mathbf{e}_{2} \mathbf{h}^{-1}\right) \\
& =\mathbf{u}_{1} \mathbf{f}_{2}+\mathbf{u}_{2} \mathbf{f}_{1}
\end{aligned}
$$

We define the notion of existential unforgeability under adaptive chosen message attack as follows.

Definition 9 (EUF-CMA Security) A signature scheme is existential unforgeable under adaptive chosen message attack (EUF-CMA) if given a public key pk to any polynomial-time adversary $\mathcal{A}$ who can access the signing oracle Sign(sk, .) and query a number of signatures, then the adversary $\mathcal{A}$ can produce a valid signature $\sigma$ for a message $m$ which has not been previously queried to the signing oracle only with negligible success probability (the success probability is denoted as $\operatorname{Pr}[$ Forge $]$ ).

The advantage Adv of an adversary $\mathcal{A}$ in successfully solving a problem is defined as follows.

Definition 10 The advantage of an adversary $\mathcal{A}$ in solving a problem B denoted as $\operatorname{Adv}(\mathrm{B})$ is defined as the probability that $\mathcal{A}$ successfully solves problem B .

We define the following assumptions which are used to prove the security of the proposed signature scheme.

Assumption 1 (Syndrome Decoding for 2-Quasi-Cyclic Code (2QC-SDP) Assumption) The syndrome decoding for 2-quasi-cyclic code assumption is the assumption that the advantage of an adversary $\mathcal{A}$ in solving 2QC-SDP is negligible, i.e. $\operatorname{Adv}(2 Q C-S D P)<\epsilon_{2 Q C-S D P}$.

Assumption 2 (Codeword Finding for 2-Quasi-Cyclic Codes (2QC-CFP) Assumption) The codeword finding for quasi-cyclic codes assumption is the assumption that the advantage of an adversary $\mathcal{A}$ in solving 2QC-CFP is negligible, i.e. $\operatorname{Adv}(2 Q C-C F P)<\epsilon_{2 Q C-C F P}$.

Assumption 3 (Decisional Codeword Finding for 2-Quasi-Cyclic Codes (2QC-DCFP) Assumption) The decisional codeword finding for 2-quasi-cyclic codes assumption is the assumption that the advantage of an adversary $\mathcal{A}$ in solving 2QC-DCFP is negligible, i.e. $\operatorname{Adv}(2 Q C-D C F P)<\epsilon_{2 Q C-D C F P}$.

Assumption 4 (Syndrome Decoding for High Weight on QC-LDPC Codes (HWQC-LDPC-SDP) Assumption) The syndrome decoding for high weight of $Q C$ LDPC codes assumption is the assumption that the advantage of an adversary $\mathcal{A}$ in solving HWQC-LDPC-SDP is negligible, i.e. $\operatorname{Adv}(H W Q C-L D P C-S D P)<\epsilon_{\text {HWQC-LDPC-SDP }}$.

Theorem 1 Under the 2QC-SDP, 2QC-DCFP, 2QC-CFP, HWQC-LDPC-SDP assumptions, the HWQCS signature scheme with parameters $\left(k, \omega_{f}, \omega_{u}, \omega_{e}, \omega_{c}, \omega_{s}, \omega_{t}\right)$ is secure under the EUF-CMA model in the classical random oracle model.

Proof. We consider a chosen-message EUF adversary $\mathcal{A}$ against the HWQCS signature scheme. To prove the security, adversary $\mathcal{A}$ interacts with the real signature scheme and makes a sequence of experiments. The adversary $\mathcal{A}$ is first given a public key $\mathbf{h}$. $\mathcal{A}$ made $q_{s}$ signing queries and $q_{\mathcal{H}}$ hash $(\mathcal{H})$ queries. Finally, $\mathcal{A}$ outputs a message/signature pair such that the message has not been queried previously to the signing oracle. Let $\mathrm{Pr}_{i}[$ Forge $]$ be the probability of an event in experiment $i$ that $\mathcal{A}$ obtains a valid signature of a message that has not been queried previously to the signing oracle. Let $\mathrm{Pr}_{0}$ [Forge] be the success probability of an adversary $\mathcal{A}$ at the beginning (Experiment 0 ). Our goal is to give an upper-bound of $\operatorname{Pr}_{0}$ [Forge].
Experiment 1. During the course of the experiment, if there is a collision in $\mathcal{H}$, then we abort the experiment. The number of queries to the hash oracle or the signing oracle throughout the experiment is at most $q_{s}+q_{\mathcal{H}}$. Thus,

$$
\mid \operatorname{Pr}_{0}[\text { Forge }]-\operatorname{Pr}_{1}[\text { Forge }] \left\lvert\, \leq \frac{q_{s}+q_{\mathcal{H}}}{\binom{k}{\omega_{c}}} .\right.
$$

Experiment 2. During the course of the experiment, $\mathcal{A}$ received a number of signatures $\sigma_{j}=\left(\mathbf{c}, \mathbf{b}, \mathbf{s}_{1}, \mathbf{s}_{2}\right)_{j}$ for $j=1, \cdots, q_{s}$. If $\mathcal{A}$ could solve for $\left(\mathbf{e}_{1}, \mathbf{e}_{2}\right)_{j}$ from $\mathbf{b}_{j}=\left(\mathbf{e}_{1}, \mathbf{e}_{2}\right)_{j}\left[\begin{array}{c}\mathbf{h} \\ \mathbf{h}^{-1}\end{array}\right]$ for some $j$, then $\mathcal{A}$ could forge a new signature. But, the probability that $\mathcal{A}$ could solve it is bounded by $\epsilon_{2 \text { QC-SDP }}$ and we abort the experiment in this case. Thus,

$$
\mid \operatorname{Pr}_{1}[\text { Forge }]-\operatorname{Pr}_{2}[\text { Forge }] \mid \leq \epsilon_{2 Q C-\text { sDP } .}
$$

Experiment 3. During the course of the experiment, $\mathcal{A}$ received a number of signatures $\sigma_{j}=\left(\mathbf{c}, \mathbf{b}, \mathbf{s}_{1}, \mathbf{s}_{2}\right)_{j}$ for $j=1, \cdots, q_{s}$. If $\mathcal{A}$ could solve for $\left(\mathbf{u}_{i} \mathbf{f}_{i}, \mathbf{e}_{i}\right)$ from $\left(\mathbf{s}_{i}\right)_{j}=\left(\mathbf{u}_{i} \mathbf{f}_{i}, \mathbf{e}_{i}\right)_{j}\left[\begin{array}{l}\mathbf{I}_{k} \\ \mathbf{c}_{j}\end{array}\right]$ for $i=1,2$ for some $j$, then $\mathcal{A}$ could forge a new signature. But, the probability that $\mathcal{A}$ could solve it is bounded by $\epsilon_{\text {HwQC-LDPc-SDP }}$ and we abort the experiment in this case. Thus,

$$
\mid \operatorname{Pr}_{2}[\text { Forge }]-\operatorname{Pr}_{3}[\text { Forge }] \mid \leq 2 \epsilon_{\text {HWQC-LDPC-SDP. }}
$$

Experiment 4. In this experiment, a public key $\mathbf{h}$ is replaced by a random $\mathbf{h}^{\prime} \in$ $\mathcal{R}^{*}$. To distinguish Experiment 4 from Experiment 3, the adversary must in fact distinguish a well-formed public key $\mathbf{h}=\mathbf{f}_{1}^{-1} \mathbf{f}_{2}$ from a random invertible element of $\mathcal{R}$. Thus, we have

$$
\mid \operatorname{Pr}_{3}[\text { Forge }]-\operatorname{Pr}_{4}[\text { Forge }] \mid \leq \epsilon_{2 \mathrm{QC}-\text { DcFP. }} .
$$

Furthermore, in this experiment, an adversary $\mathcal{A}$ has no signature information on $\mathbf{h}^{\prime}$ and needs to solve a codeword finding problem for 2-quasi-cyclic codes in order to forge a signature. Thus,

$$
\mid \operatorname{Pr}_{4}[\text { Forge }] \mid \leq \epsilon_{2 Q C-C F P} .
$$

Combining the above experiments, the success probability of the adversary $\mathcal{A}$ is

$$
\begin{aligned}
\mid \operatorname{Pr}_{0}[\text { Forge }] \mid & \leq \sum_{i=0}^{3} \mid \operatorname{Pr}_{i}[\text { Forge }]-\operatorname{Pr}_{i+1}[\text { Forge }]|+| \operatorname{Pr}_{4}[\text { Forge }] \mid \\
& \leq \epsilon_{2 Q C-C F P}+\epsilon_{2 \mathrm{QC}-\mathrm{DCFP}}+2 \epsilon_{\text {HWQC-LDPC-SDP }}+\epsilon_{2 \mathrm{QC} \text {-SDP }}+\frac{q_{s}+q_{\mathcal{H}}}{\binom{k}{\omega_{c}}} .
\end{aligned}
$$

## 4 Security Analysis

Let $\lambda$ be the security level. For the security analysis, we consider two common types of attacks, namely, key recovery attacks and signature forgeries.

### 4.1 Key Recovery Attack

Finding the secret key ( $\mathbf{f}_{1}, \mathbf{f}_{2}$ ) from the public key $\mathbf{h}=\mathbf{f}_{1}^{-1} \mathbf{f}_{2}$ is equivalent to finding the codeword ( $\mathbf{f}_{1}, \mathbf{f}_{2}$ ) with parity check matrix $\left[\begin{array}{ll}\mathbf{h} & \mathbf{I}_{k}\end{array}\right]$ such that $\left(\mathbf{f}_{1}, \mathbf{f}_{2}\right)\left[\begin{array}{c}\mathbf{h} \\ \mathbf{I}_{k}\end{array}\right]=\mathbf{0}$. The work factor of solving the codeword finding problem for quasi-cyclic parity-check codes is

$$
\mathcal{W} \mathcal{F}_{\mathcal{A}, 2 Q C C F}\left(2 k, k, 2 \omega_{f}\right)=2^{c\left[1+2 \omega_{f}(1+o(1))\right]-\log _{2} 2 k}, \quad \text { where } c=1
$$

Therefore, we can prevent key recovery attack by choosing the parameters such that $1+2 \omega_{f}(1+o(1))-\log _{2} 2 k \geq \lambda$, where $\lambda$ is the security level.

Another method to find the secret key $\left(\mathbf{f}_{1}, \mathbf{f}_{2}\right)$ is by performing exhaustive search for $\mathbf{f}_{1}$ and checking whether $\mathbf{f}_{1} \mathbf{h}$ is of small Hamming weight $w_{f}$. The complexity of performing this exhaustive search is $\binom{k}{\omega_{f}}$. So, we must choose the parameters such that $\log _{2}\binom{k}{\omega_{f}} \geq \lambda$, where $\lambda$ is the security level.

Based on the above, we choose the parameters such that

$$
\min \left\{\log _{2}\binom{k}{\omega_{f}}, 1+2 \omega_{f}(1+o(1))-\log _{2} 2 k\right\} \geq \lambda
$$

This ensures that the scheme is resistant against key recovery attacks.

### 4.2 Signature Forgery

### 4.2.1 Collision

For a signature scheme based on the Schnorr scheme, it is important to address the issue of collisions between different messages. In order to prevent collisions, one way is to use a collision-free hash function. Another way is to use a secure hash function such that the collision is minimal, that is, satisfying $\log _{2}\binom{k}{\omega_{c}} \geq 2 \lambda$, where $\lambda$ is the security level.

### 4.2.2 Forgery From Known Signature

We consider the following methods to forge a signature.

### 4.2.2.1 Forgery via Syndrome Decoding Algorithm

From a given signature, we have $\mathbf{b}=\mathbf{e}_{1} \mathbf{h}+\mathbf{e}_{2} \mathbf{h}^{-1}, \mathbf{s}_{i}=\mathbf{u}_{i} \mathbf{f}_{i}+\mathbf{c} \mathbf{e}_{i}$, where $i=1,2$. Equivalently, $\mathbf{b}=\left(\mathbf{e}_{1}, \mathbf{e}_{2}\right)\left[\begin{array}{c}\mathbf{h} \\ \mathbf{h}^{-1}\end{array}\right], \mathbf{s}_{i}=\left(\mathbf{u}_{i} \mathbf{f}_{i}, \mathbf{e}_{i}\right)\left[\begin{array}{c}\mathbf{I}_{k} \\ \mathbf{c}\end{array}\right]$ for $i=1,2$.
(1) One may use syndrome decoding algorithms to recover $\left(\mathbf{e}_{1}, \mathbf{e}_{2}\right)$ from $\mathbf{b}=$ $\left(\mathbf{e}_{1}, \mathbf{e}_{2}\right)\left[\begin{array}{c}\mathbf{h} \\ \mathbf{h}^{-1}\end{array}\right]$. The work factor is
$\mathcal{W} \mathcal{F}_{\mathcal{A}, 2 Q C S D}\left(2 k, k, 2 \omega_{e}\right)=2^{c\left[1 / 2+2 \omega_{e}(1+o(1))\right]-\left(\log _{2} 2 k\right) / 2}, \quad$ where $c=1$.
In order to prevent this attack, we choose $k, \omega_{e}$ such that

$$
1 / 2+2 \omega_{e}(1+o(1))-\left(\log _{2} 2 k\right) / 2 \geq \lambda,
$$

where $\lambda$ is the security level.
(2) One may also use syndrome decoding algorithms to recover ( $\left.\mathbf{e}_{1}, \mathbf{e}_{2}, \mathbf{u}_{1} \mathbf{f}_{1}, \mathbf{u}_{2} \mathbf{f}_{2}\right)$ from

$$
\left[\begin{array}{c}
\mathbf{b} \\
\mathbf{s}_{1} \\
\mathbf{s}_{2}
\end{array}\right]=\left[\begin{array}{cccc}
\mathbf{h} & \mathbf{h}^{-1} & \mathbf{0}_{k} & \mathbf{0}_{k} \\
\mathbf{c} & \mathbf{0}_{k} & \mathbf{I}_{k} & \mathbf{0}_{k} \\
\mathbf{0}_{k} & \mathbf{c} & \mathbf{0}_{k} & \mathbf{I}_{k}
\end{array}\right]\left[\begin{array}{c}
\mathbf{e}_{1} \\
\mathbf{e}_{2} \\
\mathbf{u}_{1} \mathbf{f}_{1} \\
\mathbf{u}_{2} \mathbf{f}_{2}
\end{array}\right]
$$

Note that the weight of $\left(\mathbf{e}_{1}, \mathbf{e}_{2}, \mathbf{u}_{1} \mathbf{f}_{1}, \mathbf{u}_{2} \mathbf{f}_{2}\right)$ is $\omega=2\left(\omega_{e}+\operatorname{wt}\left(\mathbf{u}_{1} \mathbf{f}_{1}\right)\right)$. So, the work factor is

$$
\mathcal{W} \mathcal{F}_{\mathcal{A}, 4 \mathrm{QCSD}}(4 k, k, \omega)=\frac{\min \left\{\binom{4 k}{\omega}, 2^{4 k-k}\right\}}{\binom{4 k-k}{\omega} \sqrt{4 k-k}}=\frac{\min \left\{\binom{4 k}{\omega}, 2^{3 k}\right\}}{\binom{3 k}{\omega} \sqrt{3 k}} .
$$

(3) Another method to find the ephemeral secret $\left(\mathbf{e}_{1}, \mathbf{e}_{2}\right)$ is by performing exhaustive search on $\mathbf{e}_{1}$ and checking whether $\mathbf{e}_{2}=\mathbf{b h}+\mathbf{e}_{1} \mathbf{h}^{2}$ is of small Hamming weight $w_{e}$. The complexity of performing this method is $\binom{k}{\omega_{e}}$. In order to prevent this attack, we choose $k$, $\omega_{e}$ such that $\log _{2}\binom{k}{\omega_{e}} \geq \lambda$, where $\lambda$ is the security level.

Suppose an adversary can recover $\left(\mathbf{e}_{1}, \mathbf{e}_{2}\right)$ using any of the above methods. Then, the adversary obtains $\mathbf{u}_{i} \mathbf{f}_{i}=\mathbf{s}_{i}-\mathbf{c} \mathbf{e}_{i}$ for $i=1,2$. Afterwards, he can forge
a new signature by generating new $\mathbf{b}^{\prime}=\mathbf{e}_{1}^{\prime} \mathbf{h}+\mathbf{e}_{2}^{\prime} \mathbf{h}^{-1}$ and setting $\mathbf{s}_{i}^{\prime}=\mathbf{u}_{i} \mathbf{f}_{i}+\mathbf{c}^{\prime} \mathbf{e}_{i}^{\prime}$, for $i=1,2$.

Based on the above analysis, in order to resist forgery attacks with security level $\lambda$, we choose the parameters $k, \omega, \omega_{e}$ satisfying the following conditions:

$$
\min \left\{\log _{2}\binom{k}{\omega_{e}}, \quad 1 / 2+2 \omega_{e}(1+o(1))-\left(\log _{2} 2 k\right) / 2, \log _{2} \frac{\left.\min \left\{\begin{array}{c}
4 k \\
\omega
\end{array}\right), 2^{3 k}\right\}}{\binom{3 k}{\omega} \sqrt{3 k}}\right\} \geq \lambda
$$

### 4.2.2.2 Forgery via Bit-Flipping Algorithm

Given a signature, we have $\mathbf{s}_{i}=\mathbf{u}_{i} \mathbf{f}_{i}+\mathbf{c e}_{i}$, where $i=1,2$. One may try to apply the bit-flipping algorithm on $\mathbf{s}_{i}=\left(\mathbf{u}_{i} \mathbf{f}_{i}, \mathbf{e}_{i}\right)\left[\begin{array}{c}\mathbf{I}_{k} \\ \mathbf{c}\end{array}\right]$ for $i=1,2$ to recover $\mathbf{e}_{i}$.

In this case, $n=2 k, H=\left[\begin{array}{c}\mathbf{I}_{k} \\ \mathbf{c}\end{array}\right]$ and the threshold $\tau=\left\lfloor\rho \cdot \omega_{c}\right\rfloor$, where $\rho$ is the probability that $\left(\mathbf{e}_{i}\right)_{j}=\left(\mathbf{s}_{i}\right)_{j}=1$ for $j \in\{0, \cdots, k-1\}$ and will be given in the following proposition.
Proposition 2 If $\mathbf{c}_{0}=1$ and $\left(\mathbf{s}_{i}\right)_{j}=1$, then $\rho=\operatorname{Prob}\left[\left(\mathbf{e}_{i}\right)_{j}=\left(\mathbf{s}_{i}\right)_{j}=1\right]$ is equal to
$\left(1-P\left(k, \omega_{u}, \omega_{f}\right)\right) *\left(1-P\left(k, \omega_{c}-1, \omega_{e}-1\right)\right)+P\left(k, \omega_{u}, \omega_{f}\right) * P\left(k, \omega_{c}-1, \omega_{e}-1\right)$.
Proof. If $\mathbf{c}_{0}=1$, then

$$
\left(\mathbf{s}_{i}\right)_{j}=\left(\mathbf{u}_{i} \mathbf{f}_{i}+\mathbf{c e}_{i}\right)_{j}=\left(\mathbf{e}_{i}\right)_{j}+\sum_{l=0}^{k-1}\left(\mathbf{u}_{i}\right)_{l}\left(\mathbf{f}_{i}\right)_{j-l \bmod k}+\sum_{\substack{0 \leq \leq \leq k-1 \\ l \neq j}} \mathbf{c}_{l}\left(\mathbf{e}_{i}\right)_{j-l \bmod k} .
$$

Note that the probability that $\left(\mathbf{u}_{i} \mathbf{f}_{i}\right)_{j}=1$ and $\sum_{l \neq j}\left(\mathbf{c}_{i}\right)_{l}\left(\mathbf{e}_{i}\right)_{j-l \bmod k}=1$ are $P\left(k, \omega_{u}, \omega_{f}\right)$ and $P\left(k, \omega_{c}-1, \omega_{e}-1\right)$ respectively. Hence, the probability that $\left(\mathbf{e}_{i}\right)_{j}=\left(\mathbf{s}_{i}\right)_{j}=1$ is
$\left(1-P\left(k, \omega_{u}, \omega_{f}\right)\right) *\left(1-P\left(k, \omega_{c}-1, \omega_{e}-1\right)\right)+P\left(k, \omega_{u}, \omega_{f}\right) * P\left(k, \omega_{c}-1, \omega_{e}-1\right)$.
As in Problem 4, we choose the parameters such that $\operatorname{wt}\left(\mathbf{u}_{i} \mathbf{f}_{i}\right)+\omega_{e} \gg \sqrt{2 k}$, $\frac{\mathrm{wt}\left(\mathbf{u}_{i} \mathbf{f}_{i}\right)+\mathrm{wt}\left(\mathbf{c e}_{i}\right)}{k}>\frac{1}{2}$ and $\omega_{c} \ll \sqrt{k}$. With this choice of parameters, the bitflipping algorithm will not be able to decode correctly to obtain $\mathbf{e}_{i}$ for $i=1,2$. Hence, one cannot obtain $\mathbf{u}_{i} \mathbf{f}_{i}$ and forge a new signature.

### 4.2.3 Forgery Without Knowing Any Signature

Note that an adversary can generate $\mathbf{b}=\mathbf{e}_{1} \mathbf{h}+\mathbf{e}_{2} \mathbf{h}^{-1}$. To forge a signature, the adversary has to produce $\mathbf{s}_{i}$ of low weight. As the adversary needs to produce $\mathbf{u}_{i} \mathbf{f}_{i}$ of low Hamming weight and $\mathbf{u}_{1} \mathbf{f}_{1} \mathbf{h}$ such that $\mathbf{u}_{2} \mathbf{f}_{2} \mathbf{h}^{-1}$ are also of low Hamming weight, therefore $\mathrm{wt}\left(\mathbf{u}_{i} \mathbf{f}_{i}\right)$ must be set to low. In order to ensure this, we need to define the normal distribution and present the following lemma and corollary.

Let $\mathcal{N}\left(0, \sigma^{2}\right)$ be the normal distribution with mean 0 and standard deviation $\sigma$. Its density function is $\rho_{\sigma}(x)=\left(\frac{1}{\sqrt{2 \pi \sigma^{2}}}\right) e^{-\frac{x^{2}}{2 \sigma^{2}}}$ for $x \in \mathbb{R}$.

Lemma 2 [16] For $k>2, Z \sim \mathcal{N}\left(0, \sigma^{2}\right)$, then

$$
\operatorname{Pr}[|z|>k \sigma \mid z \leftarrow Z] \leq \frac{1}{2}\left(e^{-k^{2}}+e^{-\frac{k^{2}}{2}}\right) .
$$

Corollary 1 (1) For $\kappa>2, Y \sim \mathcal{N}\left(\mu, \sigma^{2}\right)$, then $\operatorname{Pr}[|y-\mu|>\kappa \sigma \mid y \leftarrow Y] \leq$ $\frac{1}{2}\left(e^{-\kappa^{2}}+e^{-\frac{\kappa^{2}}{2}}\right)$.
(2) Let $n$ be a large positive integer and $0<p<1$. If $Y$ is a binomial distribution with parameters $n$ and $p$ (denoted $\operatorname{Bin}(n, p)$ ), then $Y$ approximates to $\mathcal{N}\left(\mu, \sigma^{2}\right)$, where $\mu=n p$ and $\sigma=\sqrt{n p(1-p)}$.
(3) In (2), if $0<l<p<1$ and $\kappa=\frac{(p-l) \sqrt{n}}{\sqrt{p(1-p)}}$, then

$$
\operatorname{Pr}[|y-n p|>(p-l) n \mid y \leftarrow Y] \leq \frac{1}{2}\left(e^{-\kappa^{2}}+e^{-\frac{\kappa^{2}}{2}}\right)<e^{-\kappa^{2} / 2}
$$

Setting $n=k, p=\frac{1}{2}$ and $l<\frac{1}{2}$ in Corollary 1 (3), we have $\operatorname{Pr}\left[\left|y-\frac{k}{2}\right|>\right.$ $\left.\left.\left(\frac{1}{2}-l\right) k \right\rvert\, y \leftarrow \operatorname{Bin}(k, p)\right]<e^{-\kappa^{2} / 2}$. To ensure that the probability is negligible, we should choose $\kappa$ such that $\kappa=(1-2 l)) \sqrt{k}$ and $\frac{1}{2}\left(e^{-\kappa^{2}}+e^{-\kappa^{2} / 2}\right)<e^{-\kappa^{2} / 2}<2^{-\lambda}$, that is,

$$
\frac{\kappa^{2}}{2} \log _{2} e>\lambda \Longrightarrow \kappa>\sqrt{\frac{2 \lambda}{\log _{2} e}}
$$

Letting $\kappa_{0}=\sqrt{\frac{2 \lambda}{\log _{2} e}}$, we have

| $\lambda$ | 128 | 192 | 256 |
| :---: | :---: | :---: | :---: |
| $\kappa_{0}$ | 13.320 | 16.314 | 18.838 |

This means that if an adversary randomly picks an element a in place of $\mathbf{u}_{i} \mathbf{f}_{i}$ for $i=1,2$, then the probability that $\left|\mathrm{wt}(\mathbf{a})-\frac{k}{2}\right| \leq \kappa \sqrt{k / 4}$ is more than $1-2^{-\lambda}$. Hence, by selecting appropriate $l, k$ such that $(1-2 l) \sqrt{k} \geq \kappa_{0}$, we can ensure that the adversary cannot find a of weight less than $l k$. Therefore, it is not possible to forge a signature with probability more than $2^{-\lambda}$.

## 5 Parameters Selections

Based on the above security analysis, the parameters $\left(k, \omega_{f}, \omega_{u}, \omega_{e}, \omega_{c}, \omega_{s}\right)$ of the signature scheme must be chosen properly in order to achieve $\lambda$-bit computa-
tional security. The following conditions are to be fulfilled:

$$
\begin{aligned}
& \min \left\{\log _{2}\binom{k}{\omega_{f}}, 1+2 \omega_{f}(1+o(1))-\log _{2} 2 k\right\} \geq \lambda, \\
& \log _{2}\binom{k}{\omega_{c}} \geq 2 \lambda, \\
& \min \left\{\log _{2}\binom{k}{\omega_{e}}, \frac{1}{2}+2 \omega_{e}(1+o(1))-\frac{\log _{2} 2 k}{2}, \log _{2} \frac{\min \left\{\binom{4 k}{\omega}, 2^{3 k}\right\}}{\binom{3 k}{\omega} \sqrt{3 k}}\right\} \geq \lambda, \\
&(1-2 l) \sqrt{k}>\sqrt{\frac{2 \lambda}{\log _{2} e}} \\
& \frac{\mathrm{wt}^{2}\left(\mathbf{u}_{i} \mathbf{f}_{i}\right)+\omega_{e}}{}>\sqrt{2 k} \\
& \frac{\mathrm{wt}\left(\mathbf{u}_{i} \mathbf{f}_{i}\right)+\mathrm{wt}\left(\mathbf{c e}_{i}\right)}{k}>\frac{1}{2}
\end{aligned}
$$

The parameters for various security levels are given in the following Table 1.

Table 1. The parameters of the HWQCS signature

| Name | $\lambda$ | $k$ | $\omega_{f}$ | $\omega_{u}$ | $\omega_{e}$ | $\omega_{c}$ | $\frac{\mathrm{wt}(\mathbf{s})}{k}$ | $\frac{\mathrm{wt}(\mathbf{u f})}{k}$ | $\frac{\mathrm{wt}(\mathbf{t})}{k}$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Para-1 | 128 | 12539 | 145 | 33 | 141 | 31 | 0.3863 | 0.2694 | 0.3937 |
| Para-2 | 192 | 18917 | 185 | 41 | 177 | 39 | 0.3938 | 0.2779 | 0.4013 |
| Para-3 | 256 | 25417 | 201 | 51 | 191 | 51 | 0.3978 | 0.2786 | 0.4019 |

To compute the size of HWQCS signature scheme, the public key size is $\lceil k / 8\rceil$ bytes, the secret key size is $2 *\left\lceil\left\lceil\log _{2} k\right\rceil * \omega_{f} / 8\right\rceil$ bytes and the signature size is $3 *\lceil k / 8\rceil+\left\lceil\left\lceil\log _{2} k\right\rceil * \omega_{c} / 8\right\rceil$ bytes. We list their sizes for various security levels in Table 2.

Table 2. Size of Signature Schemes (at certain classical security levels)

| Scheme | Security | Size (in Bytes) |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  |  | PK | SK | Sg |
| HWQCS-I | 128 | 1,568 | 508 | 4,759 |
| HWQCS-II | 192 | 2,365 | 694 | 7,169 |
| HWQCS-III | 256 | 3,178 | 754 | 9,630 |

As listed in Table 2, the public key size, secret key size and signature size of the proposed signature scheme HWQCS-I are 1568 bytes, 508 bytes and 4759 bytes respectively for 128 -bit classical security level.

We provide comparison of the key sizes and signature size for various codebased signature schemes in Table 3.

Table 3. Comparison of Various Code-based Signature Schemes (at certain classical security levels)

| Scheme | PK size | SK size | Sg size | C.Sec |
| :---: | ---: | ---: | ---: | ---: |
| HWQCS-I | 1.568 KB | 508 B | 4.759 KB | 128 |
| Durandal-I19 [3] | 15.25 KB | 2.565 KB | 4.060 KB | 128 |
| WAVE23 [32] | 3.60 MB | 2.27 MB | 737 B | 128 |
| CCJ23 [14] | 90 B | 231 B | 12.52 KB | 128 |
| SDitH23 [2] | 120 B | 404 B | 8.26 KB | 128 |
| BG23 [10] | 1 KB | 2 KB | 13.5 KB | 128 |
| cRVDC19 [9] | 0.152 KB | 0.151 KB | 22.480 KB | 125 |
| CVE18 [8] | 7.638 KB | 0.210 KB | 436.600 KB | 80 |

In Table 3, it can be observed that the signature size of the proposed signature scheme HWQCS-I is smaller than the other signature schemes except for the WAVE23 signature scheme [32] and the Durandal-I19 signature scheme [3]. However, it should be noted that the public key sizes for both the WAVE23 and Durandal-I19 signature schemes exceed ten thousand bytes. These are larger than that of the signature scheme HWQCS-I. Moreover, recently there is an attack on Durandal-I19 [6] which requires it to increase its parameter sizes.

Although the public key size of the CCJ23 signature scheme [14] and the SDitH23 signature scheme [2] are relatively small, but their signature sizes are more than eight thousand bytes. Overall, the proposed signature scheme HWQCS-I has shorter combined key and signature sizes than other signature schemes.

## 6 Conclusion

In this paper, we constructed a new Hamming metric code-based signature scheme (called HWQCS signature scheme). The security of HWQCS signature is based on the hardness of the syndrome decoding problem and the codeword finding problem on 2-quasi-cyclic codes, as well as on high error for quasi-cyclic
low parity-check codes respectively. We provided security proof of the HWQCS signature scheme under the random oracle model and gave detailed analysis on the security of the HWQCS signature scheme against Bit-Flipping attack and statistical attack. Furthermore, we also provided concrete parameter choices for the HWQCS signature scheme and compared its key sizes and signature size to other existing signature schemes. The signature scheme HWQCS-I outperforms other code-based signature schemes with a public key size of 1568 bytes, secret key size of 508 bytes and signature size of 4759 bytes at 128 -bit security level.

## References

1. Aguilar-Melchor, C., et al.: Hamming Quasi-Cyclic (HQC). Submission to the NIST post quantum standardization process (2017). https://www.pqc-hqc.org/doc/ hqc-specification_2021-06-06.pdf
2. Aguilar-Melchor, $\overline{\text { C., et al.: The Syndrome Decoding in the Head (SD-in-the-Head) }}$ Signature Scheme. Submission to the NIST call for additional post-quantum signatures (2023). https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/ round-1/spec-files/SDitH-spec-web.pdf
3. Aragon, N., Blazy, O., Gaborit, P., Hauteville, A., Zémor, G.: Durandal: a rank metric based signature scheme. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 728-758. Springer, Cham (2019). https://doi.org/ 10.1007/978-3-030-17659-4_25
4. Aragon, N., et al.: BIKE: Bit Flipping Key Encapsulation. Submission to the NIST post quantum standardization process (2017). https://bikesuite.org/files/v5. 0/BIKE_Spec.2022.10.10.1.pdf
5. Aragon, N., Baldi, M., Deneuville, J.C., Khathuria, K., Persichetti, E., Santini, P. : Cryptanalysis of a code-based full-time signature. Designs, Codes and Cryptography 89 (9) (2021) 2097-2112. https://doi.org/10.1007/s10623-021-00902-7
6. Aragon, N., Dyseryn, V., Gaborit, P.: Analysis of the Security of the PSSI Problem and Cryptanalysis of the Durandal Signature Scheme. In: Handschuh, H., Lysyanskaya, A. (eds) CRYPTO 2023. LNCS, vol. 14083. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-38548-3_5
7. Becker, A., Joux, A., May, A., Meurer, A.: Decoding Random Binary Linear Codes in $2^{n / 20}$ : How $1+1=0$ improves information set decoding. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 520-536. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_31
8. Bellini, E., Caullery, F., Hasikos, A., Manzano, M., Mateu, V.: Code-based signature schemes from identification protocols in the rank metric. In: Camenisch, J., Papadimitratos, P. (eds.) CANS 2018. LNCS, vol. 11124, pp. 277-298. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00434-7_14
9. Bellini, E., Caullery, F., Gaborit, P., Manzano, M., Mateu, V.: Improved Veron identification and signature schemes in the rank metric. In: IEEE International Symposium on Information Theory, pp. 1872-1876 (2019). https://doi.org/10.1109/ ISIT.2019.8849585
10. Bidoux, L., Gaborit, P.: Compact Post-quantum signatures from proofs of knowledge leveraging structure for the PKP, SD and RSD Problems. In: Hajji,S. El., et al. (Eds.): C2SI 2023, LNCS, vol. 13874, pp. 10-42. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-33017-9_2
11. Berlekamp, E., McEliece, R., van Tilborg, H.: On the inherent intractability of certain coding problems (corresp.). IEEE Trans. Inf. Theory 24(3), 384-386 (1978). https://doi.org/10.1109/TIT.1978.1055873
12. Bernstein, D.J., et al.: Classic McEliece: conservative code-based cryptography. Submission to the NIST post quantum standardization process (2017). https:// classic.mceliece.org/mceliece-rationale-20221023.pdf
13. Bernstein, D.J., Lange, T., Peters, C.: Smaller decoding exponents: ball-collision decoding. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 743-760. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_42
14. Carozza, E., Couteau, G. Joux, A. : Short Signatures from Regular Syndrome Decoding in the Head. In: Hazay, C., Stam, M. (eds) EUROCRYPT 2023. LNCS, vol. 14008, pp. 532-563. Springer, Cham (2023). https://doi.org/10.1007/ 978-3-031-30589-4_19
15. Cayrel P.L., Véron P., El Yousfi Alaoui S.M.: A zero-knowledge identification scheme based on the q-ary syndrome decoding problem. In: Biryukov, A., Gong, G., Stinson, D.R. (eds) SAC 2010. LNCS, vol. 6544, pp. 171-186. Springer, Berlin (2011). https://doi.org/10.1007/978-3-642-19574-7_12
16. Chiani, M., Dardari, D., Simon, M.K.: New exponential bounds and approximations for the computation of error probability in fading channels. IEEE Trans. Wireless Commun. 2(4), 840-845 (2003)
17. Courtois, N.T., Finiasz, M., Sendrier, N.: How to achieve a McEliece-based digital signature scheme. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 157-174. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_10
18. D'Alconzo, G., Meneghetti, A., Piasenti, P.: Security issues of CFS-like digital signature algorithms. arXiv preprint arXiv:2112.00429, 2021. https://arxiv.org/abs/ 2112.00429
19. Debris-Alazard, T., Sendrier, N., Tillich, J. P.: Wave: a new family of trapdoor one-way preimage sampleable functions based on codes. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11921, pp. 21-51. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34578-5_2
20. Dumer, I.: On minimum distance decoding of linear codes. In: Proceedings of the 5 th Joint Soviet-Swedish International Workshop Information Theory, pp. 50-52 (1991)
21. Esser, A., Bellini, E.: Syndrome decoding estimator. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) PKC 2022. LNCS, vol. 13177, pp. 112-141. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-030-97121-2_5
22. Feneuil, T., Joux, A., Rivain, M.: Syndrome decoding in the head: shorter signatures from zero-knowledge proofs. In: Dodis, Y., Shrimpton, T. (eds) CRYPTO 2022. LNCS, vol. 13508, pp. 541-572. Springer, Cham (2022). https://doi.org/10. 1007/978-3-031-15979-4_19
23. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186-194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
24. Kabatianskii, G., Krouk, E., Smeets, B.: A digital signature scheme based on random error-correcting codes. In: Darnell, M. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 161-167. Springer, Heidelberg (1997). https://doi.org/10.1007/ BFb0024461
25. Lau, T.S.C., Tan, C.H., Prabowo, T.F.: Key recovery attacks on some rank metric code-based signatures. In: Albrecht, M. (ed.) IMACC 2019. LNCS, vol. 11929, pp. 215-235. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-35199-1_11
26. Lee, P.J., Brickell, E.F.: An observation on the security of McEliece's public-key cryptosystem. In: Barstow, D., et al. (eds.) EUROCRYPT 1988. LNCS, vol. 330, pp. 275-280. Springer, Heidelberg (1988). https://doi.org/10.1007/3-540-45961-8_25
27. Leon, J.: A probabilistic algorithm for computing minimum weights of large errorcorrecting codes. In: IEEE Trans. Inform. Theory 34.5 (1988), pp. 1354-1359
28. May, A., Meurer, A., Thomae, E.: Decoding random linear codes in $\tilde{O}\left(2^{0.054 n}\right)$. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 107-124. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_6
29. May, A., Ozerov, I.: On computing nearest neighbors with applications to decoding of binary linear codes. In: Oswald, E., Fischlin, Marc (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 203-228. Springer, Heidelberg (2015). https://doi.org/10.1007/ 978-3-662-46800-5_9
30. McEliece, R.: A public-key cryptosystem based on algebraic coding theory. Jet Propulsion Lab., Pasadena, CA, USA, DSN Progress Rep. 42-44, pp. 114-116, 1978.
31. Meneghetti, A., Picozzi, C., Tognolini, G.: A Post-Quantum Digital Signature Scheme from QC-LDPC Codes. IACR Cryptology ePrint Archive 2022/1477 (2022). https://eprint.iacr.org/2022/1477
32. Sendrier, N.: Wave Parameter Selection. IACR Cryptology ePrint Archive 2023/588 (2023). https://eprint.iacr.org/2023/588
33. Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. Probl. Control Inf. Theory, vol. 15, no. 2, pp. 159-166, 1986.
34. Otmani, A., Tillich, J. P.: An efficient attack on all concrete KKS proposals. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 98-116. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25405-5_7
35. Prabowo, T.F., Tan, C.H.: Attack on a Code-Based Signature Scheme from QCLDPC Codes. In: El Hajji, S., Mesnager, S., Souidi, E.M. (eds.) Codes, Cryptology and Information Security (C2SI 2023). LNCS, vol. 13874, pp. 136-149. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-33017-9_9
36. Prange, E.: The use of information sets in decoding cyclic codes, IRE Transactions on Information Theory, Vol. 8, No. 5, pp. 5-9, 1962.
37. Ren, F., Zheng, D., Wang, W.: An efficient code based digital signature algorithm. Int. J. Netw. Secur., 19(6):1072-1079 (2017). https://doi.org/10.6633/IJNS. 201711. 19(6). 24
38. Sendrier, N.: Decoding one out of many. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 51-67. Springer, Heidelberg (2011). https://doi.org/10.1007/ 978-3-642-25405-5_4
 from the lyubashevsky framework, Theoretical Computer Science 835 (2020) 15-30. https://doi.org/10.1016/j.tcs.2020.05.011
39. Stern, J.: A method for finding codewords of small weight. In: Cohen, G., Wolfmann, J. (eds.) Coding Theory 1988. LNCS, vol. 388, pp. 106-113. Springer, Heidelberg (1989). https://doi.org/10.1007/BFb0019850
40. Stern, J.: A new identification scheme based on syndrome decoding. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 13-21. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48329-2_2
41. Canto Torres, R., Sendrier, N.: Analysis of information set decoding for a sub-linear error weight. In: Takagi, T. (ed.) PQCrypto 2016. LNCS, vol. 9606, pp. 144-161. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29360-8_10
42. Vasseur, V.: Post-quantum cryptography: a study of the decoding of QC-MDPC codes. Ph.D. thesis, Université de Paris (2021)

# Efficient Result-Hiding Searchable Encryption with Forward and Backward Privacy 

Takumi Amada ${ }^{1}$, Mitsugu Iwamoto ${ }^{1}$, and Yohei Watanabe ${ }^{1,2}$<br>1 The University of Electro-Communications, Tokyo, Japan<br>\{t.amada, mitsugu, watanabe\}@uec.ac.jp<br>${ }^{2}$ National Institute of Information and Communications Technology, Tokyo, Japan.


#### Abstract

Dynamic searchable symmetric encryption (SSE) realizes efficient update and search operations for encrypted databases, and there has been an increase in this line of research in the recent decade. Dynamic SSE allows the leakage of insignificant information to ensure efficient search operations, and it is important to understand and identify what kinds of information are insignificant. In this paper, we propose an efficient dynamic SSE scheme Laura under the small leakage, which leads to appealing security requirements such as forward privacy, (TypeII) backward privacy, and result hiding. Laura is constructed based on Aura (NDSS 2021) and is almost as efficient as Aura while only allowing less leakage than Aura. We also provide experimental results to show the concrete efficiency of Laura.


Keywords: Dynamic searchable encryption • Backward Privacy • Encrypted database.

## 1 Introduction

Searchable symmetric encryption (SSE) [12, 24] provides a way to search a large database efficiently (e.g., cloud storage) for encrypted data. In particular, SSE that supports update operations is called dynamic SSE [20], which has attracted attention over the past decade $[10,16,19,20,22,23]$.

Forward and Backward Privacy. Dynamic SSE aims to efficiently perform keyword searches on encrypted data while revealing some insignificant information to the server. A common understanding of what kinds of leakage are insignificant has been updated by exploring leakage-abuse attacks $[5,9,17$, 28] against SSE. In particular, file injection attacks demonstrated by Zhang et al. [28] showed that forward privacy [7], which guarantees that the adversary cannot learn if newly-added files contain previously-searched keywords, must be a de facto standard security requirement for dynamic SSE.

Backward privacy [8], which guarantees that search operations reveal no useful information on previously-deleted files even if they contain searched keywords, has been spotlighted since it sounds like another natural security requirement. However, it is more difficult to achieve backward privacy than forward privacy
since it is just like we require the server to forget previously-stored information. For example, we have to hide even information about when and which files are added and/or deleted to meet backward privacy. Therefore, one of the current major research interests in dynamic SSE is how efficiently we construct dynamic SSE schemes with backward privacy.

Importance of Result-Hiding SSE. As described above, leakage-abuse attacks tell us which information should not be leaked during update and search operations. Existing attacks are classified into passive and active ones. Passive attacks (e.g., [17]) aim to identify keywords behind search queries from admitted leakage information and seem more likely to happen in the real world than active attacks (e.g., [28]), which require that the server can force the client to upload arbitrary files. A major drawback of passive attacks is that they also require partial information of the stored data as extra information in addition to the leakage profiles. This is quite an unrealistic assumption. Hence, the subsequent works (e.g., $[5,9]$ ) have attempted to weaken the assumption. Recently, Bkackstone et al. [5] showed passive attacks that only require $5 \%$ of the client's data, whereas the Islam et al.'s seminal work [17] requires at least $95 \%$ of the client's data. In particular, it is worth noting that their attacks only use access pattern leakage, which is a standard leakage profile of dynamic SSE. Although there are, fortunately, countermeasures such as volume-hiding techniques [18], they significantly decrease the efficiency of dynamic SSE schemes. Thus, it becomes more important to seek efficient constructions of result-hiding schemes, which are dynamic SSE schemes mitigating access pattern leakage.

### 1.1 Our Contribution

In this paper, we propose Laura, a new result-hiding dynamic SSE scheme with forward and Type-II backward privacy, which is the most investigated security level of backward privacy. Laura is constructed based on Aura [25]; Laura is built from only symmetric-key primitives, specifically, from any pseudorandom function (PRF), any symmetric-key encryption (SKE), and any approximate membership query (AMQ) data structure. Laura achieves better practical efficiency to Aura and requires less leakage than Aura; this is the reason why we call our scheme Laura, which stands for Low-leakage Aura.

We give experimental results to show the concrete efficiency of Laura and v-Laura, which is a variant of Laura; their deletion and search procedures are almost as efficient as Aura, and their addition procedures are substantially more efficient than Aura. For example, Laura and v-Laura take less than a second to add 200,000 entries, while Aura takes about a minute. For concrete efficiency comparison among Laura, v-Laura, and Aura, see Section 6.

As a side result, we also figure out that in Aura (as well as Laura and v-Laura), the client is assumed to never re-add any keyword-identifier pair ( $w$, id) once deleted, where id is a file identifier. This assumption seems reasonable in practice since id should be replaced with a new one if the client wants to re-add a previously-deleted file whose identifier is id. We also show a variant of Laura,

Table 1: Efficiency comparison among Type-II backward-private dynamic SSE schemes. Suppose that the client has performed search and update operations $t$ times in total. $d$ and $n$ are the total numbers of distinct keywords and files, respectively. $a_{w}, n_{w}$, and $n_{w, \text { del }}^{(\operatorname{srch})}$ are the total numbers of all updates for $w$, files currently containing a keyword $w$, and times a keyword $w$ has been affected by search operations since the last search for $w$, respectively. It clearly holds $a_{w} \geq \widehat{n}_{w} \geq n_{w}$, where $\widehat{n}_{w}:=n_{w}+n_{w, \text { del }}^{(\text {srch })} . N$ is the total numbers of (document, keyword) pairs, i.e., $N:=\Sigma_{w} n_{w}$. Let $N^{\prime}:=\Sigma_{w} \widehat{n}_{w}$ and $\widehat{N}:=\Sigma_{w} a_{w}$. Namely, it holds $\widehat{N} \geq N^{\prime} \geq N .|\sigma|$ and $|E D B|$ denote bit-lengths of client's state information and encrypted database. RT and RH stand for roundtrips and result hiding, respectively. SK indicates whether the scheme is constructed from only symmetrickey primitives. RU stands for re-updatability, which allows the client to re-add a previously-deleted entry ( $w$, id) to EDB.

$\dagger$ Amortized analysis.
${ }^{\ddagger}$ To be precise, the deletion procedure depends on the time complexity of the underlying AMQ structure, which is $\mathcal{O}(1)$ in almost all existing constructions.
${ }^{\#}$ Let $\widehat{a}_{w}:=a_{w}+\log \widehat{N}$ and $\widehat{n}_{w, \text { del }}^{(\mathrm{srch})}:=\widehat{n}_{w} \cdot n_{w, \text { del }}^{(\mathrm{srch})}$ for compact notations.
called s-Laura, that removes the assumption, i.e., it allows the client to re-add previous-deleted entries to the encrypted database, although s-Laura requires extra search costs.

Efficiency Comparison. We compare the asymptotic efficiency of dynamic SSE schemes with forward and Type-II backward privacy in Table 1. Note that we evaluate the server-side complexities of update and search algorithms. Although the efficiency of Laura and v-Laura seems comparable to Fides [8] and Aura [25], Laura and v-Laura has clear advantages over them; Fides employs public-key primitives such as trapdoor permutations for its building block. Moreover, Fides returns a (tentative) search result that contains deleted identifiers. Therefore, the client themself has to remove such deleted ones to obtain the correct search result. Although Laura and v-Laura also require for the client to remove deleted identifiers, the client can easily find them thanks to the underlying approximate
membership query (AMQ) data structure. Aura achieves the minimum roundtrip, however, the size of encrypted databases is large. Furthermore, Aura reveals the access pattern and therefore is not a result-hiding scheme. Among the dynamic SSE schemes that satisfy all properties (RH, SK, and RU) listed in the table, $s$-Laura is more efficient than $\mathrm{SD}_{a}$ and $\mathrm{SD}_{d}$.

## 2 Preliminaries

Notations. For any integer $a \in \mathbb{Z}$, let $[a]:=\{1,2, \ldots, a\}$. For a finite set $\mathcal{X}$, we use $x \stackrel{\&}{\leftarrow} \mathcal{X}$ to represent processes of choosing an element $x$ from $X$ uniformly at random. For a finite set $\mathcal{X}$, we denote by $\mathcal{X} \leftarrow x$ and $|\mathcal{X}|$ the addition $x$ to $\mathcal{X}$ and cardinality of $\mathcal{X}$, respectively. Concatenation is denoted by $\|$. In the description of the algorithm, all arrays, strings, and sets are initialized to empty ones. We consider probabilistic polynomial time (PPT) algorithms. For any non-interactive algorithm A , out $\leftarrow \mathrm{A}$ (in) means that A takes in as input and outputs out. In this paper, we consider two-party interactive algorithms between a client and a server, and it is denoted by (out ${ }_{C}$; out $\left.{ }_{S}\right) \leftarrow A\left(\mathrm{in}_{\mathrm{C}}\right.$; in ${ }_{\mathrm{S}}$ ), where in $\mathrm{in}_{\mathrm{C}}$ and ins are input of client and server, respectively and outc and outs are output of client and server, respectively. If necessary, we mention the transcript trans and describe the algorithm as $\left\langle\left(\right.\right.$ out $_{\mathrm{c}}$; outs), trans $\rangle \leftarrow A$ (inc; ins). The security parameter and negligible function are denoted by $\kappa$ and negl $(\cdot)$, respectively.

Pseudorandom Functions (PRFs). A family of functions $\pi:=\left\{\pi_{\mathrm{k}_{\text {PRE }}}:\{0,1\}^{*}\right.$ $\left.\rightarrow\{0,1\}^{m}\right\}_{\mathrm{k}_{\text {PRF }} \in\{0,1\}^{\kappa}}$, where $m=\operatorname{poly}(\kappa)$, is said to be a (variable-input-length) PRF family if for sufficiently large $\kappa \in \mathbb{N}$ and all PPT algorithm D , it holds $\left|\operatorname{Pr}\left[\mathrm{D}^{\pi\left(\mathrm{k}_{\mathrm{PRF}}, \cdot\right)}\left(1^{\kappa}\right)=1 \mid \mathrm{k}_{\mathrm{PRF}} \stackrel{\leftarrow}{\leftarrow}\{0,1\}^{\kappa}\right]-\operatorname{Pr}\left[\mathrm{D}^{\mathrm{R}(\cdot)}\left(1^{\kappa}\right)=1 \mid \mathrm{R} \stackrel{\S}{\leftarrow} \mathcal{R}\right]\right|<\operatorname{neg}(\kappa)$, where $\mathcal{R}$ is a set of all mappings $R:\{0,1\}^{*} \rightarrow\{0,1\}^{m}$.

Symmetric-Key Encryption (SKE). An SKE $\Pi_{\text {SKE }}$ consists of three PPT algorithms $\Pi_{\mathrm{SKE}}=(\mathrm{G}, \mathrm{E}, \mathrm{D}) . \mathrm{G}$ takes a security parameter $\kappa$ as input and outputs a secret key $\mathrm{k}_{\text {SKE }}$, and E takes a plaintext m and $\mathrm{k}_{\mathrm{SKE}}$ as input and outputs the ciphertext c. D takes c with $\mathrm{k}_{\mathrm{SKE}}$ and outputs m or $\perp$ as a symbol of failure. In this paper, we assume $\Pi_{\text {SKE }}$ is CPA security. For formal definitions, we refer the readers to [21]. Also, if necessary, we explicitly describe a nonce used in an SKE. Specifically, for nonce $r$, the encryption and decryption algorithms are denoted by $\mathrm{E}\left(\mathrm{k}_{\text {SKE }}, \mathrm{m} ; r\right)$ and $\mathrm{D}\left(\mathrm{k}_{\mathrm{SKE}}, \mathrm{c} ; r\right)$, respectively. The ciphertext is treated as $r \| \mathrm{c}$. Note that (nonce-based) CTR and CBC modes in block ciphers satisfy CPA security and the above properties.

Approximate Membership Query (AMQ) Structure. Probabilistic data structures, known as Approximate Membership Query (AMQ) data structures, provide membership queries with compact data sizes by allowing "false positives." The most appealing feature of AMQ structures is to make the falsepositive probability small enough by setting specific parameters appropriately. We consider AMQ structures that support both insertion and deletion operations. While the Bloom filter [6], one of the well-known AMQ structures, does
not support deletion, recent ones, such as the cuckoo filter [15] and quotient filter [4], do. Formally, an arbitrary set $\mathcal{U} \in\{0,1\}^{*}$, an AMQ data structure $\Pi_{\mathrm{AMQ}}=(\mathrm{AMQ} . G e n, \mathrm{AMQ}$. Insert, AMQ .Delete, AMQ .Lookup) consists of the following PPT algorithms:
$-(\mathcal{T}$, aux $) \leftarrow$ AMQ. $\operatorname{Gen}(\mathcal{U}$, par $)$ : it takes $\mathcal{U}$ and a parameter par as input, and outputs an initial structure $\mathcal{T}$ and auxiliary information aux. The parameter par depends on the construction of the specific AMQ structure.
$-\mathcal{T}^{\prime} \leftarrow \mathrm{AMQ}$. Insert $(\mathcal{T}, x$, aux): it takes as input a data structure $\mathcal{T}$, an element $x \in \mathcal{U}$ to be added, and aux, and outputs an updated structure $\mathcal{T}^{\prime}$.
$-\mathcal{T}^{\prime} \leftarrow$ AMQ. Delete $(\mathcal{T}, x$, aux $)$ : it takes as input a data structure $\mathcal{T}$, an element $x \in \mathcal{U}$ to be deleted, and aux, and outputs an updated structure $\mathcal{T}^{\prime}$.

- true/false $\leftarrow$ AMQ.Lookup $(\mathcal{T}, x$, aux): it takes as input a data structure $\mathcal{T}$, an element $x \in \mathcal{U}$ to be queried, and aux, and outputs true or false.

AMQ structures meet the following two properties. Due to the page limitation, we omit the formal description and will give it in the full version.

- Completeness: Let $\mathcal{S}$ be a set of elements that have been inserted (and not deleted). For all $x \in \mathcal{S}$, it holds AMQ.Lookup $(\mathcal{T}, x$, aux $)=$ true, where $\mathcal{T}$ is the corresponding structure.
- Bounded False-Positive Probability: Let $n:=|\mathcal{S}|$. Then, there exists $\mu_{n} \in$ $(0,1]$ such that it holds $\operatorname{Pr}[\mathrm{AMQ} \cdot \operatorname{Lookup}(\mathcal{T}, x$, aux $)=$ true $] \leq \mu_{n}$ for any $x \in \mathcal{U} \backslash \mathcal{S}$.


## 3 Dynamic SSE

### 3.1 Notation for Dynamic SSE

$\Lambda:=\{0,1\}^{\lambda}$ is a set of possible keywords (sometimes called a dictionary), where $\lambda=\operatorname{poly}(\kappa)$. A document $f_{\text {id }}$ has its unique identifier id $\in\{0,1\}^{\ell}$, which is irrelevant to the contents of $f_{\text {id }}$, where $\ell=\operatorname{poly}(\kappa)$. A counter $t$ represents the global counter through the protocol; it is initialized to 0 at setup and incremented for each search or update operation. A database $\mathrm{DB}^{(t)}$ at $t$ is represented as a set of keyword-identifier pairs $(w$, id $)$, i.e., $\mathrm{DB}^{(t)}:=\left\{\left(w_{i}, \mathrm{id}_{i}\right)\right\}_{i=1}^{N(t)}$, where $N(t)$ is the number of pairs stored in the server at $t$. We denote $\mathrm{ID}^{(t)}$ by a set of identifiers in $\mathrm{DB}^{(t)}$. That is, $\mathrm{ID}^{(t)}:=\left\{\right.$ id $\mid \forall w \in \Lambda,(w$, id $\left.) \in \mathrm{DB}^{(t)}\right\}$. Similarly, $\mathcal{W}^{(t)}$ is denoted by a set of keywords in $\mathrm{DB}^{(t)}$, i.e., $\mathcal{W}^{(t)}:=\left\{w \mid \forall \mathrm{id} \in \mathrm{ID}^{(t)},(w, \mathrm{id}) \in \mathrm{DB}^{(t)}\right\}$.

### 3.2 Model

Dynamic SSE consists of three PPT algorithms (Setup, Update, Search). Firstly, the client runs Setup to generate a secret key, initial state information, and an initial encrypted database, which is sent to the server. The client interacts with the server and runs Update and Search repeatedly to add or delete a pair ( $w$, id) and search for keywords.

Definition 1 (Dynamic SSE). A Dynamic SSE $\Sigma:=$ (Setup, Update, Search) over $\Lambda$ consists of the following PPT algorithms:
$-\left(k, \sigma^{(0)}, \mathrm{EDB}^{(0)}\right) \leftarrow \operatorname{Setup}\left(1^{\kappa}\right):$ it is an non-interactive algorithm that takes a security parameter $\kappa$ as input and outputs a secret key $k$, initial state information $\sigma^{(0)}$, and initial encrypted database $\mathrm{EDB}^{(0)}$.
$-\left(\sigma^{(t+1)} ; \mathrm{EDB}^{(t+1)}\right) \leftarrow$ Update $\left(k, \mathrm{op}\right.$, in, $\left.\sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)$ : it is an interactive algorithm that takes $k$, an operation label $\mathrm{op} \in\{$ add, del\}, the corresponding input in $:=(w, \mathrm{id})$, and $\sigma^{(t)}$ as input of the client and encrypted database $\mathrm{EDB}^{(t)}$ as input of the server, and outputs updated state information $\sigma^{(t+1)}$ for the client and updated encrypted database $\mathrm{EDB}^{(t+1)}$ for the server.
$-\left(\mathcal{X}_{q}^{(t)}, \sigma^{(t+1)} ; \mathrm{EDB}^{(t+1)}\right) \leftarrow \operatorname{Search}\left(k, q, \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right):$ it is an interactive algorithm that takes $k$, a searched keyword $q$, and $\sigma^{(t)}$ as input of the client and encrypted database $\mathrm{EDB}^{(t)}$ as input of the server, and outputs updated state information $\sigma^{(t+1)}$ and a search result $\mathcal{X}_{q}^{(t)}$ for the client and updated encrypted database $\operatorname{EDB}^{(t+1)}$ for the server.

Briefly, the correctness of the above model ensures that it holds $\mathcal{X}_{q}^{(t)}=\{$ id $\in$ $\mathrm{ID}^{(t)} \mid(q$, id $\left.) \in \mathrm{DB}^{(t)}\right\}$ with overwhelming probability for any keyword $q \in \Lambda$. For a formal definition, we refer the readers to [10].

### 3.3 Security

Dynamic SSE guarantees that the (honest-but-curious) server does not learn any information beyond some explicit information leakage during a sequence of operations. Therefore, such information leakage is characterized as a leakage function $\mathcal{L}:=\left(\mathcal{L}_{\text {Setup }}, \mathcal{L}_{\text {Upd }}, \mathcal{L}_{\text {Srch }}\right)$, where $\mathcal{L}_{\text {Setup }}, \mathcal{L}_{\text {Upd }}$, and $\mathcal{L}_{\text {Srch }}$ are functions that refer to information leaked during Setup, Update, and Search, respectively.
$\mathcal{L}$-Adaptive Security. We define $\mathcal{L}$-adaptive security of SSE in a simulationbased manner. We consider two experiments: a real experiment Real in which the Dynamic SSE scheme is performed in the real world and an ideal experiment Ideal that at most leaks a leakage function $\mathcal{L}$. Specifically, a real experiment Real ${ }_{D}$ is performed by the client and a PPT algorithm $\mathrm{D}=\left(\mathrm{D}_{1}, \mathrm{D}_{2}, \ldots, \mathrm{D}_{Q+1}\right)$, while ideal experiment Ideal ${ }_{D, S, \mathcal{L}}$ is performed by $D$ and a simulator $S=\left(S_{0}, S_{1}, \ldots, S_{Q}\right)$ with leakage function $\mathcal{L}$. In each experiment, D adaptively queries and attempts to distinguish between the two experiments. If D cannot distinguish between them, D has not learned more information than the leakage function $\mathcal{L}$; we call this $\mathcal{L}$-adaptive security. Each experiment is formally given in Fig. 1, and the security definition is as follows [27].

Definition 2 ( $\mathcal{L}$-Adaptive Security). Let $\Sigma$ be a Dynamic SSE scheme. $\Sigma$ is $\mathcal{L}$-adaptively secure, with regard to a leakage function $\mathcal{L}$, if for any PPT algorithm D , any sufficiently large $\kappa \in \mathbb{N}$, and any $Q:=\operatorname{poly}(\kappa)$, there exists a PPT algorithm S s.t. $\mid \operatorname{Pr}[\operatorname{Real}(\kappa, Q)=1]-\operatorname{Pr}\left[\right.$ Ideal $\left._{\mathrm{D}, \mathrm{s}, \mathcal{L}}(\kappa, Q)=1\right] \mid<\operatorname{negl}(\kappa)$.

```
Real Experiment: \(\operatorname{Real}_{\mathrm{D}}(\kappa, Q)\)
    \(\left(k, \sigma^{(0)}, \operatorname{EDB}^{(0)}\right) \leftarrow \operatorname{Setup}\left(1^{\kappa}\right)\)
    \(s t_{D}:=\left\{\mathrm{EDB}^{(0)}\right\}\)
    for \(t=1\) to \(Q\) do
        query \(\leftarrow \mathrm{D}_{t}\left(\mathrm{st}_{\mathrm{D}}\right)\)
        if query \(=(\mathrm{upd}, \mathrm{op}\), in) then
            \(\left\langle\left(\sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)\right.\), trans \(\left.^{(t)}\right\rangle\)
            \(\leftarrow \operatorname{Update}\left(k\right.\), op, in, \(\left.\sigma^{(t-1)} ; \mathrm{EDB}^{(t-1)}\right)\)
        if query \(=(\operatorname{srch}, q)\) then
            \(\left\langle\left(\sigma^{(t)}, \mathcal{X}_{q}^{(t-1)} ; \mathrm{EDB}^{(t)}\right), \operatorname{trans}^{(t)}\right\rangle\)
            \(\leftarrow \operatorname{Search}\left(k, q, \sigma^{(t-1)} ; \mathrm{EDB}^{(t-1)}\right)\)
        \(\mathrm{st}_{\mathrm{D}} \leftarrow\left(\mathrm{EDB}^{(t)}\right.\), trans \(\left.^{(t)}\right)\)
    \(b \leftarrow \mathrm{D}_{Q+1}\left(\mathrm{st}_{\mathrm{D}}\right)\)
    return \(b\)
```

```
Ideal Experiment: Ideal \({ }_{\mathrm{D}, \mathrm{S}, \mathcal{L}}(\kappa, Q)\)
```

Ideal Experiment: Ideal ${ }_{\mathrm{D}, \mathrm{S}, \mathcal{L}}(\kappa, Q)$
$\left(\operatorname{EDB}^{(0)}\right.$, sts $\left._{\mathrm{s}}\right) \leftarrow \mathrm{S}_{0}\left(\mathcal{L}_{\text {Setup }}(\kappa)\right)$
$\left(\operatorname{EDB}^{(0)}\right.$, sts $\left._{\mathrm{s}}\right) \leftarrow \mathrm{S}_{0}\left(\mathcal{L}_{\text {Setup }}(\kappa)\right)$
$\mathrm{st}_{\mathrm{D}}:=\left\{\mathrm{EDB}^{(0)}\right\}$
$\mathrm{st}_{\mathrm{D}}:=\left\{\mathrm{EDB}^{(0)}\right\}$
for $t=1$ to $Q$ do
for $t=1$ to $Q$ do
query $\leftarrow \mathrm{D}_{t}\left(\mathrm{st}_{\mathrm{D}}\right)$
query $\leftarrow \mathrm{D}_{t}\left(\mathrm{st}_{\mathrm{D}}\right)$
if query $=(u p d, o p$, in) then
if query $=(u p d, o p$, in) then
$\left\langle\left(\mathrm{st}_{\mathrm{S}}^{\prime} ; \mathrm{EDB}^{(t)}\right)\right.$, trans $\left.^{(t)}\right\rangle$
$\left\langle\left(\mathrm{st}_{\mathrm{S}}^{\prime} ; \mathrm{EDB}^{(t)}\right)\right.$, trans $\left.^{(t)}\right\rangle$
$\leftarrow \mathrm{S}_{t}\left(\mathrm{st}_{\mathrm{s}}, \mathcal{L}_{\mathrm{Upd}}(t\right.$, op, in $\left.) ; \mathrm{EDB}^{(t-1)}\right)$
$\leftarrow \mathrm{S}_{t}\left(\mathrm{st}_{\mathrm{s}}, \mathcal{L}_{\mathrm{Upd}}(t\right.$, op, in $\left.) ; \mathrm{EDB}^{(t-1)}\right)$
if query $=(\operatorname{srch}, q)$ then
if query $=(\operatorname{srch}, q)$ then
$\left\langle\left(\right.\right.$ st $\left._{S}^{\prime} ; \mathrm{EDB}^{(t)}\right)$, trans $\left.^{(t)}\right\rangle$
$\left\langle\left(\right.\right.$ st $\left._{S}^{\prime} ; \mathrm{EDB}^{(t)}\right)$, trans $\left.^{(t)}\right\rangle$
$\leftarrow \mathrm{S}_{t}\left(\mathrm{sts}_{\mathrm{s}}, \mathcal{L}_{\mathrm{Srch}}(t, q) ; \mathrm{EDB}^{(t-1)}\right)$
$\leftarrow \mathrm{S}_{t}\left(\mathrm{sts}_{\mathrm{s}}, \mathcal{L}_{\mathrm{Srch}}(t, q) ; \mathrm{EDB}^{(t-1)}\right)$
$\mathrm{st}_{\mathrm{D}} \leftarrow\left(\mathrm{EDB}^{(t)}\right.$, trans $\left.^{(t)}\right)$
$\mathrm{st}_{\mathrm{D}} \leftarrow\left(\mathrm{EDB}^{(t)}\right.$, trans $\left.^{(t)}\right)$
$\mathrm{st}_{\mathrm{S}}:=\mathrm{st}_{\mathrm{S}}^{\prime}$
$\mathrm{st}_{\mathrm{S}}:=\mathrm{st}_{\mathrm{S}}^{\prime}$
$b \leftarrow \mathrm{D}_{Q+1}\left(\mathrm{st} \mathrm{t}_{\mathrm{D}}\right)$
$b \leftarrow \mathrm{D}_{Q+1}\left(\mathrm{st} \mathrm{t}_{\mathrm{D}}\right)$
return $b$

```
    return \(b\)
```

Fig. 1: Real and ideal experiments.

Forward and Backward Privacy. The well-known security notions for update operations are forward privacy [7] and backward privacy [8].

Forward privacy, roughly speaking, ensures that while running an update of a keyword-identifier pair ( $q$, id), no information about the keyword $q$ is exposed to the server. This means that the keyword $q$ cannot be associated with all previous searches and update operations. Forward privacy is an important security requirement since Zhang et al. [28] showed effective attacks against non-forwardprivate dynamic SSE schemes. The formal definition is as follows:

Definition 3 (Forward Privacy [7]). Let $\Sigma$ be a $\mathcal{L}$-adaptively secure dynamic SSE scheme. $\Sigma$ is forward private if $\mathcal{L}_{\text {Upd }}($ for $\mathrm{op}=$ add) can be written as $\mathcal{L}_{\mathrm{Upd}}(t$, add,$(q, \mathrm{id}))=\mathcal{L}^{\prime}(t$, add, id$)$, where $\mathcal{L}^{\prime}$ is stateless function.

On the other hand, loosely speaking, backward privacy guarantees that while running a search for a keyword $q$, the least possible (ideally, no) information about the deleted pair ( $q$, id) is leaked to the server. However, if leakage regarding deletion operations is to be completely eliminated, significant costs are required due to efficiency trade-offs. Therefore, Bost et al. [8] introduced three levels of backward privacy: from Type-I with the least leakage to Type-III with the most leakage. In this paper, we focus on Type-II backward privacy, which achieves a good balance between security levels and achievable efficiency. To describe their definition, we define several functions of leaked information as follows. Let $\mathcal{Q}^{(t)}$ be the set of all operations of each counter $u \in[t]$, and its elements are described as $(u, q) \in \mathcal{Q}^{(t)}$ for a search for a keyword $q$ and $(u, \mathrm{op},(q$, id $\left.))\right) \in \mathcal{Q}^{(t)}$ for an update of a keyword-identifier pair ( $q$, id), where op $\in\{$ add, del $\}$.

- Search pattern $\mathrm{SP}_{q}^{(t)}$ : A set of counters at which the keyword $q$ has been searched. That is, $\mathrm{SP}_{q}^{(t)}:=\left\{u \mid(u, q) \in \mathcal{Q}^{(t)}\right\}$.
- Access pattern $\operatorname{TimeDB}_{q}^{(t)}: \mathrm{A}$ set of pairs of an identifier id $\in \mathrm{ID}^{(t)}$ that includes a keyword $q$ at $t$ and a counter $u$ when the corresponding keywordidentifier pair ( $q$, id) was added. That is,

$$
\operatorname{TimeDB}_{q}^{(t)}:=\left\{\begin{array}{l|l}
\left(u^{\text {add }}, \text { id }\right) & \begin{array}{l}
\left(u^{\text {add }}, \text { add },(q, \text { id })\right) \in \mathcal{Q}^{(t)} \\
\wedge \forall u^{\text {del }},\left(u^{\text {del }}, \text { del },(q, \text { id })\right)
\end{array} \notin \mathcal{Q}^{(t)}
\end{array}\right\},
$$

where we assume $u^{\text {add }}<u^{\text {del }}$ without the loss of generality.

- Update pattern Update ${ }_{q}^{(t)}$ : It is a set of counters for all update operations on $q$, i.e., Update ${ }_{q}^{(t)}:=\left\{u \mid(u\right.$, add,$(q$, id $)) \in \mathcal{Q}^{(t)} \vee(u$, del, $(q$, id $\left.)) \in \mathcal{Q}^{(t)}\right\}$.

Using the above functions, Type-II backward privacy is defined as follows.
Definition 4 (Type-II Backward Privacy [8]). Let $\Sigma$ be a $\mathcal{L}$-adaptively secure dynamic SSE scheme. $\Sigma$ is Type-II backward private if $\mathcal{L}_{\text {Upd }}$ and $\mathcal{L}_{\text {Srch }}$ can be written as:
$\mathcal{L}_{\text {Upd }}(t$, op,$(q$, id $))=\mathcal{L}^{\prime}(t$, op,$q)$ and $\mathcal{L}_{\text {Srch }}(t, q)=\mathcal{L}^{\prime \prime}\left(\mathrm{SP}_{q}^{(t)}, \operatorname{TimeDB}_{q}^{(t)}\right.$, Update $\left._{q}^{(t)}\right)$, where $\mathcal{L}^{\prime}$ and $\mathcal{L}^{\prime \prime}$ are stateless functions.

Result Hiding. As mentioned in the introduction, taking into account the recent progress in leakage abuse attacks, it is important to realize an efficient dynamic SSE scheme that never leaks identifiers of search results. Such a scheme is called a result-hiding one. Although several result-hiding schemes [13, 8] are already known, to the best of our knowledge, there is no formal definition of the result-hiding property. Therefore, we first define it formally. We consider the following leakage functions.

- Concealed access pattern $\operatorname{Time}_{q}^{(t)}$ : It is a set of counters contained in TimeDB ${ }_{q}^{(t)}$. That is, $\operatorname{Time}_{q}^{(t)}:=\left\{u \mid \exists\right.$ id s.t. $(u$, id $\left.) \in \operatorname{TimeDB}_{q}^{(t)}\right\}$.
- Deletion history DelHist ${ }_{q}^{(t)}$ : It is a set of pairs of two counters at which each of addition and deletion operations is performed on the same ( $q$, id) pair. That is,

$$
\text { DelHist } \left._{q}^{(t)}:=\left\{\begin{array}{l|l}
\left(u^{\text {add }}, u^{\text {del }}\right)
\end{array}\right) \begin{array}{l}
\exists \text { id s.t. }\left(u^{\text {add }}, \text { add },(q, \text { id })\right) \in \mathcal{Q}^{(t)} \\
\wedge\left(u^{\text {del }}, \text { del },(q, \text { id })\right) \in \mathcal{Q}^{(t)}
\end{array}\right\}
$$

Although DelHist ${ }_{q}^{(t)}$ is a well-known leakage function to define Type-III backward privacy, we also use it to define the result-hiding property.

Definition 5 (Result-Hiding Dynamic SSE). Let $\Sigma$ be a $\mathcal{L}$-adaptively secure dynamic SSE scheme. $\Sigma$ is called a result-hiding scheme if $\mathcal{L}_{\text {Upd }}$ and $\mathcal{L}_{\text {Srch }}$ can be written as:

$$
\mathcal{L}_{\text {Upd }}(t, \mathrm{op},(q, \text { id }))=\mathcal{L}^{\prime}(t, \text { op }, q) \text { and } \mathcal{L}_{\text {Srch }}(t, q)=\mathcal{L}^{\prime \prime}\left(\mathrm{SP}_{q}^{(t)}, \operatorname{Time}_{q}^{(t)}, \operatorname{DelHist}_{q}^{(t)}\right),
$$ where $\mathcal{L}^{\prime}$ and $\mathcal{L}^{\prime \prime}$ are stateless functions.

Namely, result-hiding schemes do not leak any identifiers during updates and searches. Note that the search operation may leak all information related to the counters of update operations on $q$ since the result-hiding property should be a property in which result-hiding schemes reveal no information on identifiers themselves contained in search results.

Remark 1. One may think that the result-hiding property conflicts with a common use case of dynamic SSE, where the server returns both a search result and the corresponding actual documents. The property prevents the server from returning the actual documents unless the client reveals the search result to the server; the reveal means the leakage of the access pattern and makes the result-hiding property meaningless! Nevertheless, in such a common use case, the result-hiding property should be valuable since the client can choose whether the client reveals the access pattern. Of course, the property would be more appealing in other use cases, e.g., where actual documents are stored on another server.

## 4 Laura: Low-leakage Aura

We propose a new efficient dynamic SSE scheme that meets forward privacy, Type-II backward privacy, and the result-hiding property. Although the construction approach of our scheme is based on Aura, our scheme allows less leakage than Aura. Thus, we call our scheme Laura, which stands for low-leakage Aura.

### 4.1 Construction Idea

Construction Overview of Aura. Sun et al. [25] introduced a core building block of Aura, called symmetric revocable encryption (SRE). Briefly speaking, SRE supports puncturable decryption. In SRE, plaintexts are encrypted along with a tag. A decryption key associated with a certain revoked set, containing revoked tags, cannot decrypt ciphertexts related to the revoked tags. In Aura, SRE's puncturable decryption functionality allows the server to decrypt ciphertexts without leaking deleted entires as follows. When adding ( $w$, id), the client encrypts id with a tag $\tau$ and the ciphertext is stored on the server. When deleting ( $w$, id), the client adds the corresponding tag $\tau$ to a revoked tag set $\mathcal{R}_{w}$ on $w$, stored in the local storage. When searching for $w$, the client retrieves the revoked tag set $\mathcal{R}_{w}$ and generates a decryption key associated with $\mathcal{R}_{w}$. The server decrypts ciphertexts with the key and obtains id if the corresponding tag $\tau^{\prime}$ is not revoked (i.e., $\tau^{\prime} \notin \mathcal{R}_{w}$ ); it obtains $\perp$ otherwise due to the puncturable decryption functionality. Therefore, the client can delegate the process of removing deleted entries to the server; it does not leak when and which identities have been deleted. The client just receives and outputs the search result from the server. Consequently, Aura is the first (efficient) dynamic SSE that supports both non-interactive search operations and Type-II backward privacy. However, there is still room for improvement as follows:

1) Although a Bloom filter [6] is used to compress the revoked tag set $\mathcal{R}_{w}$, the client has to store them on the local storage. It is desirable to reduce the amount of local storage on the client side (i.e., state information) as much as possible.
2) Aura employs logical deletion; for a deletion operation of ( $w$, id), an entry (del, $(w$, id $)$ ) is added to an encrypted database EDB. As a result, the size of EDB in Aura is $\widehat{N}=\sum_{w} a_{w}$, where $a_{w}$ is the total number of updates for $w$.
3) As seen above, the server decrypts the ciphertexts and gets the access patterns. Namely, Aura is not a result-hiding scheme.

Our Approach. A common approach to realizing result-hiding schemes is to have the client decrypt the search results $[8,11,13]$. With this approach in mind, our scheme is based on Aura combined with Etemad et al.'s forward-private scheme [14], which are not result-hiding schemes; we no longer employ SRE but the concept of revoked tags. The construction idea for Laura is to perform a variant of logical deletion using tags; sending the server a revoked tag $\tau$ of a deleted pair ( $w$, id), instead of the (encrypted) pair itself, when deleting ( $w$, id ). Therefore, the client does not have to remember the tags. Laura maintains the revoked tags with an (arbitrary) AMQ data structure that supports deletion operations, whereas Sun et al. [25] only considered the Bloom filter for Aura. Hence, the client easily finds the deleted entries with the AMQ.Lookup algorithm, which leads to the result-hiding property while keeping efficiency. ${ }^{3}$

Moreover, we also achieve a smaller EDB through re-addition techniques [14, 26]: for a search query on $w$, the server retrieves all values related to the query from EDB and deletes them. After getting the search result, the client re-adds all entries except for deleted ones for the next search. We summarize what our approach resolves.

1) Laura achieves a smaller (concrete) storage size on the client side than Aura.
2) Laura achieves a smaller $|\mathrm{EDB}|=N^{\prime}=\sum_{w}\left(n_{w}+n_{w, \text { del }}^{(\mathrm{srch})}\right)$ than Aura, where $n_{w, \text { del }}^{(\text {srch })}$ is the total number of times a keyword $w$ has been affected by search operations since the last search for $w$. It clearly holds $\widehat{N} \geq N^{\prime}$.
3) Laura is a result-hiding scheme. Furthermore, compared to existing these schemes, Laura achieves compression of EDB and efficient removal of deleted entries due to the AMQ data structure.

In addition to the above benefits, Laura is more practically efficient than Aura. We will see that in Section 6.
${ }^{3}$ Though the server needs to send the AMQ structure to the client during the search operation, the size of the structure is reasonably small. For example, if we select the cuckoo filter [15] as the AMQ structure, its size is 0.79 MB for 100,000 deleted entries with the false-positive probability $p=10^{-4}$. As a reference, according to the Aura paper [25], $\mathrm{SD}_{d}$ [13] requires 8.58 MB of total communication costs for search.

```
Algorithm: Laura
Setup (1 \({ }^{\kappa}\) )
Client:
    \(\mathrm{k}_{\mathrm{PRF}}, \mathrm{k}_{\mathrm{RH}}, \mathrm{k}_{\mathrm{SKE}} \stackrel{\$}{\leftarrow}\{0,1\}^{\kappa}\)
    \((\mathcal{T}\), aux \() \leftarrow\) AMQ.Gen \(\left(\{0,1\}^{\lambda}\right.\), par \()\)
    \(\mathrm{fc}_{w}, \mathbf{s c}_{w}\), Index[] \(:=\varepsilon / / \varepsilon\) is an empty value
    \(\operatorname{return}\left(k:=\left(\mathrm{k}_{\mathrm{PRF}}, \mathrm{k}_{\mathrm{RH}}, \mathrm{k}_{\mathrm{SKE}}\right), \sigma^{(0)}:=\left(\mathrm{sc}_{w}, \mathrm{fc}_{w}\right), \mathrm{EDB}^{(0)}:=(\right.\) Index, \(\mathcal{T}\), aux \(\left.)\right)\)
Update \(\left(k\right.\), add \(,(w\), id \(\left.), \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)\)
Client:
    \(\tau \leftarrow \pi\left(\mathrm{k}_{\mathrm{RH}}, w \| \mathrm{id}\right)\)
    if \(\mathrm{sc}_{w}\) is undefined then
        \(\left(\mathrm{sc}_{w}, \mathrm{fc}_{w}\right):=(0,0)\)
    \(\mathrm{fc}_{w}:=\mathrm{fc}_{w}+1 / /\) increment \(\mathrm{fc}_{w}\)
    \(\mathrm{K}_{w}^{\left(\mathrm{sc}_{w}\right)} \leftarrow g\left(\mathrm{k}_{\mathrm{PRF}}, w \| \mathbf{s c}_{w}\right) / /\) generate the PRF key for address
    addr \(\leftarrow h\left(\mathrm{~K}_{w}^{\left(\mathrm{sc}_{w}\right)}, \mathrm{fc}_{w}\right), \quad\) val \(\leftarrow \mathrm{E}\left(\mathrm{k}_{\mathrm{SKE}}, \tau \| \mathrm{id}\right)\)
    Send \(\operatorname{trans}_{1}^{(t)}:=(\) addr, val) to the server
    return \(\sigma^{(t+1)}:=\left(\mathrm{sc}_{w}, \mathrm{fc}_{w}\right)_{w \in \mathcal{W}^{(t+1)}}\)
```


## Server:

```
10: Index[addr] := val
11: return \(\operatorname{EDB}^{(t+1)}:=(\operatorname{Index}, \mathcal{T}\), aux \()\)
\(\underline{\operatorname{Update}\left(k, \operatorname{del},(w, \mathrm{id}), \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)}\)
```


## Client:

```
if \(\mathrm{fc}_{w}\) is defined then
\[
\tau \leftarrow \pi\left(\mathrm{k}_{\mathrm{RH}}, w \| \mathrm{id}\right)
\]
Send \(\operatorname{trans}{ }_{1}^{(t)}:=\tau\) to the server
return \(\sigma^{(t+1)}:=\sigma^{(t)}\)
```


## Server:

```
\[
\mathcal{T}^{\prime} \leftarrow \text { AMQ.Insert }(\mathcal{T}, \tau, \text { aux })
\]
\[
\text { return } \mathrm{EDB}^{(t+1)}:=\left(\operatorname{Index}, \mathcal{T}^{\prime}, \text { aux }\right)
\]
```

Fig. 2: Setup and Update of our dynamic SSE scheme Laura.

### 4.2 Our Construction

Let $\pi:\{0,1\}^{*} \rightarrow\{0,1\}^{\lambda}$ and $g:\{0,1\}^{*} \rightarrow\{0,1\}^{\kappa}$ be (variable-input-length) PRF families and $h:\{0,1\}^{*} \rightarrow\{0,1\}^{\eta}$ be a hash function, where $\lambda$ and $\eta$ are polynomials in $\kappa$. Let $\Pi_{\mathrm{AMQ}}=(\mathrm{AMQ}$. Gen, AMQ.Insert, AMQ.Delete, AMQ.Lookup) be an AMQ data structure. We propose a dynamic SSE scheme Laura = (Setup, Update, Search) from $\Pi_{\text {АмQ }}, \pi, g$, and $h$. The pseudo-codes for Laura are given in Figs. 2 and 3, and we provide overviews of each algorithm below.

Setup: Setup $\left(1^{\kappa}\right)$. The client generates a secret key $k:=\left(\mathrm{k}_{\mathrm{SKE}}, \mathrm{k}_{\mathrm{PRF}}, \mathrm{k}_{\mathrm{RH}}\right)$, where $k_{\text {SKE }}$ is an SKE secret key and $k_{\text {PRF }}$ and $k_{\text {RH }}$ are PRF keys used to compute ad-

```
Algorithm: Laura
Search \(\left(k, q, \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)\)
Client:
    \(\mathrm{K}_{q}^{\left(\mathrm{sc}_{w}\right)} \leftarrow g\left(\mathrm{k}_{\mathrm{PRF}}, q \| \mathbf{s c}_{q}\right)\)
    Send \(\operatorname{trans}{ }_{1}^{(t)}:=\left(\mathrm{K}_{q}^{\left(\mathrm{sc}_{w}\right)}, \mathrm{fc}_{q}\right)\) to the server
Server:
    for \(i=1\) to \(\mathrm{fc}_{q}\) do
        addr \(\leftarrow h\left(\mathrm{~K}_{q}^{(\mathrm{sc} w)}, i\right), \quad\) val \(:=\operatorname{Index}[\operatorname{addr}], \quad \mathcal{C}_{q}^{(t)} \leftarrow\) val
        Index[addr] \(:=\) NULL // delete old addresses
    Send \(\operatorname{trans}_{2}^{(t)}:=\left(\mathcal{C}_{q}^{(t)}, \mathcal{T}\right.\), aux \()\) to the client \(/ /\) Send copy of \(\mathcal{T}\)
Client:
    for \(\forall \mathrm{c} \in \mathcal{C}_{q}^{(t)}\) do
        \(\tau \|\) id \(\leftarrow \mathrm{D}\left(\mathrm{k}_{\mathrm{SKE}}, \mathrm{c}\right) / /\) the first \(\lambda\) MSBs of val is tag
        if AMQ.Lookup \((\mathcal{T}, \tau\), aux \()=\) true then \(/ /\) logical deletion of \((q\), id \()\)
            \(\mathcal{D}_{q}^{(t)} \leftarrow \tau\)
        else // search result
            \(\mathcal{X}_{q}^{(t)} \leftarrow \mathrm{id}, \quad \mathcal{Y}_{q}^{(t)} \leftarrow(\tau, \mathrm{id})\)
    \(\mathbf{s c}_{q}:=\mathbf{s c}_{q}+1, \quad \mathrm{fc}_{q}:=\left|\mathcal{X}_{q}^{(t)}\right| / /\) update state
    \(\widehat{\mathrm{K}}_{q}^{\left(\mathrm{sc}_{q}\right)} \leftarrow g\left(\mathrm{k}_{\mathrm{PRF}}, q \| \mathrm{SC}_{q}\right) / /\) generate new keys
    ctr \(:=1\)
    for \(\forall(\tau\), id \() \in \mathcal{Y}_{q}^{(t)}\) do
        \(\widehat{\text { addr }} \leftarrow h\left(\widehat{\mathrm{~K}}_{q}^{\left(\mathrm{sc} c_{q}\right)}, \mathrm{ctr}\right), \quad \widehat{\operatorname{val}} \leftarrow \mathrm{E}\left(\mathrm{k}_{\mathrm{SKE}}, \tau \| \mathrm{id}\right)\)
        \(\mathcal{R}_{q}^{(t)} \leftarrow(\widehat{\mathrm{addr}}, \widehat{\mathrm{val}}), \quad \mathrm{ctr}:=\operatorname{ctr}+1\)
    Send \(\operatorname{trans}_{3}^{(t)}:=\left(\mathcal{D}_{q}^{(t)}, \mathcal{R}_{q}^{(t)}\right)\) to the server
    return \(\left(\mathcal{X}_{q}^{(t)}, \sigma^{(t+1)}:=\left(\mathrm{sc}_{q}, \mathrm{fc}_{q}\right)_{q \in \mathcal{W}^{(t+1)}}\right)\)
```

Server:
for $\forall(\widehat{\text { addr }}, \widehat{\text { val }}) \in \mathcal{R}_{q}^{(t)}$ do
Index[addr] $:=\widehat{\text { val }} / /$ set new addresses and value
for $\forall \tau \in \mathcal{D}_{q}^{(t)}$ do
$\mathcal{T}^{\prime} \leftarrow \operatorname{AMQ}$.Delete $(\mathcal{T}, \tau$, aux $), \quad \mathcal{T}:=\mathcal{T}^{\prime}$
return $\mathrm{EDB}^{(t+1)}:=($ Index, $\mathcal{T}$, aux)

Fig. 3: Search of our dynamic SSE scheme Laura.
dresses and tags, respectively. The client initializes two counters $\mathrm{fc}_{w}$ and $\mathrm{sc}_{w}$, an array Index, and an AMQ data structure $\mathcal{T}$ (along with its auxiliary information aux). The client sets the state information $\sigma^{(0)}:=\left(\mathrm{fc}_{w}, \mathrm{sc}_{w}\right)$, and sends $\mathrm{EDB}^{(0)}:=(\operatorname{Index} \mathcal{T}$, aux $)$ to the server.

Addition: Update $\left(k\right.$, add $\left.,(w, i d), \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)$. First, the client retrieves the file counter $\mathrm{fc}_{w}$ and the search counter $\mathrm{sc}_{w}$ in $\sigma^{(t)}$ and increments $\mathrm{fc}_{w}$. The client next derives a PRF key $\mathrm{K}_{w}^{\left(\mathrm{sc}_{w}\right)}$ from the PRF key $\mathrm{k}_{\text {PRF }}$ using the keyword $w$ to calculate an address addr. Also, the client computes a tag $\tau$, which will be sent
to the server during the deletion operation, of the pair ( $w$, id) from the PRF key $\mathrm{k}_{\mathrm{RH}}$, and encrypts $\tau \|$ id with the SKE secret key $\mathrm{k}_{\mathrm{SKE}}$. The server adds the ciphertext to Index[addr] in $\mathrm{EDB}^{(t)}$.
Deletion: Update $\left(k\right.$, del, $(w$, id $\left.), \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)$. The client only computes the tag $\tau$ of the pair ( $w$, id) using the PRF key $\mathrm{k}_{\text {RH }}$ and sends it to the server. The server executes AMQ.Insert to insert $\tau$ into the data structure $\mathcal{T}$ in $\operatorname{EDB}^{(t)}$.
Search: Search $\left(k, q, \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)$. First, the client creates the PRF key $\left.\mathrm{K}_{q}^{(\mathrm{sc}}{ }^{2}\right)$ for the search keyword $q$ and sends it together with $\mathrm{fc}_{q}$ to the server. For every $i=$ $1, \ldots, \mathrm{fc}_{q}$, the server computes an address $g\left(\mathrm{~K}_{q}^{\left(\mathrm{sc}_{w}\right)}, i\right)$ and adds its stored value val to the set $\mathcal{C}_{q}^{(t)}$. The server sends $\mathcal{C}_{q}^{(t)}$ and a copy of the data structure $\mathcal{T}$ to the client and frees the memory of all the addresses accessed. For every value val $\in$ $\mathcal{C}_{q}^{(t)}$, the client checks whether it has been deleted as follows. The client decrypts val and obtains $\tau \|$ id, and executes AMQ.Lookup with $\tau$ to check whether the pair ( $w$, id) has been logically deleted. If AMQ.Lookup outputs false, id is added to the search result $\mathcal{X}_{q}^{(t)}$. Next, the client re-adds the pairs ( $w$, id) except for the deleted ones. The client increments $\mathrm{sc}_{q}$, and adds the pairs in the same way to the above addition procedure. The server updates $\operatorname{EDB}^{(t)}$ as in the addition procedure and also receives a tag set $\mathcal{D}_{q}^{(t)}$ of the deleted entry. For every tag $\tau \in \mathcal{D}_{q}^{(t)}$, the server executes AMQ.Delete to remove the tags from the data structure $\mathcal{T}$. This re-addition procedure is important to provide forward privacy and reduce the size of EDB and $\mathcal{T}$.

### 4.3 Security Analysis

Correctness. Before analyzing the security of Laura, we show that it satisfies the correctness. Laura might output wrong search results due to false positives in the underlying AMQ data structure $\Pi_{\mathrm{AMQ}}$. The correctness error probability depends on the false-positive probability; due to the bounded false-positive probability property, there exists, and we can evaluate an upper bound $\mu_{n}$ of the false-positive probability. Therefore, by setting the parameters of $\Pi_{\text {АмQ }}$ appropriately, one can make the correctness error probability negligible.
Security. To show the security of Laura, we consider a leakage function called deletion pattern $\operatorname{DelTime}{ }_{q}^{(t)}$, which is a set of counters for all deletion operations on $w$. Namely,

$$
\operatorname{DelTime}_{q}^{(t)}:=\left\{\begin{array}{l|l}
u^{\text {del }} & \begin{array}{l}
\text { ヨid s.t. }\left(u^{\text {add }}, \text { add },(q, \text { id })\right) \in \mathcal{Q}^{(t)} \\
\wedge\left(u^{\text {del }}, \operatorname{del},(q, \text { id })\right) \in \mathcal{Q}^{(t)}
\end{array}
\end{array}\right\}
$$

where we assume $u^{\text {add }}<u^{\text {del }}$ without the loss of generality.
Theorem 1. If $\Pi_{\mathrm{SKE}}$ is CPA-secure, $\Pi_{\mathrm{AMQ}}$ is an AMQ data structure, $\pi$ and $g$ are (variable-input-length) PRF families, and $h$ is a random oracle, the dynamic SSE scheme Laura $=($ Setup, Update, Search $)$ in Figs. 2 and 3 is an $\mathcal{L}$-adaptively
secure result-hiding scheme that supports forward privacy and Type-II backward privacy, with the following leakage function $\mathcal{L}=\left(\mathcal{L}_{\text {Setup }}, \mathcal{L}_{\text {Upd }}, \mathcal{L}_{\text {Srch }}\right)$ :

$$
\begin{aligned}
& \mathcal{L}_{\text {Setup }}\left(1^{\kappa}\right)=\Lambda, \quad \mathcal{L}_{\text {Upd }}(t, \text { op }, \text { in })=(t, \text { op }) \\
& \mathcal{L}_{\text {Srch }}(t, q)=\left(\mathrm{SP}_{q}^{(t)}, \text { Update }_{q}^{(t)}, \operatorname{DelTime}_{q}^{(t)}\right),
\end{aligned}
$$

for any $t$ and any $q \in \Lambda$.
Note that DelTime ${ }_{q}^{(t)}$ can be derived from Update ${ }_{q}^{(t)}$ and op included in $\mathcal{L}_{\text {Upd }}$ : $\operatorname{DelTime}_{q}^{(t)}:=\left\{u \in \operatorname{Update}_{q}^{(t)} \mid \mathcal{L}_{\text {Upd }}(u\right.$, op, $(q$, id $))=(u$, del $\left.)\right\}$. Since Time ${ }_{q}^{(t)}$ and DelHist ${ }_{q}^{(t)}$ imply Update ${ }_{q}^{(t)}$, our construction clearly meets both Type-II backward privacy and the result-hiding property.

Proof (Sketch). Due to the page limitation, we give a proof sketch. We will provide the detailed proof in the full version. We prove that the simulator $S$ can simulate the update and search operations only with the leakage functions $\mathcal{L}$.

Addition. With leakage $\mathcal{L}_{\text {Upd }}(t$, add, in $)=(t$, add $)$ for a query (upd, add, in), S simulates a transcript $\operatorname{trans}_{1}^{(t)}:=($ addr, c$)$. In the real experiment Real, addr and c are $\eta$-bit pseudo-random numbers and ciphertexts of $\tau \|$ id, respectively. If $h$ is a random oracle and $\Pi_{\text {SKE }}$ is CPA-secure, addr and c are indistinguishable from an $\eta$-bit random string $r$ and a ciphertext $\mathrm{c}^{\prime}$ of $0^{\lambda+l}$, except with negligible probability, respectively. Hence, S can set trans ${ }_{1}^{(t)}:=\left(r, \mathrm{c}^{\prime}\right)$.
Deletion. With leakage $\mathcal{L}_{\text {Upd }}(t$, del, in $)=(t$, del $)$ for a query (upd, del, in), S simulates a transcript trans ${ }_{1}^{(t)}:=\tau$. If $\pi$ is a PRF family, $\tau$ is indistinguishable from a $\lambda$-bit random string $r^{\prime}$ except with negligible probability. Therefore, S can set trans ${ }_{1}^{(t)}:=r^{\prime}$.
Search. With leakage $\mathcal{L} \operatorname{Srch}(t, q)=\left(\mathrm{SP}_{q}^{(t)}\right.$, Update ${ }_{q}^{(t)}$, $\left.\operatorname{DeITime}_{q}^{(t)}\right)$ for a query $(\operatorname{srch}, q), \mathrm{S}$ simulates transcripts $\operatorname{trans}_{1}^{(t)}:=\left(\mathrm{K}_{q}^{\left(\mathrm{sc}_{w}\right)}, \mathrm{fc}_{q}\right), \operatorname{trans}_{2}^{(t)}:=\left(\mathcal{C}_{q}^{(t)}, \mathcal{T}\right.$, aux), and $\operatorname{trans}{ }_{3}^{(t)}:=\left(\mathcal{D}_{q}^{(t)}, \mathcal{R}_{q}^{(t)}\right)$. Roughly speaking, due to the security of the underlying PRF $g$, S can set a $\kappa$-bit random string as $\mathrm{K}_{q}^{\left(\mathrm{sc}_{w}\right)}$. Since $\mathrm{fc}_{q}$ can be derived from Update $_{q}^{(t)}$ and $\operatorname{DeITime}_{q}^{(t)}, \mathrm{S}$ can simulate $\operatorname{trans}_{1}^{(t)}$. Since $\mathcal{C}_{q}^{(t)}$ is a set of all ciphertexts generated during the addition operation for $q, \mathrm{~S}$ retrieves a ciphertext simulated at every $u \in$ Update $_{q}^{(t)} \backslash \operatorname{DeITime}_{q}^{(t)}$ and sets them as $\mathcal{C}_{q}^{(t)} \cdot{ }^{4}$ S easily simulates $\mathcal{T}$ and aux since tags for $w$, which are entered into AMQ.Insert and AMQ.Delete, are correctly simulated during the deletion operation. Hence, S can simulate trans ${ }_{2}^{(t)}$. The set $\mathcal{D}_{q}^{(t)}$ of deleted tags can also be simulated as above. $\mathcal{R}_{q}^{(t)}$ can be simulated as in the case of the addition since each ( $\widehat{\mathrm{addr}}, \widehat{\mathrm{val}}$ ) $\in \mathcal{R}_{q}^{(t)}$ is generated in the same manner as the addition operation. Therefore, S can simulate $\operatorname{trans}_{3}^{(t)}$.

[^18]```
Algorithm: v-Laura
Setup (1 \({ }^{\kappa}\) )
Client:
    \(\mathrm{k}_{\mathrm{PRF}}, \mathrm{k}_{\mathrm{RH}}, \mathrm{k}_{\mathrm{SKE}} \stackrel{\$}{\leftarrow}\{0,1\}^{\kappa}\)
    \(\mathrm{fc}_{w}, \mathbf{s c}_{w}, \mathbf{F}[]\), Index[], Cache[] \(:=\varepsilon / / \varepsilon\) is an empty value
    return \(\left(k:=\left(\mathrm{k}_{\mathrm{PRF}}, \mathrm{k}_{\mathrm{RH}}, \mathrm{k}_{\mathrm{SKE}}\right), \sigma^{(0)}:=\left(\mathrm{sc}_{w}, \mathrm{fc}_{w}, \mathbf{F}\right), \mathrm{EDB}^{(0)}:=(\right.\) Index, Cache \(\left.)\right)\)
\(\underline{\text { Update }\left(k, \text { add },(w, i d), \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)}\)
Client:
    \(\tau \leftarrow \pi\left(\mathrm{k}_{\mathrm{RH}}, w \| \mathrm{id}\right)\)
    if \(\mathrm{sc}_{w}\) is undefined then
        \(\left(\mathrm{sc}_{w}, \mathrm{fc}_{w}\right):=(0,0)\)
        \(\left(\mathcal{T}_{w}\right.\), aux \() \leftarrow\) AMQ.Gen \(\left(\{0,1\}^{\lambda}\right.\), par \()\)
        \(\mathbf{F}[w]:=\left(\mathcal{T}_{w}, \mathrm{aux}\right)\)
    \(\mathrm{fc}_{w}:=\mathrm{fc}_{w}+1 / /\) increment \(\mathrm{fc}_{w}\)
    \(\mathrm{K}_{w}^{\left(\mathrm{sc}_{w}\right)} \leftarrow g\left(\mathrm{k}_{\mathrm{PRF}}, w \| \mathbf{s c}_{w}\right) / /\) generate the PRF key for address
    \(\mathrm{c} \leftarrow \mathrm{E}\left(\mathrm{k}_{\mathrm{SKE}}, \mathrm{id} ; \tau\right) / /\) Encryption with nonce
    addr \(\leftarrow h\left(\mathrm{~K}_{w}^{\left(\mathrm{sc}_{w}\right)}, \mathrm{fc}_{w}\right), \quad\) val \(:=\tau \| \mathrm{c}\)
    Send \(\operatorname{trans}_{1}^{(t)}:=\) (addr, val) to the server
    return \(\sigma^{(t+1)}:=\left(\left(\mathrm{sc}_{w}, \mathrm{fc}_{w}\right)_{w \in \mathcal{W}^{(t+1)}}, \mathbf{F}\right)\)
Server:
12: Index[addr] := val
13: return \(\mathrm{EDB}^{(t+1)}:=(\) Index, Cache)
\(\underline{\operatorname{Update}\left(k, \operatorname{del},(w, \text { id }), \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)}\)
Client:
    if \(\mathrm{fc}_{w}\) is defined then
        \(\tau \leftarrow \pi\left(\mathrm{k}_{\mathrm{RH}}, w \| \mathrm{id}\right)\)
        \(\left(\mathcal{T}_{w}\right.\), aux \() \leftarrow \mathbf{F}[w]\)
        \(\mathcal{T}_{w}^{\prime} \leftarrow\) AMQ.Insert \(\left(\mathcal{T}_{w}, \tau\right.\), aux \()\)
        \(\mathbf{F}[w]:=\left(\mathcal{T}_{w}^{\prime}, \mathrm{aux}\right)\)
    return \(\sigma^{(t+1)}:=\left(\left(\mathrm{sc}_{w}, \mathrm{fc}_{w}\right)_{w \in \mathcal{W}^{(t+1)}}, \mathbf{F}\right)\)
```

Fig. 4: Setup and Update of our dynamic SSE scheme v-Laura.

## 5 Extensions

### 5.1 A Variant of Laura: v-Laura

Although Laura is very efficient with small client storage, there is a trade-off between it and the communication cost, as noted in the footnote in Sec. 4.1. Specifically, the server has to send the AMQ structure together with a search result during the search algorithm (line 6 in Fig. 3). The idea to reduce communication cost is to store the AMQ structure on the client side for each keyword, as in Aura. For clients with ample storage or narrow bandwidth, a more suitable

```
Algorithm: v-Laura
Search \(\left(k, q, \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)\)
Client:
    \(\mathrm{K}_{q}^{\left(\mathrm{sc}_{w}\right)} \leftarrow g\left(\mathrm{k}_{\mathrm{PRF}}, q \| \mathrm{sc}_{q}\right)\)
    \(\mathrm{tkn}_{q} \leftarrow g\left(\mathrm{k}_{\mathrm{PRF}}, q\right)\)
    \(\left(\mathcal{T}_{q}\right.\), aux \():=\mathbf{F}[q]\)
    Send \(\operatorname{trans}_{1}^{(t)}:=\left(\mathrm{K}_{q}^{\left(\mathrm{sc}_{w}\right)}, \mathrm{tkn}_{q}, \mathrm{fc}_{q},\left(\mathcal{T}_{q}\right.\right.\), aux \(\left.)\right)\) to the server
Server:
    \(\mathcal{C}_{q}^{(t)}:=\) Cache \(\left[t \mathrm{kn}_{q}\right]\)
    for \(i=1\) to \(\mathrm{fc}_{q}\) do
        addr \(\leftarrow h\left(\mathrm{~K}_{q}^{(\mathrm{sc} w)}, i\right), \quad\) val \(:=\operatorname{Index}[\) addr \(], \quad \mathcal{C}_{q}^{(t)} \leftarrow\) val
        Index[addr] := NULL // delete old addresses
    for \(\forall\) val \(\in \mathcal{C}_{q}^{(t)}\) do
        parse val \(=\tau \| \mathrm{c} / /\) the first \(\lambda\) MSBs of val is tag(nonce)
        if AMQ.Lookup \(\left(\mathcal{T}_{q}, \tau\right.\), aux \()=\) true then \(/ /\) logical deletion of \((w\), id \()\)
            \(\mathcal{C}_{q}^{(t)}:=\mathcal{C}_{q}^{(t)} \backslash\{\) val \(\}\)
    Cache \(\left[\mathrm{tkn}_{q}\right]:=\mathcal{C}_{q}^{(t)}\)
    Send \(\operatorname{trans}_{2}^{(t)}:=\mathcal{C}_{q}^{(t)}\) to the client
    return EDB \(^{(t+1)}:=(\) Index, Cache)
Client:
    for \(\forall(\tau, \mathrm{c}) \in \mathcal{C}_{q}^{(t)}\) do
        \(\mathcal{X}_{q}^{(t)} \leftarrow \mathrm{D}\left(\mathrm{k}_{\text {skE }}, \mathrm{c} ; \tau\right) / /\) decrypt c to get search result
    \(\left(\mathcal{T}_{q}^{\prime}\right.\), aux \() \leftarrow\) AMQ.Gen \(\left(\{0,1\}^{\lambda}\right.\), par \()\)
10: \(\mathrm{fc}_{q}:=0, \quad \mathrm{sc}_{q}:=\mathrm{sc}_{q}+1, \quad \mathbf{F}[q]:=\left(\mathcal{T}_{q}^{\prime}\right.\), aux \() / /\) update state
11: return \(\left(\mathcal{X}_{q}^{(t)}, \sigma^{(t+1)}:=\left(\left(\mathrm{sc}_{q}, \mathrm{fc}_{q}\right)_{q \in \mathcal{W}^{(t+1)}}, \mathbf{F}\right)\right)\)
```

Fig. 5: Search of our dynamic SSE scheme v-Laura.
and efficient variant scheme than Laura, called v-Laura, can be constructed. At first glance, it seems to be the same as Aura, but the following are differences;

1) AMQ is used only as a compression of the deleted tag set without SRE functionality. Therefore, efficient AMQs can be selected, not limited to the bloom filter used for SRE in Aura. The v-Laura also achieves efficient search by eliminating SRE processing, which is dominant in Aura searches (see Sec. 5).
2) The server removes the deleted entries using AMQ structure while the client decrypts the search results to achieve result-hiding, similar to Laura.
3) The v-Laura can compress the size of val in EDB with the idea of using $\tau$ as a nonce in encryption. In some block cipher modes of CPA-secure $\Pi_{\mathrm{SKE}}$, the nonce is used for security and is stored with the ciphertext. Since $\tau$ plays the role of nonce, it can compress the size of the original nonce.

The pseudo-codes for v-Laura are given in Figs. 4 and 5, and we provide overviews of each algorithm below. However, we omit the same part of Laura.

Setup: Setup $\left(1^{\kappa}\right)$. The client generates a secret key $k:=\left(\mathrm{k}_{\mathrm{SKE}}, \mathrm{k}_{\mathrm{PRF}}, \mathrm{k}_{\mathrm{RH}}\right)$. The client initializes two counters $\mathrm{fc}_{w}$ and $\mathrm{sc}_{w}$, and three array Index and Cache and $\mathbf{F}$. The client sets the state information $\sigma^{(0)}:=\left(\mathrm{fc}_{w}, \mathrm{sc}_{w}, \mathbf{F}\right)$, and sends $\mathrm{EDB}^{(0)}:=$ (IndexCache) to the server.

Addition: Update $\left(k\right.$, add, $\left.(w, \mathrm{id}), \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)$. First, the client calculates an address addr and tag $\tau$, like Laura. Also, the client encrypts id using $\tau$ as nonce (i.e., $\left.\mathrm{c} \leftarrow \mathrm{E}\left(\mathrm{k}_{\mathrm{SKE}}, \mathrm{id} ; \tau\right)\right)$ and sends addr and val $:=\tau \| \mathrm{c}$ to the server. The server adds val to Index[addr] in $\mathrm{EDB}^{(t)}$.
Deletion: Update $\left(k, \operatorname{del},(w, i d), \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)$. The client only computes the tag $\tau$ and executes AMQ.Insert to insert $\tau$ into the data structure $\mathcal{T}_{w}$ for $w$ in $\sigma^{(t)}$.
Search: Search $\left(k, q, \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)$. First, the client creates $\mathrm{K}_{q}^{\left(\mathrm{sc}_{w}\right)}$ and $\mathrm{tkn} \mathrm{n}_{q}$ with the PRF key $\mathrm{k}_{\text {PRF }}$ and sends them together with $\mathrm{fc}_{q}$ and $\mathcal{T}_{q}$ to the server. The server gets Cache $\left[\mathrm{tkn}_{q}\right]$ as a set $\mathcal{C}_{q}^{(t)}$. For every $i=1, \ldots, \mathrm{fc}_{q}$, the server computes an address $g\left(\mathrm{~K}_{q}^{(\mathrm{sc} w)}, i\right)$ and adds its stored value val to the set $\mathcal{C}_{q}^{(t)}$. For every val $\in \mathcal{C}_{q}^{(t)}$, the server parse val $:=\tau \| c$ and executes AMQ.Lookup with $\tau$ and $\mathcal{T}_{q}$ to check whether the pair ( $q$, id) has been logically deleted. If AMQ.Lookup outputs true, val is removed from $\mathcal{C}_{q}^{(t)}$. Next, the server sets $\mathcal{C}_{q}^{(t)}$ to Cache[tkn $\left.{ }_{q}\right]$ and updates $\operatorname{EDB}^{(t)}$, and sends $\mathcal{C}_{q}^{(t)}$ to the client. For every value val $\in \mathcal{C}_{q}^{(t)}$, the client decrypts val to obtain id and adds it to the search result $\mathcal{X}_{q}^{(t)}$. Finally, the client initializes $\mathcal{T}_{q}$ and $\mathrm{fc}_{q}$ and increments $\mathrm{sc}_{q}$.
v-Laura also satisfies Theorem 1. The proof is shown in full version.

### 5.2 A Strongly Secure variant of Laura: s-Laura

As explained in the introduction, Aura implicitly requires every pair of ( $w$, id) to be added at most only once; it does not allow the client to re-add previously deleted pairs. Indeed, Laura and v-Laura work well under the same assumption. In other words, if the client wants to add and delete a pair ( $w$, id) multiple times, those schemes are no longer Type-II backward private. This limitation stems from the fact that the corresponding tag of the pair ( $w$, id) is generated deterministically in those schemes. The extended scheme s-Laura, which stands for strongly-secure Laura, allows to run Update of pair ( $w$, id) any number of times. The basic idea of $s$-Laura is that the deletion tag of the pair ( $w$, id) changes with each deletion. The client holds extra information $\mathrm{dc}_{w}$ which increments for each deletion regarding $w$. When pair ( $w$, id) is deleted, a delete tag $\tau_{\mathrm{dc}_{w}}$ is generated from $\tau$ and $\mathrm{dc}_{w}$. The client then computes tags $\tau_{1}, \ldots, \tau_{\mathrm{dc}}^{w}$ from $\tau$ and $\mathrm{dc}_{w}$, and executes AMQ.Lookup with $\tau_{i}$ for every $i \in\left[\mathrm{dc}_{w}\right]$ to check whether the pair ( $w$, id) has been logically deleted. If AMQ.Lookup outputs false for all tags, id is added to the search result $\mathcal{X}_{q}^{(t)}$. However, the search time of s-Laura increases linearly with the number of deletions, as shown in Table. 1. Hence, s-Laura has
not been evaluated for implementation in Sec. 6. Efficient construction is a future work.

We give the pseudo-codes for s-Laura in Appendix A, and provide overviews of each algorithm below.

Setup: Setup $\left(1^{\kappa}\right)$. The client generates a secret key $k:=\left(\mathrm{k}_{\mathrm{SKE}}, \mathrm{k}_{\mathrm{PRF}}, \mathrm{k}_{\mathrm{RH}}\right)$, where $k_{\text {SKE }}$ is an SKE secret key and $k_{\text {PRF }}$ and $k_{\text {RH }}$ are PRF keys used to compute addresses and tags, respectively. The client initializes three counters $\mathrm{fc}_{w}, \mathrm{sc}_{w}$, and $\mathrm{dc}_{w}$, an array Index, and an AMQ data structure $\mathcal{T}$ (along with its auxiliary information aux). The client sets the state information $\sigma^{(0)}:=\left(\mathrm{fc}_{w}, \mathrm{sc}_{w}, \mathrm{dc}_{w}\right)$, and sends $\operatorname{EDB}^{(0)}:=(\operatorname{Index} \mathcal{T}$, aux $)$ to the server.

Addition: Update $\left(k\right.$, add, $\left.(w, i d), \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)$. First, the client retrieves the file counter $\mathrm{fc}_{w}$ and the search counter $\mathrm{sc}_{w}$ in $\sigma^{(t)}$ and increments $\mathrm{fc}_{w}$. The client next derives a PRF key $\mathrm{K}_{w, 0}^{(\mathrm{sc} w)}$ from the PRF key $\mathrm{k}_{\text {PRF }}$ using the keyword $w$ to calculate an address addr. Also, the client computes a persistent tag $\tau$, which will be used to derive an ephemeral tag $\tau_{i}$ during the deletion operation, of the pair ( $w$, id) from the PRF key $\mathrm{k}_{\text {RH }}$, and encrypts $\tau \|$ id with the SKE secret key $\mathrm{k}_{\text {SKE }}$. The server adds the ciphertext to Index[addr] in $\mathrm{EDB}^{(t)}$.
Deletion: Update $\left(k\right.$, del, $\left.(w, i d), \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)$. First, the client retrieves the deletion counter $\mathrm{dc}_{w}$ and increments it. The client computes the persistent $\operatorname{tag} \tau$ as in the addition operation. Then, the client derives a key $\mathrm{K}_{w, 1}^{(\mathrm{sc} w)}$ from the PRF key $\mathrm{k}_{\text {PRF }}$ using the keyword $w$ and generates an ephemeral tag $\tau_{\mathrm{d} c_{w}}$ from the derived key $\mathrm{K}_{w, 1}^{\left(\mathrm{sc}_{w}\right)}$, the persistent tag $\tau$, and the counter $\mathrm{dc}_{w}$. The server executes AMQ.Insert to insert $\tau$ into the data structure $\mathcal{T}$ in $\operatorname{EDB}^{(t)}$.

Search: Search $\left(k, q, \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)$. First, the client creates the PRF key $\mathrm{K}_{q, 0}^{\left(\mathrm{sc}_{w}\right)}$ for the search keyword $q$ and sends it together with $\mathrm{fc}_{q}$ to the server. For every $i=$ $1, \ldots, \mathrm{fc}_{q}$, the server computes an address $g\left(\mathrm{~K}_{w, 0}^{\left(\mathrm{sc}_{w}\right)}, i\right)$ and adds its stored value val to the set $\mathcal{C}_{q}^{(t)}$. The server sends $\mathcal{C}_{q}^{(t)}$ and a copy of the data structure $\mathcal{T}$ to the client and frees the memory of all the addresses accessed. For every value val $\in$ $\mathcal{C}_{q}^{(t)}$, the client checks whether it has been deleted as follows. The client decrypts val and obtains $\tau \|$ id. The client then computes ephemeral tags $\tau_{1}, \ldots, \tau_{\mathrm{dc}}^{q} \boldsymbol{}$ from $\tau$ and $\mathrm{dc}_{q}$, and executes AMQ.Lookup with $\tau_{i}$ for every $i \in\left[\mathrm{dc}_{q}\right]$ to check whether the pair ( $w, \mathrm{id}$ ) has been logically deleted. If AMQ.Lookup outputs false for all ephemeral tags, id is added to the search result $\mathcal{X}_{q}^{(t)}$. Next, the client re-adds the pairs ( $w$, id) except for the deleted ones. The client increments $\mathrm{sc}_{q}$, and adds the pairs in the same way to the above addition procedure. The server updates $E \mathrm{EDB}^{(t)}$ as in the addition procedure and also receives a set $\mathcal{D}_{q}^{(t)}$ of the ephemeral tags of the deleted entry. For every ephemeral tag $\tau_{i} \in \mathcal{D}_{q}^{(t)}$, the server executes AMQ.Delete to remove the tags from the data structure $\mathcal{T}$. This re-addition procedure is important to provide forward privacy and reduce the size of $\mathcal{T}$.
s-Laura also satisfies Theorem 1. The proof is shown in full version.


Fig. 6: Addition cost.


Fig. 7: Search cost without deletion.

## 6 Experiments

Implementation. We implemented the proposed protocols Laura and v-Laura in C++ and evaluated their performance comparatively. ${ }^{5}$ We compare them with Aura [25] implemented in C++ [1] for each protocol. For instances and technical details of Aura, please refer to [25, 1]. These experiments were done in an Ubuntu 18.04 LTS server with 756GB RAM, using Docker (version 24.0.4) [3]. We used AES-GCM for the instantiation of SKE $\Pi_{\text {SKE }}$. The PRFs $\pi, g$, and the random oracle $h$ are realized with AES-GCM and GMAC, respectively. They are implemented using the EVP functions API on the open SSL library (version 3.0.2 15 Mar 2022), and AES-GCM is accelerated by the Intel AES-NI instruction set. For the instance of the AMQ data structure of Laura and v-Laura, we choose the cuckoo filter [15] implemented in [2].

The sizes of keys and outputs of AES and PRF are 128 bits, respectively. The identifier id and each counter (i.e. $\mathrm{fc}_{w}, \mathrm{sc}_{w}$ ) are 32 -bit integers. For experiments on search, we measure the time it takes the server to get all the decrypted identifiers in the search results. Note that both the client and server run locally and communication costs are not taken into account.

Parameter Setting. Throughout the experiments, we set the false-positive probability $p=10^{-4}$, which was also considered practically acceptable in the Aura paper [25]. To ensure that false-positive probability, we need to set the maximum number $d_{w}$ of elements inserted into the AMQ data structure in Laura and v-Laura (resp., the Bloom filter in Aura) at the beginning of the protocol. To be precise, Aura and v-Laura prepares a filter per keyword, while Laura employ only one AMQ structure for the whole system. Therefore, unless otherwise stated, we set $d_{w}=1,000$ for Aura and $v$-Laura and $d_{\Lambda}=10,000,000$ for Laura, where $d_{w}$ and $d_{\Lambda}=\sum_{w \in \Lambda} d_{w}$.
Addition Cost. We give the addition costs of Aura, Laura, and v-Laura in Fig. 6. This results surprisingly show a marked performance difference between

[^19]
## 20 T. Amada et al.



Fig. 8: Deletion cost.


Fig. 9: Search cost with deletion.
ours and Aura. Specifically, Laura and v-Laura takes less than 1.0 s to add 200,000 keyword-identifier pairs, whereas Aura takes 59.5 s . This is due to the concrete construction of the underlying SRE scheme, which requires many resources for the addition.

Search Cost without Deletion. Fig. 7 compares the search costs of Aura, Laura, and v-Laura when no entries on $w$ have been deleted. The search costs of the three schemes increase linearly with the number of pairs. When the search results is 200,000 pairs, Laura, v-Laura, and Aura take $1.05 \mathrm{~s}, 0.75 \mathrm{~s}$ and 1.18 s respectively.

Deletion Cost. As can be seen in Fig. 8, the deletion costs for Aura, Laura, and v -Laura are remarkably fast since the deletion procedures of these schemes only require the calculation of the tag corresponding to the pair to be deleted and the insertion to the filter. Specifically, for 1,000 deleted entries, Laura, v-Laura and Aura take $0.68 \mathrm{~ms}, 0.67 \mathrm{~ms}$ and 0.52 ms respectively. The Laura and v-Laura are slightly slower since the cuckoo filter [15] has the property that as more items are inserted to the filter, the frequency of kicked out an item in the insertion also increases.

Search Cost with Deletion. We show the effect of deletion on search costs in Fig. 9. After adding 2,000 pairs of ( $w$, id), we delete pairs and then search for $w$. Fig. 9 shows the search time with the range of the number of the deleted pairs from 0 to 1,000 . The Laura and v-Laura are remarkably faster than Aura. Specifically, when deleting 1,000 entries (i.e., 1,000 results of 2,000 entries), Laura, v-Laura and Aura take $0.61 \mathrm{~ms}, 0.41 \mathrm{~ms}$ and 169.0 ms respectively. Compared Aura with v-Laura, it is clear that the computational complexity of SRE is dominant. More interestingly, Aura takes longer when no deletion occurred due to the underlying SRE construction.

Acknowledgment This work was supported by JSPS KAKENHI Grant Numbers JP21H03395, JP21H03441, JP22H03590, JP23H00468, JP23H00479, JP23K17455, JST CREST JPMJCR23M2, and MEXT Leading Initiative for Excellent Young Researchers.

## References

1. Aura. https://github.com/MonashCybersecurityLab/Aura
2. Cuckoo filter. https://github.com/efficient/cuckoofilter/tree/master
3. Docker. https://www.docker.com/
4. Bender, M.A., Farach-Colton, M., Johnson, R., Kraner, R., Kuszmaul, B.C., Medjedovic, D., Montes, P., Shetty, P., Spillane, R.P., Zadok, E.: Don't thrash: How to cache your hash on flash. Proc. VLDB Endow. 5(11), 1627-1637 (2012)
5. Blackstone, L., Kamara, S., Moataz, T.: Revisiting leakage abuse attacks. In: NDSS 2020. The Internet Society (2020)
6. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM 13(7), 422-426 (1970)
7. Bost, R.: $\sum \mathrm{o} \varphi \mathrm{o}$ : Forward secure searchable encryption. In: Proc. of ACM CCS 2016. pp. 1143-1154. ACM (2016)
8. Bost, R., Minaud, B., Ohrimenko, O.: Forward and backward private searchable encryption from constrained cryptographic primitives. In: Proc. of ACM CCS 2017. pp. 1465-1482. ACM (2017)
9. Cash, D., Grubbs, P., Perry, J., Ristenpart, T.: Leakage-abuse attacks against searchable encryption. In: Proc. of ACM CCS 2015. pp. 668-679. ACM (2015)
10. Cash, D., Jaeger, J., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M.C., Steiner, M.: Dynamic searchable encryption in very-large databases: Data structures and implementation. In: Proc. of NDSS 2014. The Internet Society (2014)
11. Chamani, J.G., Papadopoulos, D., Papamanthou, C., Jalili, R.: New constructions for forward and backward private symmetric searchable encryption. In: Proc. of ACM CCS 2018. pp. 1038-1055. ACM (2018)
12. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Proc. of ACM CCS 2006. pp. 79-88. ACM (2006)
13. Demertzis, I., Chamani, J.G., Papadopoulos, D., Papamanthou, C.: Dynamic searchable encryption with small client storage. In: Proc. of NDSS 2020. The Internet Society (2020)
14. Etemad, M., Küpçü, A., Papamanthou, C., Evans, D.: Efficient dynamic searchable encryption with forward privacy. PoPETs 2018(1), 5-20 (2018)
15. Fan, B., Andersen, D.G., Kaminsky, M., Mitzenmacher, M.D.: Cuckoo filter: Practically better than bloom. In: Proc. of CoNEXT 2014. pp. 75-88 (2014)
16. Hahn, F., Kerschbaum, F.: Searchable encryption with secure and efficient updates. In: ACM SIGSAC Conference on Computer and Communications Security, CCS 2014. pp. 310-320. ACM, New York, NY, USA (2014)
17. Islam, M.S., Kuzu, M., Kantarcioglu, M.: Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In: Proc. of NDSS 2012. The Internet Society (2012)
18. Kamara, S., Moataz, T.: Computationally volume-hiding structured encryption. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology - EUROCRYPT 2019. pp. 183-213. Springer International Publishing, Cham (2019)
19. Kamara, S., Papamanthou, C.: Parallel and dynamic searchable symmetric encryption. In: Proc. of FC 2013. pp. 258-274. Springer (2013)
20. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In: Proc. of ACM CCS 2012. pp. 965-976. ACM (2012)
21. Katz, J., Lindell, Y.: Introduction to Modern Cryptography, Second Edition. CRC Press (2014)
22. Miers, I., Mohassel, P.: IO-DSSE: scaling dynamic searchable encryption to millions of indexes by improving locality. In: Network and Distributed System Security Symposium, NDSS 2017 (2017)
23. Naveed, M., Prabhakaran, M., Gunter, C.: Dynamic searchable encryption via blind storage. In: IEEE Symposium on Security and Privacy, S\&P 2014. pp. 639654 (May 2014)
24. Song, D.X., Wagner, D.A., Perrig, A.: Practical techniques for searches on encrypted data. In: IEEE S\&P 2000. pp. 44-55. IEEE (2000)
25. Sun, S., Steinfeld, R., Lai, S., Yuan, X., Sakzad, A., Liu, J.K., Nepal, S., Gu, D.: Practical non-interactive searchable encryption with forward and backward privacy. In: Proc. of NDSS 2021. The Internet Society (2021)
26. Wang, J., Chow, S.S.M.: Omnes pro uno: Practical multi-writer encrypted database. In: 31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022. pp. 2371-2388. USENIX Association (2022)
27. Watanabe, Y., Ohara, K., Iwamoto, M., Ohta, K.: Efficient dynamic searchable encryption with forward privacy under the decent leakage. In: Proc. of ACM CODASPY 2022. pp. 312-323. ACM (2022)
28. Zhang, Y., Katz, J., Papamanthou, C.: All your queries are belong to us: The power of file-injection attacks on searchable encryption. In: Proc. of USENIX Security 2016. pp. 707-720. USENIX Association (2016)

## A Formal Description of s-Laura

We give the concrete procedures of s-Laura in Figs 10 and 11.

```
Algorithm: s-Laura
Setup \(\left(1^{\kappa}\right)\)
Client:
    1: \(\mathrm{k}_{\mathrm{PRF}}, \mathrm{k}_{\mathrm{RH}}, \mathrm{k}_{\mathrm{SKE}} \stackrel{\$}{\leftarrow}\{0,1\}^{\kappa}\)
    2: \((\mathcal{T}\), aux \() \leftarrow\) AMQ.Gen \(\left(\{0,1\}^{\lambda}\right.\), par \()\)
    3: \(\mathrm{fc}_{w}, \mathrm{sc}_{w}, \mathrm{dc}_{w}\), Index []\(:=\varepsilon / / \varepsilon\) is an empty value
    4: \(\operatorname{return}\left(k:=\left(\mathrm{k}_{\mathrm{PRF}}, \mathrm{k}_{\mathrm{RH}}, \mathrm{k}_{\mathrm{SKE}}\right), \sigma^{(0)}:=\left(\mathrm{sc}_{w}, \mathrm{fc}_{w}, \mathrm{dc}_{w}\right), \mathrm{EDB}^{(0)}:=(\operatorname{Index}, \mathcal{T}\right.\), aux \(\left.)\right)\)
Update \(\left(k\right.\), add, \((w\), id \(\left.), \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)\)
```


## Client:

```
1: \(\tau \leftarrow \pi\left(\mathrm{k}_{\mathrm{RH}}, w \| \mathrm{id}\right)\)
2: if \(\mathrm{sc}_{w}\) is undefined then
3: \(\quad\left(\mathrm{sc}_{w}, \mathrm{fc}_{w}, \mathrm{dc}_{w}\right):=(0,0,0)\)
4: \(\mathrm{fc}_{w}:=\mathrm{fc}_{w}+1 / /\) increment \(\mathrm{fc}_{w}\)
5: \(\mathrm{K}_{w, 0}^{\left(\mathrm{sc}_{w}\right)}:=g\left(\mathrm{k}_{\mathrm{PRF}}, w\left\|\mathrm{sc}_{w}\right\| 0\right) / /\) generate the PRF key for address
6: addr \(\leftarrow h\left(\mathrm{~K}_{w, 0}^{\left(\mathrm{sc}_{w}\right)}, \mathrm{fc}_{w}\right)\)
\(7:\) val \(\leftarrow \mathrm{E}\left(\mathrm{k}_{\mathrm{SKE}}, \tau \| \mathrm{id}\right)\)
8: Send trans \({ }_{1}^{(t)}:=\) (addr, val) to the server
9: return \(\sigma^{(t+1)}:=\left(\mathrm{sc}_{w}, \mathrm{fc}_{w}, \mathrm{dc}_{w}\right)_{w \in \mathcal{W}^{(t+1)}}\)
```

Server:
10: Index[addr] $:=\mathrm{val}$
11: return $\mathrm{EDB}^{(t+1)}:=(\operatorname{Index}, \mathcal{T}$, aux)
$\operatorname{Update}\left(k, \operatorname{del},(w\right.$, id $\left.), \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)$

## Client:

1: if $\mathrm{dc}_{w}$ is defined then
2: $\quad \mathrm{dc}_{w}:=\mathrm{dc}_{w}+1$
3: $\quad \tau \leftarrow \pi\left(\mathrm{k}_{\mathrm{RH}}, w \| \mathrm{id}\right)$
4: $\quad \mathrm{K}_{w, 1}^{(\mathrm{sc} w)}:=g\left(\mathrm{k}_{\mathrm{PRF}}, w\left\|\mathrm{sc}_{w}\right\| 1\right)$
5: $\quad \tau_{\mathrm{dc}_{w}} \leftarrow \pi\left(\mathrm{~K}_{w, 1}^{\left(\mathrm{sc}_{w}\right)}, \tau \| \mathrm{dc}_{w}\right)$
6: $\quad$ Send $\operatorname{trans}_{1}^{(t)}:=\tau_{\mathrm{dc}_{w}}$ to the server
return $\sigma^{(t+1)}:=\left(\mathrm{sc}_{w}, \mathrm{fc}_{w}, \mathrm{dc}_{w}\right)_{w \in \mathcal{W}^{(t+1)}}$
Server:
8: $\mathcal{T}^{\prime} \leftarrow$ AMQ.Insert $\left(\mathcal{T}, \tau_{\mathrm{dc}_{w}}\right.$, aux $)$
9: return $\mathrm{EDB}^{(t+1)}:=\left(\operatorname{Index}, \mathcal{T}^{\prime}\right.$, aux)
Fig. 10: Setup and Update of our dynamic SSE scheme s-Laura.

## Algorithm: s-Laura

## Search $\left(k, q, \sigma^{(t)} ; \mathrm{EDB}^{(t)}\right)$

## Client:

1: $\mathrm{K}_{q, 0}^{(\mathrm{sc} w)}:=g\left(\mathrm{k}_{\mathrm{PRF}}, q\left\|\mathrm{sc}_{w}\right\| 0\right)$
2: Send $\operatorname{trans}{ }_{1}^{(t)}:=\left(\mathrm{K}_{q, 0}^{\left(\mathrm{sc}_{w}\right)}, \mathrm{fc}_{q}\right)$ to the server

## Server:

3: for $i=1$ to $\mathrm{fc}_{q}$ do
4: $\quad$ addr $\leftarrow h\left(\mathrm{~K}_{q, 0}^{\left(\mathrm{sc}_{w}\right)}, i\right), \quad \mathcal{C}_{q}^{(t)} \leftarrow \operatorname{Index}[$ addr]
5: Index[addr] $:=$ NULL // delete old addresses
6: Send $\operatorname{trans}_{2}^{(t)}:=\left(\mathcal{C}_{q}^{(t)}, \mathcal{T}\right.$, aux $)$ to the client // Send copy of $\mathcal{T}$

## Client:

$\mathrm{K}_{q, 1}^{\left(\mathrm{sc}_{w}\right)}:=g\left(\mathrm{k}_{\mathrm{PRF}}, q\left\|\mathrm{sc}_{q}\right\| 1\right)$
for $\forall \mathrm{c} \in \mathcal{C}_{q}^{(t)}$ do // define Loop1 for Jump
$\tau \| \mathrm{id} \leftarrow \mathrm{D}\left(\mathrm{k}_{\mathrm{SKE}}, \mathrm{c}\right) / /$ the first $\lambda$ MSBs of val is tag
for $i=1$ to $\mathrm{dc}_{q}$ do
$\tau_{i} \leftarrow \pi\left(\mathrm{~K}_{q, 1}^{\left(\mathrm{sc}_{w}\right)}, \tau \| i\right)$
if AMQ.Lookup $\left(\mathcal{T}, \tau_{i}\right.$, aux $)=$ true then

$$
\mathcal{D}_{q}^{(t)} \leftarrow \tau_{i}
$$

Jump Loop1 and next element
$\mathcal{X}_{q}^{(t)} \leftarrow \mathrm{id}, \quad \mathcal{Y}_{q}^{(t)} \leftarrow(\mathrm{id}, \tau)$
$\mathrm{sc}_{q}:=\mathrm{sc}_{q}+1, \quad \mathrm{fc}_{q}:=\left|\mathcal{X}_{q}^{(t)}\right|, \quad \mathrm{dc}_{q}:=0 / /$ update state
$\widehat{\mathrm{K}}_{q, 0}^{\left(\mathrm{sc}_{q, 0}\right)}:=g\left(\mathrm{k}_{\mathrm{PRF}}, q\left\|\mathrm{sc}_{q}\right\| 0\right) / /$ generate new keys
ctr $:=1$
for $\forall(\tau$, id $) \in \mathcal{Y}_{q}^{(t)}$ do $\mathcal{R}_{q}^{(t)} \leftarrow\left(h\left(\widehat{\mathrm{~K}}_{q, 0}^{\left(\mathrm{sc}_{q, 0}\right)}, \operatorname{ctr}\right), \mathrm{E}\left(\mathrm{k}_{\mathrm{SKE}}, \tau \| \mathrm{id}\right)\right) / /$ new $(\widehat{\text { addr }}, \widehat{\mathrm{val}})$ pair $\mathrm{ctr}:=\mathrm{ctr}+1$
Send $\operatorname{trans}_{3}^{(t)}:=\left(\mathcal{D}_{w}^{(t)}, \mathcal{R}_{q}^{(t)}\right)$ to the server
return $\left(\mathcal{X}_{q}^{(t)}, \sigma^{(t+1)}:=\left(\mathrm{sc}_{w}, \mathrm{fc}_{q}, \mathrm{dc}_{q}\right)_{q \in \mathcal{W}^{(t+1)}}\right)$

## Server:

24: for $\forall(\widehat{\operatorname{addr}}, \widehat{\mathrm{val}}) \in \mathcal{R}_{q}^{(t)}$ do
25: Index[大addr] $:=\widehat{\mathrm{val}} / /$ set new addresses and value
26: for $\forall \tau_{i} \in \mathcal{D}_{q}^{(t)}$ do
27: $\quad \mathcal{T}^{\prime} \leftarrow \operatorname{AMQ}$.Delete $\left(\mathcal{T}, \tau_{i}\right.$, aux $), \quad \mathcal{T}:=\mathcal{T}^{\prime}$
28: return $\mathrm{EDB}^{(t+1)}:=($ Index, $\mathcal{T}$, aux)
Fig. 11: Search of our dynamic SSE scheme s-Laura.

# Finsler Encryption* 

Tetsuya Nagano ${ }^{1}$ and Hiroaki Anada ${ }^{2}$<br>University of Nagasaki, Nagasaki 851-2195, Japan<br>hnagano@sun.ac.jp<br>Aomori University, Tokyo 134-0087, Japan<br>anada@aomori-u.ac.jp


#### Abstract

Inspired by previous work with the first example proposed at SecITC 2020, we give a general description of Finsler encryption that is based on a Finsler space, which uses a kind of a differentiable geometry on a smooth manifold, with appropriate quantization as the security parameter. Key generation, encryption and decryption algorithms are introduced in detail, and a further example is presented. Then we analyse security properties of Finsler encryption. First, as the dimension (as another security parameter) increases, the length of the secret key also increases, and hence the computational hardness becomes stronger. Second, we prove indistinguishability against chosen-plaintext attacks.


Keywords: Finsler geometry • Differential geometry • Linear parallel displacement problem • Underdetermined systems of equations • Mappingdecomposition problem

## 1 Introduction

Finsler encryption is a new cryptographic system that has recently been studied. In previous work[10] proposed at SecITC 2020, an example was given in the case of dimension 2. To capture the intuition, we first state the outline of this system briefly. First of all, we choose a Finsler space with the asymmetric property (See Appendix (2)). Next, the geodesics and the linear parallel displacement must be decided. Both of these are defined by certain differential equations system. And the equation of the energy of a vector is calculated. The key generation is performed using linear parallel displacement of vectors and preserved norms. The obtained key is an $n+1$-dimensional vector consisting of rational expressions with several parameters as components. The $n$ is the dimension of Finsler space. The encryption algorithm generates the ciphertext by calculating several sums of vectors obtained by substituting several given parameter values. On the other hand, the decryption algorithm is performed based on the value of parameter $\tau$ obtained from a system of simultaneous linear equations with unknown plaintext components and homogeneous quadratic equations involving the squared

[^20]norms of vectors. In the next section, we will present a detailed explanation of the Finsler space used to generate Finsler encryption and its key generation, encryption and decryption. In the following section, we will explain in detail the strength of Finsler encryption, but the intuitive outline is as follows.

If an attacker attempts to decrypt a ciphertext that is encrypted with a public key, he must solve a system of underdetermined equations. This is because, by setting $k$ to be greater than or equal to $n+1$, the number of unknown variables becomes greater than the number of equations that can be obtained from the ciphertext and the public key. Generally, solutions to underdetermined systems of equations can only be obtained in the form that includes unknown constants, which we call "the property of SUS". Therefore, determining one plaintext from countless solutions is impossible. Next, finding a "linear parallel displacement" is an assumably computationally hard problem, which we call the Linear Parallel Displacement problem (LPD problem). We emphasize that the problem arises from the structure of asymmetric Finsler spaces, and currently no algorithm to solve it known. The last one is the difficulty of solving the composite mapping problem, which we call Mapping-decomposition problem. That is, the energy expression is a product of five regular matrices. It is difficult to decompose the energy function, which is a product of five regular matrices, to obtain the five regular matrices.

In this paper, we formalize Finsler encryption in the case of general dimension $n$. Then we study the strength of our Finsler encryption. Note that we implicitly use the general theory on Finsler geometry and linear parallel displacement, that can be seen in previous publications.

## 2 Preliminaries

### 2.1 Public-Key Encryption

A public-key encryption scheme PKE consists three probabilistic polynomialtime (PPT) algorithms; PKE = (KeyGen, Enc, Dec).

- KeyGen $\left(1^{\lambda}\right) \rightarrow(\mathbf{P K}, \mathbf{S K})$. On input the security parameter $1^{\lambda}$, this PPT algorithm generates a secret key SK and the corresponding public key PK. It returns (PK, SK).
- Enc $(\mathbf{P K}, m) \rightarrow c t$. On input the public key $\mathbf{P K}$ and a message $m$, this PPT algorithm generates a ciphertext ct. It returns ct.
- $\operatorname{Dec}(\mathbf{S K}, c t) \rightarrow \hat{m}$. On input the secret key SK and a ciphertext $c t$, this deterministic polynomial-time algorithm generates a decrypted message $\hat{m}$. It returns $\hat{m}$.

Correctness should hold for PKE. That is; for any $1^{\lambda}$ and any $m$,
$\operatorname{Pr}\left[m=\hat{m} \mid \operatorname{KeyGen}\left(1^{\lambda}\right) \rightarrow(\mathbf{S K}, \mathbf{P K}) ; \operatorname{Enc}(\mathbf{P K}, m) \rightarrow c t ; \operatorname{Dec}(\mathbf{S K}, c t) \rightarrow \hat{m}\right]=1$.
(cf. [19-21])

### 2.2 IND-CPA Security of PKE

We prove here the security of indistinguishability against chosen-plaintext attacks is defined by the following experimental algorithm $\operatorname{Exp} \mathrm{ExEF}, \mathbf{A}_{\text {ind-cpa }}^{\text {ind }}$, where $\mathbf{A}$ is any given PPT algorithm.

$$
\begin{aligned}
& \operatorname{Exp}_{\mathrm{PKE}, \mathbf{A}}^{\mathrm{ind}-\mathrm{cpa}}\left(1^{\lambda}\right) \\
& \left.\quad(\mathbf{S K}, \mathbf{P K}) \leftarrow \operatorname{KeyGen}\left(1^{\lambda}\right)\right) ;\left(m_{0}, m_{1}\right) \leftarrow \mathbf{A}(\mathbf{P K}) \\
& b \in_{R}\{0,1\} ; c t \leftarrow \operatorname{Enc}\left(\mathbf{P K}, m_{b}\right) ; b^{\prime} \leftarrow \mathbf{A}(c t) \\
& \text { If } b=b^{\prime} \text { then return } 1 \text { else return } 0
\end{aligned}
$$

The advantage of $\mathbf{A}$ over PKE is defined as

$$
\operatorname{Adv}_{\mathrm{PKE}, \mathbf{A}}^{\text {ind-cpa }}(\lambda) \stackrel{\text { def }}{=}\left|\operatorname{Pr}\left[\operatorname{Exp}_{\mathrm{PKE}, \mathbf{A}}^{\text {ind-cpa }}\left(1^{\lambda}\right)=1\right]-(1 / 2)\right| .
$$

PKE is said to be IND-CPA secure if, for any PPT algorithm $\mathbf{A}, \mathbf{A d v}_{\mathrm{PKE}, \mathbf{A}}^{\text {ind-cpa }}(\lambda)$ is negligible in $\lambda$ (cf. [18, 19]).

## 3 Finsler encryption

### 3.1 Finsler space

Generally, Finsler space $(M, F)$ over the set of real numbers $\mathbb{R}$ is defined as a pair consisting of a smooth $n$-dimensional manifold $M$ and a scalar function $F$ on its tangent bundle $T M([1-6])$. Let $x=\left(x^{1}, \cdots, x^{n}\right)$ be the coordinate of the base manifold $M$ and $y=\left(y^{1}, \cdots, y^{n}\right)$ the coordinate of a tangent vector $y$ on $T_{x} M . F=F(x, y)$ is called the Finsler metric or the fundamental function and plays role giving the norm $\|y\|$ of a tangent vector $y$. The Finsler metric $F(x, y)$ determines everything in the space. The metric tensor $g_{i j}(x, y)$ which is very important quantity is calculated from $F(x, y)$ as follows:

$$
\begin{gathered}
g_{i j}(x, y):=\frac{1}{2} \frac{\partial^{2} F^{2}}{\partial y^{i} \partial y^{j}}, \\
\|y\|_{x}=F(x, y)=\sqrt{\sum_{i, j} g_{i j}(x, y) y^{i} y^{j}},(i, j=1, \cdots, n) .
\end{gathered}
$$

We use the asymmetric property of linear parallel displacement of tangent vectors to construct a new public key encryption.

Necessary objects(See Appendix (2),(3),(4))
(1) Metric tensor field $g_{i j}(x, y)$,
(2) Nonlinear connection $N_{j}^{i}(x, y)$,
(3) Horizontal connection $F_{r j}^{i}(x, y)$,
(where the indices $i, j, r=1,2, \cdots, n=\operatorname{dim} M$ )
(4) Geodesic $c=c(t)$
(5) Linear parallel displacement (LPD) $\Pi_{c}$ on $c$ is constructed by the solution of the following differential equations:

$$
(\star) \frac{d v^{i}}{d t}+\sum_{j, r} F_{j r}^{i}(c, \dot{c}) v^{j} \dot{c}^{r}=0 \quad\left(\dot{c}^{r}=\frac{d c^{r}}{d t}\right)
$$

and we call the linear map $\Pi_{c}: v\left(t_{0}\right) \in T_{p} M \longrightarrow v\left(t_{1}\right) \in T_{q} M$ a linear parallel displacement along $c([7,14-17])$.
(6) The energy $E(v)$ of a vector $v=\left(v^{1}, \cdots, v^{n}\right)$ on $c$ :

$$
E(v):=\sum_{i, j} g_{i j}(c, \dot{c}) v^{i} v^{j}
$$

## Example.

We introduce 2-dimensional Finsler space as follows (i.e. the case $n=2$ ) (cf. [8-10]):

$$
M:=\mathbb{R}^{2}
$$

$(\star \star) F(x, y, \dot{x}, \dot{y})=\sqrt{a^{2} \dot{x}^{2}+b^{2} \dot{y}^{2}}-h_{1} x \dot{x}-h_{2} y \dot{y}\left(a, b, h_{1}, h_{2}:\right.$ positive constant $)$,
where $(x, y)$ is the coordinate of the base manifold $M$, and $(\dot{x}, \dot{y})$ is the coordinate of $T_{(x, y)} M$, namely, $x=x^{1}, y=x^{2}, \dot{x}=y^{1}, \dot{y}=y^{2}$.

Geodesics in this Finsler space are any straight lines. So we choose a geodesic as follows

$$
c_{m}(t)=\left(c^{1}(t), c^{2}(t)\right)=\left(\frac{1}{a \sqrt{1+m^{2}}} t, \frac{m}{b \sqrt{1+m^{2}}} t\right)\left(y=\frac{a m}{b} x\right)
$$

And the linear transformation $C(\tau)$ on $T_{p} M(p$ : start point $)$ is

$$
C(\tau):=\left(\begin{array}{cc}
\tau & -1 \\
1 & \tau
\end{array}\right)
$$

Then we have 7 parameters $\left(a, b, h_{1}, h_{2}, m, t_{0}, t_{1}\right)$, where $t_{0}, t_{1}$ mean the start point and the end point of the linear parallel displacement on the geodesic $c$, respectively. In this case the linear parallel displacement $\Pi_{c_{m}}(t)$ is the solution of $(\star)$ as follows

$$
\Pi_{c_{m}}(t)=\left(\begin{array}{cc}
B_{1}^{1} & B_{2}^{1} \\
B_{1}^{2} & B_{2}^{2}
\end{array}\right) \quad(\text { See Appendix }(5))
$$

and the energy equation $E\left(v_{1}\right)$ is

$$
E\left(v_{1}\right):=<v_{1}, v_{1}>_{\dot{c}}=\sum_{i, j} g_{i j}(c, \dot{c}) v_{1}^{i} v_{1}^{j}={ }^{t} v_{1} G v_{1}
$$

where $G=\left(\begin{array}{ll}g_{11} & g_{12} \\ g_{21} & g_{22}\end{array}\right)$,

$$
\begin{aligned}
& g_{11}=\frac{1}{a^{2} b^{2}\left(m^{2}+1\right)^{2}}\left(b^{2} m^{4} a^{4}+b^{2} a^{4}+2 b^{2} m^{2} a^{4}\right. \\
& \left.-\left(h_{2} m^{4} a^{4}+3 b^{2} h_{1} m^{2} a^{2}+2 b^{2} h_{1} a^{2}\right) t+\left(b^{2} h_{1}^{2}+b^{2} h_{1}^{2} m^{2}\right) t^{2}\right) \\
& g_{12}=-\frac{\left(h_{2} a^{2} m+b^{2} h_{1} m^{3}\right) t-\left(h_{1} h_{2} m^{3}+h_{1} h_{2} m\right) t^{2}}{a b\left(m^{2}+1\right)^{2}} \\
& g_{21}=g_{12}, \\
& g_{22}=\frac{1}{a^{2} b^{2}\left(m^{2}+1\right)^{2}}\left(a^{2} m^{4} b^{4}+a^{2} b^{4}+2 a^{2} m^{2} b^{4}\right. \\
& \left.-\left(h_{1} b^{4}+2 a^{2} h_{2} m^{4} b^{2}+3 a^{2} h_{2} m^{2} b^{2}\right) t+\left(a^{2} h_{2}^{2} m^{4}+a^{2} h_{2}^{2} m^{2}\right) t^{2}\right) .
\end{aligned}
$$

However, the components $B_{1}^{1}, B_{2}^{1}, B_{1}^{2}, B_{2}^{2}$ are expressed by rationalization as follows:
Rationalization of Forms: For new parameters $l$ and $\tau$ or $t_{2}$, they are changing as follows:.

$$
\begin{aligned}
l^{2} & :=a^{2} b^{2}\left(1+m^{2}\right)-\left(b^{2} h_{1}+a^{2} h_{2} m^{2}\right) t_{0} \\
& \tau^{2}\left(\text { or } t_{2}^{2}\right):=l^{2}-\left(b^{2} h_{1}+a^{2} h_{2} m^{2}\right) t
\end{aligned}
$$

where $l$ must be elected as $t_{0}$ is a rational number. The methods of Rationalization, however, are many (See $\S 3.5,2)$.

### 3.2 KeyGen, Enc and Dec of Finsler Encryption

The description hereafter is under the assumption that a real number is approximately represented with a rational number that is a ratio of the form (a $\lambda$-bit integer) $/(\mathrm{a} \lambda$-bit integer). Our Finsler encryption scheme FE consists of three polynomial-time (in $\lambda$ ) algorithms KeyGen, Enc and Dec(cf. [11-13]).
KeyGen( $1^{\lambda}$ )
Step1. $c(t)$ : a geodesic, $p\left(t_{0}\right)$ : start point, $q\left(t_{1}\right)$ : end point
Step2. $v$ : a vector in $\mathbb{Z}_{+}^{n}$ (a plaintext), $d v$ : a positive difference vector, $v_{0}=$ $\left(v_{0}^{i}\right)=v+d v$
Step3. $v_{1}=C(\tau) v_{0}(C(\tau)$ is a regular matrix)
Step4. $v_{2}=\Pi_{c}\left(t_{2}\right) v_{1}\left(\Pi_{c}\left(t_{2}\right)\right.$ is the matrix of LPD)
Step5. $E\left(v_{1}\right)=E\left(v_{2}\right)=\sum_{i=0}^{n} E_{i}$ where $E_{1}, \ldots, E_{n} \in_{R} \mathbb{Q}\left[v_{0}, \tau, t_{2}\right], E_{0}:=E\left(v_{1}\right)-$ $\sum_{i=1}^{n} E_{i}$ (because $E\left(v_{1}\right)$ is preserved by LPD)
Step6. $E\left(v_{1}\right)=E\left(v_{2}\right)=\sum_{i=0}^{n} \frac{E_{i}}{f_{i} v_{0}^{i}} f_{i} v_{0}^{i}$ where $f_{0}, \ldots, f_{n} \in_{R} \mathbb{Q}_{+} ; v_{0}^{0}=1$
Step 7. $V_{3}=\Pi_{c}(\tau)^{t}\left(\frac{E_{1}}{f_{1} v_{0}^{1}}, \cdots, \frac{E_{n}}{f_{n} v_{0}^{n}}\right)={ }^{t}\left(V_{3}^{1}, \cdots, V_{3}^{n}\right)$
Step 8. $\left(\frac{E_{0}}{f_{0}}, V_{3}^{1}, \cdots, V_{3}^{n}\right)$ : an encryption key
$\mathbf{P K}:=\left(\frac{E_{0}}{f_{0}}, V_{3}^{1}, \cdots, V_{3}^{n}\right), \mathbf{S K}:=\left\{\left(f_{0}, \cdots, f_{n}\right), \Pi_{c}\left(t_{2}\right), E\left(v_{1}\right)\right\}$
Return ( $\mathbf{P K}, \mathbf{S K}$ ).

Note that, for the above $\mathbf{P K}$ and $\mathbf{S K}$, the set of plaintexts should be $\mathbb{Z}_{+}^{n}$ and the set of ciphertexts should be a certain subset $C y$ of $\mathbb{Q}^{(n+1)^{2}}$.

Next, we obtain the ciphertext $c t$ of a plaintext $v=\left(v^{i}\right)$ by using $1+(n+1) k$ parameters, where $k>n$ as follows:
$\operatorname{Enc}(\mathbf{P K}, v) \quad / / \mathbf{P K}=\left(\frac{E_{0}}{f_{0}}, V_{3}^{1}, \cdots, V_{3}^{n}\right)$
Step1. $k$ : Choose a natural number $k$ which is above $n$.
Step2. $\alpha, \beta_{1}, \cdots, \beta_{(n+1) k}$ : Each other different rational numbers
Step3. $\left\{v, \tau \leftarrow \alpha, t_{2} \leftarrow \beta_{1}\right\} \rightarrow e_{1}=\frac{1}{k}\left(\frac{E_{0}}{f_{0}}, v_{3}^{1}, \cdots, v_{3}^{n}\right)$
$\vdots \quad \vdots$
$\left\{v, \tau \leftarrow \alpha, t_{2} \leftarrow \beta_{(n+1) k}\right\} \rightarrow e_{(n+1) k}=\frac{1}{k}\left(\frac{E_{0}}{f_{0}}, v_{3}^{1}, \cdots, v_{3}^{n}\right)$
Step4. $c t_{1}:=\sum_{i=1}^{k} e_{i}, c t_{2}:=\sum_{i=k+1}^{2 k} e_{i}, \cdots, c t_{n+1}:=\sum_{i=n k+1}^{(n+1) k} e_{i}$
Step5. $c t=\left\{c t_{1}, \cdots, c t_{n+1}\right\}$ : a ciphertext
Return $c t$.

Finally, we can decrypt $c t$ and recover the plaintext $v$ by using the secret key $\mathbf{S K}=\left\{\left(f_{0}, \cdots, f_{n}\right), \Pi_{c}\left(t_{2}\right), E\left(v_{1}\right)\right\}$ as follows:
$\operatorname{Dec}(\mathbf{S K}, c t) \quad / / \mathbf{S K}:=\left\{\left(f_{0}, \cdots, f_{n}\right), \Pi_{c}\left(t_{2}\right), E\left(v_{1}\right)\right\}$
Step1. $\left(f_{0}, f_{1}, \cdots, f_{n}\right) \rightarrow s x:=\left(f_{0}, f_{1} X_{1}, \cdots, f_{n} X_{n}\right)$
Step2. $\overline{c t}_{1}:=\left(c t_{1}[[1]], \Pi_{c}^{-1}(\tau)^{t}\left(c t_{1}[[2]], \cdots, c t_{1}[[n+1]]\right)\right)$
$\vdots \quad \vdots \quad \vdots$
$\bar{c} t_{n+1}:=\left(c t_{n+1}[[1]], \Pi_{c}^{-1}(\tau)^{t}\left(c t_{n+1}[[2]], \cdots, c t_{n+1}[[n+1]]\right)\right)$
Step3. $E X_{1}:=<s x, \bar{c}_{1}>, \cdots, E X_{n+1}:=<s x, \bar{c} t_{n+1}>$
Step4.

$$
(I)\left\{\begin{array}{c}
E X_{1}=E X_{n+1} \\
\vdots \quad \vdots \quad \vdots \\
E X_{n}=E X_{n+1}
\end{array}\right.
$$

(System of simultaneous linear equations with $X_{1}, \cdots, X_{n}$ )
Step5. $\bar{X}_{1}, \cdots, \bar{X}_{n}$ : formal solution of simultaneous linear equations $(I)$ with unknown $\tau$
Step6. $\left.E X_{1}\right|_{X_{1} \leftarrow \bar{X}_{1}, \cdots, X_{n} \leftarrow \bar{X}_{n}}-\left.E\left(v_{1}\right)\right|_{v_{0}^{1} \leftarrow \bar{X}_{1}, \cdots, v_{0}^{n} \leftarrow \bar{X}_{n}}=0$
(algebraic equation of $\tau$ )
Step7. Solve the rational number solution $\tau=\alpha$ and substitute them for $\bar{X}_{1}, \cdots, \bar{X}_{n}$

$$
v_{0}=\left(v_{0}^{1}, \cdots, v_{0}^{n}\right)=\left(\left.\bar{X}_{1}\right|_{\tau \leftarrow \alpha}, \cdots,\left.\bar{X}_{n}\right|_{\tau \leftarrow \alpha}\right)
$$

Step8. Finally, obtain the plaintext $v$ as follows

$$
v=v_{0}-d v
$$

Return $v$.

## Example

In the Finsler space ( $\left(\star \star\right.$ ) in p.4, we put $\left(a, b, h_{1}, h_{2}, m, t_{0}, t_{1}\right)=\left(1,1,1,1,1, \frac{1}{2}, 1\right)$, then

$$
\begin{gathered}
\text { SK: } \\
\qquad \begin{array}{c}
\left(f_{0}, f_{1}, f_{2}\right):=\left(m h_{1}, a t_{0} h_{2}, b t_{1} h_{2}^{2}\right)=\left(1, \frac{1}{2}, 1\right) \\
\Pi_{c_{m}}(\tau)=\left(\begin{array}{cc}
\frac{\tau+1}{2 \tau^{2}} & -\frac{\tau-1}{2 \tau^{2}} \\
-\frac{\tau-1}{2 \tau^{2}} & \frac{\tau+1}{2 \tau^{2}}
\end{array}\right) . \\
E\left(v_{1}\right)=G\left(v_{1}, v_{1}\right)={ }^{t} v_{2} G v_{2}={ }^{t} v_{1}{ }^{t} \Pi_{c} G \Pi_{c} v_{1}={ }^{t} v_{0}{ }^{t} C{ }^{t} \Pi_{c} G \Pi_{c} C v_{0} \\
=\frac{1}{8}\left(3 \tau^{2}-2 \tau+3\right)\left(v_{0}^{1}\right)^{2}+\frac{1}{4}\left(1-\tau^{2}\right) v_{0}^{1} v_{0}^{2}+\frac{1}{8}\left(3 \tau^{2}+2 \tau+3\right)\left(v_{0}^{2}\right)^{2}
\end{array}
\end{gathered}
$$

PK:

$$
\mathbf{P K}=\left(\frac{E_{0}}{f_{0}}, V_{3}^{1}, V_{3}^{2}\right) \quad(\text { See Appendix }(6))
$$

From $E\left(v_{1}\right)=\left(\frac{E_{0}}{f_{0}}\right) f_{0}+\left(\frac{E_{1}}{f_{1} v_{0}^{1}}\right) f_{1} v_{0}^{1}+\left(\frac{E_{2}}{f_{2} v_{0}^{2}}\right) f_{2} v_{0}^{2} \rightarrow V=\left(\frac{E_{1}}{f_{1} v_{0}^{1}}, \frac{E_{2}}{f_{2} v_{0}^{2}}\right)$, $\left(V_{3}^{1}, V_{3}^{2}\right)=V_{3}=\Pi_{c}(\tau) V$. Then, PK is obtained.

## 4 Security Analysis

### 4.1 Strength of SK

In this section, the strength of each secret key $\left(f_{0}, \cdots, f_{n}\right), \Pi_{c}\left(t_{2}\right)$ and $E\left(v_{1}\right)$ is stated about the security from a viewpoint of a calculation amount.

1. $\left(f_{0}, \cdots, f_{n}\right)$ : Each component is arbitrary rational number.
2. $\Pi_{c}\left(t_{2}\right)$ : The regular matrix $\Pi_{c}\left(t_{2}\right)$ is derived from a certain simultaneous differential equations. The differential equations are made by the Finsler metric function $F$. Therefore nobody knows the equations without $F($ LPD problem, see Appendix (1)). Further, in general, the linear parallel displacement of a Finsler space satisfying asymmetric property is asymmetric, namely,

$$
\Pi_{c}^{-1} \neq \Pi_{c^{-1}}
$$

is satisfied. This means that any informations of $\Pi_{c}^{-1}$ used in the algorithm of decryption are not obtained from $\Pi_{c^{-1}}$, where $c^{-1}$ is the inverse curve of $c . \Pi_{c}$ is an one-way function(cf.[8, 9]).
3. $E\left(v_{1}\right)$ : The energy of the vector $v_{1}$. This equation is directly affected by the matrix $C(\tau)$. If you replace $C(\tau)$ for the following matrix

$$
\left(\begin{array}{cr}
\tau & 1 \\
\tau-1 & 1
\end{array}\right)
$$

then the expression of $E\left(v_{1}\right)$ is changed as follows

$$
E\left(v_{1}\right)=\frac{1}{8}\left(4 \tau^{2}-4 \tau+3\right)\left(v_{0}^{1}\right)^{2}+\frac{1}{2}(2 \tau-1) v_{0}^{1} v_{0}^{2}+\frac{1}{2}\left(v_{0}^{2}\right)^{2} .
$$

Therefore nobody knows three coefficients $\frac{1}{8}\left(4 \tau^{2}-4 \tau+3\right), \frac{1}{2}(2 \tau-1)$ and $\frac{1}{2}$ without recognition of $C(\tau) . C(\tau)$ is completely arbitrary regular matrix.
On the other hand, the matrix $E$ is composed by three regular matrixes $C(\tau), \Pi_{c}(\tau)$ and $G$, namely,

$$
E={ }^{t} C^{t} \Pi_{c} G \Pi_{c} C,\left(E\left(v_{1}\right)={ }^{t} v_{0} E v_{0}\right),
$$

where $G$ is called the Finsler metric tensor field. If $E$ can be decomposed, then the attacker can get $C(\tau), \Pi_{c}(\tau)$ and $G$. Then the attacker can decrypt any ciphertext. However, to decompose $E$ to 5 -pieces regular matrix ${ }^{t} C,{ }^{t} \Pi_{c}, G, \Pi_{c}, C$ is computationally hard under the assumption of Mapping-Decomposition Problem(cf.[12, 13]).

### 4.2 Strength of PK

In the encryption algorithm, the ciphertext ct is made from $(1+(n+1) k)$ parameters $\beta_{i}$ at Step3. Each component $c t_{i}(i=1, \cdots, n+1)$ of $c t=\left\{c t_{1}, \cdots, c t_{n+1}\right\}$ is made by $k$-pieces parameters $\beta_{j}(j=(i-1) k+1, \cdots, i k)$. Thus, algebraic equations made by the public key PK and $c t$ have the property that the number of its unknown variables is more than ones of equations. For example, in the former case $\mathbf{P K}=\left(\frac{E_{0}}{f_{0}}, V_{3}^{1}, V_{3}^{2}\right)$, if $k=2$, we have the following equation:
If a ciphertext $c t=\left(c t_{1}, c t_{2}, c t_{3}\right)=\left(c t_{11}, c t_{12}, c t_{13}, c t_{21}, c t_{22}, c t_{23}, c t_{31}, c t_{32}, c t_{33}\right)$, $c t_{1}=\left.\left(c t_{11}, c t_{12}, c t_{13}\right) \leftarrow \frac{1}{2}\left(\frac{E_{1}}{f_{0}}, V_{3}^{1}, V_{3}^{2}\right)\right|_{t_{2} \leftarrow \beta_{1}}+\left.\frac{1}{2}\left(\frac{E_{1}}{f_{0}}, V_{3}^{1}, V_{3}^{2}\right)\right|_{t_{2} \leftarrow \beta_{2}}$
$c t_{2}=\left.\left(c t_{21}, c t_{22}, c t_{23}\right) \leftarrow \frac{1}{2}\left(\frac{E_{1}}{f_{0}}, V_{3}^{1}, V_{3}^{2}\right)\right|_{t_{2} \leftarrow \beta_{3}}+\left.\frac{1}{2}\left(\frac{E_{1}}{f_{0}}, V_{3}^{1}, V_{3}^{2}\right)\right|_{t_{2} \leftarrow \beta_{4}}$
$c t_{3}=\left.\left(c t_{31}, c t_{32}, c t_{33}\right) \leftarrow \frac{1}{2}\left(\frac{E_{1}}{f_{0}}, V_{3}^{1}, V_{3}^{2}\right)\right|_{t_{2} \leftarrow \beta_{5}}+\left.\frac{1}{2}\left(\frac{E_{1}}{f_{0}}, V_{3}^{1}, V_{3}^{2}\right)\right|_{t_{2} \leftarrow \beta_{6}}$
for example, from $c t_{1}$, we have following three equations
$c t_{11}=\left.\frac{1}{2} \frac{E_{1}}{f_{0}}\right|_{t_{2} \leftarrow \beta_{1}}+\left.\frac{1}{2} \frac{E_{1}}{f_{0}}\right|_{t_{2} \leftarrow \beta_{2}}, c t_{12}=\left.\frac{1}{2} V_{3}^{1}\right|_{t_{2} \leftarrow \beta_{1}}+\left.\frac{1}{2} V_{3}^{1}\right|_{t_{2} \leftarrow \beta_{2}}, c t_{13}=\left.\frac{1}{2} V_{3}^{2}\right|_{t_{2} \leftarrow \beta_{1}}+$ $\left.\frac{1}{2} V_{3}^{2}\right|_{t_{2} \leftarrow \beta_{2}}$.
From $\mathrm{ct}_{2}$,
$c t_{21}=\left.\frac{1}{2} \frac{E_{1}}{f_{0}}\right|_{t_{2} \leftarrow \beta_{3}}+\left.\frac{1}{2} \frac{E_{1}}{f_{0}}\right|_{t_{2} \leftarrow \beta_{4}}, c t_{22}=\left.\frac{1}{2} V_{3}^{1}\right|_{t_{2} \leftarrow \beta_{3}}+\left.\frac{1}{2} V_{3}^{1}\right|_{t_{2} \leftarrow \beta_{4}}, c t_{23}=\left.\frac{1}{2} V_{3}^{2}\right|_{t_{2} \leftarrow \beta_{3}}+$ $\left.\frac{1}{2} V_{3}^{2}\right|_{t_{2} \leftarrow \beta_{4}}$
From $\mathrm{ct}_{3}$,
$c t_{31}=\left.\frac{1}{2} \frac{E_{1}}{f_{0}}\right|_{t_{2} \leftarrow \beta_{5}}+\left.\frac{1}{2} \frac{E_{1}}{f_{0}}\right|_{t_{2} \leftarrow \beta_{6}}, c t_{32}=\left.\frac{1}{2} V_{3}^{1}\right|_{t_{2} \leftarrow \beta_{5}}+\left.\frac{1}{2} V_{3}^{1}\right|_{t_{2} \leftarrow \beta_{6}}, c t_{33}=\left.\frac{1}{2} V_{3}^{2}\right|_{t_{2} \leftarrow \beta_{5}}+$ $\left.\frac{1}{2} V_{3}^{2}\right|_{t_{2} \leftarrow \beta_{6}}$

Thus, in total, we have 9 -pieces unknown variables $v_{0}^{1}, v_{0}^{2}, \tau, \beta_{1}, \cdots, \beta_{6}$ and 9 -pieces equations. Here $k$ is known, however. In general, for $c t_{11}$, the attacker must solve the following equation.

$$
\left.\frac{1}{k} \frac{E_{1}}{f_{0}}\right|_{t_{2} \leftarrow \beta_{1}}+\cdots+\left.\frac{1}{k} \frac{E_{1}}{f_{0}}\right|_{t_{2} \leftarrow \beta_{k}}=c t_{11}
$$

is satisfied. Namely, let $\left(v_{0}^{1}, v_{0}^{2}, \tau, k, t_{21}, \cdots, t_{2 k}\right)$ be unknown variables, then the attacker must solve the following equation with $(4+k)$-pieces unknowm variables

$$
\begin{aligned}
& \frac{1}{64 k t_{21}^{4}} \times \\
& \left(t _ { 2 1 } ^ { 6 } \left(3 \tau^{2}\left(v_{0}^{1}\right)^{2}-6 \tau\left(v_{0}^{1}\right)^{2}+3\left(v_{0}^{1}\right)^{2}-6 \tau^{2} v_{0}^{1} v_{0}^{2}+6 v_{0}^{1} v_{0}^{2}+3 \tau^{2}\left(v_{0}^{2}\right)^{2}\right.\right. \\
& \left.+6 \tau\left(v_{0}^{2}\right)^{2}+3\left(v_{0}^{2}\right)^{2}\right) \\
& +t_{21}^{5}\left(-8 \tau^{2}\left(v_{0}^{1}\right)^{2}-8 \tau\left(v_{0}^{1}\right)^{2}+16\left(v_{0}^{1}\right)^{2}-8 \tau^{2} v_{0}^{1} v_{0}^{2}+48 \tau v_{0}^{1} v_{0}^{2}\right. \\
& \left.+8 v_{0}^{1} v_{0}^{2}+16 \tau^{2}\left(v_{0}^{2}\right)^{2}+8 \tau\left(v_{0}^{2}\right)^{2}-8\left(v_{0}^{2}\right)^{2}\right) \\
& +t_{21}^{4}\left(-2 \tau^{2}\left(v_{0}^{1}\right)^{2}+28 \tau\left(v_{0}^{1}\right)^{2}+10\left(v_{0}^{1}\right)^{2}+28 \tau^{2} v_{0}^{1} v_{0}^{2}+24 \tau v_{0}^{1} v_{0}^{2}\right. \\
& \left.-28 v_{0}^{1} v_{0}^{2}+10 \tau^{2}\left(v_{0}^{2}\right)^{2}-28 \tau\left(v_{0}^{2}\right)^{2}-2\left(v_{0}^{2}\right)^{2}\right) \\
& +t_{21}^{3}\left(16 \tau^{2}\left(v_{0}^{1}\right)^{2}+16 \tau\left(v_{0}^{1}\right)^{2}-32\left(v_{0}^{1}\right)^{2}+16 \tau^{2} v_{0}^{1} v_{0}^{2}-96 \tau v_{0}^{1} v_{0}^{2}\right. \\
& \left.-16 v_{0}^{1} v_{0}^{2}-32 \tau^{2}\left(v_{0}^{2}\right)^{2}-16 \tau\left(v_{0}^{2}\right)^{2}+16\left(v_{0}^{2}\right)^{2}\right) \\
& +t_{21}^{2}\left(44 \tau^{2}\left(v_{0}^{1}\right)^{2}-40 \tau\left(v_{0}^{1}\right)^{2}+68\left(v_{0}^{1}\right)^{2}-40 \tau^{2} v_{0}^{1} v_{0}^{2}+48 \tau v_{0}^{1} v_{0}^{2}\right. \\
& \left.+40 v_{0}^{1} v_{0}^{2}+68 \tau^{2}\left(v_{0}^{2}\right)^{2}+40 \tau\left(v_{0}^{2}\right)^{2}+44\left(v_{0}^{2}\right)^{2}\right) \\
& +24 \tau^{2}\left(v_{0}^{1}\right)^{2}-48 \tau\left(v_{0}^{1}\right)^{2}+24\left(v_{0}^{1}\right)^{2}-48 \tau^{2} v_{0}^{1} v_{0}^{2}+48 v_{0}^{1} v_{0}^{2} \\
& \left.+24 \tau^{2}\left(v_{0}^{2}\right)^{2}+48 \tau\left(v_{0}^{2}\right)^{2}+24\left(v_{0}^{2}\right)^{2}\right)+ \\
& +\cdots \cdots(\text { sum of } \mathrm{k} \text {-terms }) \cdots \cdots+ \\
& +\frac{1}{64 k t_{2 k}^{4}} \times \\
& \left(t _ { 2 k } ^ { 6 } \left(3 \tau^{2}\left(v_{0}^{1}\right)^{2}-6 \tau\left(v_{0}^{1}\right)^{2}+3\left(v_{0}^{1}\right)^{2}-6 \tau^{2} v_{0}^{1} v_{0}^{2}+6 v_{0}^{1} v_{0}^{2}+3 \tau^{2}\left(v_{0}^{2}\right)^{2}\right.\right. \\
& \left.+6 \tau\left(v_{0}^{2}\right)^{2}+3\left(v_{0}^{2}\right)^{2}\right) \\
& +t_{2 k}^{5}\left(-8 \tau^{2}\left(v_{0}^{1}\right)^{2}-8 \tau\left(v_{0}^{1}\right)^{2}+16\left(v_{0}^{1}\right)^{2}-8 \tau^{2} v_{0}^{1} v_{0}^{2}+48 \tau v_{0}^{1} v_{0}^{2}\right. \\
& \left.+8 v_{0}^{1} v_{0}^{2}+16 \tau^{2}\left(v_{0}^{2}\right)^{2}+8 \tau\left(v_{0}^{2}\right)^{2}-8\left(v_{0}^{2}\right)^{2}\right) \\
& +t_{2 k}^{4}\left(-2 \tau^{2}\left(v_{0}^{1}\right)^{2}+28 \tau\left(v_{0}^{1}\right)^{2}+10\left(v_{0}^{1}\right)^{2}+28 \tau^{2} v_{0}^{1} v_{0}^{2}+24 \tau v_{0}^{1} v_{0}^{2}\right. \\
& \left.-28 v_{0}^{1} v_{0}^{2}+10 \tau^{2}\left(v_{0}^{2}\right)^{2}-28 \tau\left(v_{0}^{2}\right)^{2}-2\left(v_{0}^{2}\right)^{2}\right) \\
& +t_{2 k}^{3}\left(16 \tau^{2}\left(v_{0}^{1}\right)^{2}+16 \tau\left(v_{0}^{1}\right)^{2}-32\left(v_{0}^{1}\right)^{2}+16 \tau^{2} v_{0}^{1} v_{0}^{2}-96 \tau v_{0}^{1} v_{0}^{2}\right. \\
& \left.-16 v_{0}^{1} v_{0}^{2}-32 \tau^{2}\left(v_{0}^{2}\right)^{2}-16 \tau\left(v_{0}^{2}\right)^{2}+16\left(v_{0}^{2}\right)^{2}\right) \\
& +t_{2 k}^{2}\left(44 \tau^{2}\left(v_{0}^{1}\right)^{2}-40 \tau\left(v_{0}^{1}\right)^{2}+68\left(v_{0}^{1}\right)^{2}-40 \tau^{2} v_{0}^{1} v_{0}^{2}+48 \tau v_{0}^{1} v_{0}^{2}\right. \\
& \left.+40 v_{0}^{1} v_{0}^{2}+68 \tau^{2}\left(v_{0}^{2}\right)^{2}+40 \tau\left(v_{0}^{2}\right)^{2}+44\left(v_{0}^{2}\right)^{2}\right) \\
& +24 \tau^{2}\left(v_{0}^{1}\right)^{2}-48 \tau\left(v_{0}^{1}\right)^{2}+24\left(v_{0}^{1}\right)^{2}-48 \tau^{2} v_{0}^{1} v_{0}^{2}+48 v_{0}^{1} v_{0}^{2} \\
& \left.+24 \tau^{2}\left(v_{0}^{2}\right)^{2}+48 \tau\left(v_{0}^{2}\right)^{2}+24\left(v_{0}^{2}\right)^{2}\right) \\
& =c t_{11} .
\end{aligned}
$$

Further, from $c t_{12}$ and $c t_{13}$,

$$
\left.\frac{1}{k} V_{3}^{1}\right|_{t_{2} \leftarrow \beta_{1}}+\cdots+\left.\frac{1}{k} V_{3}^{1}\right|_{t_{2} \leftarrow \beta_{k}}=c t_{12}
$$

$$
\left.\frac{1}{k} V_{3}^{2}\right|_{t_{2} \leftarrow \beta_{1}}+\cdots+\left.\frac{1}{k} V_{3}^{2}\right|_{t_{2} \leftarrow \beta_{k}}=c t_{13}
$$

are satisfied. After all, $(4+k)$-pieces $\left(v_{0}^{1}, v_{0}^{2}, \tau, k, t_{21}, \cdots, t_{2 k}\right)$ are unknown variables. Next, from $c t_{2}=\left(c t_{21}, c t_{22}, c t_{23}\right)$, according to the same manner, we have $(4+k)$-pieces $\left(v_{0}^{1}, v_{0}^{2}, \tau, k, t_{2(k+1)}, \cdots, t_{2(2 k)}\right)$ unknown variables and, further from $c t_{3}=\left(c t_{31}, c t_{32}, c t_{33}\right)$, we have $(4+k)$-pieces $\left(v_{0}^{1}, v_{0}^{2}, \tau, k, t_{2(2 k+1)}, \cdots, t_{2(3 k)}\right)$ unknown variables. Totally, we have $(4+3 k)$-pieces $\left(v_{0}^{1}, v_{0}^{2}, \tau, k, t_{1}, \cdots, t_{2(3 k)}\right)$ unknown variables. $3 k \geq 6$ is true if $k \geq 2$, so unknown variables number satisfies $4+3 k \geq 10$ if $k \geq 2$. The other side, equation's number is 9 , obviously. This means that the simultaneous equations made by 9 -pieces algebraic equations are not able to be solved because these are underdetermined on rational numbers (SUS problem). In general, if an $n$-dimensional vector $v$ is a plaintext, then the unknown variables are $n+2+(n+1) k$-pieces because the components of $v$ is $n$-pieces and other parameters are $(2+3 k)$-pieces $\left\{k, \tau, \beta_{1}, \cdots, \beta_{(n+1) k}\right\}$. Therefore the equation's number is $(n+1)^{2}$ and if $k \geq n+1$ is satisfied then $n+2+(n+1) k>(n+1)^{2}$ is true(Underdetermined system)([13]).

### 4.3 Length of SK

Finally, we remark the length of the secret key $\mathbf{S K}=\left\{\left(f_{0}, \cdots, f_{n}\right), \Pi_{c}(\tau), E\left(v_{1}\right)\right\}$. The length depend on the dimension $n$.
$\left(f_{0}, \cdots, f_{n}\right): n+1$-pieces arbitrary rational numbers.

$$
\begin{gathered}
\Pi_{c}(\tau)=\left(\begin{array}{ccc}
\frac{a_{11}(\tau)}{b_{11}(\tau)} & \cdots & \frac{a_{1 n}(\tau)}{b_{1 n}(\tau)} \\
\vdots & \ddots & \vdots \\
\frac{a_{n 1}(\tau)}{b_{n 1}(\tau)} & \cdots & \frac{a_{n n}(\tau)}{b_{n n}(\tau)}
\end{array}\right) \\
E\left(v_{1}\right)=\sum_{i=1}^{n} \frac{a_{i}(\tau)}{b_{i}(\tau)}\left(v_{0}^{i}\right)^{2}+\sum_{i<j, i=1, \cdots, n-1, j=2, \cdots, n} \frac{c_{i j}(\tau)}{d_{i j}(\tau)} v_{0}^{i} v_{0}^{j},
\end{gathered}
$$

where $a_{i j}(\tau)$ and $b_{i j}(\tau)$ are polynomials of $\tau$ of degree $p$ and $q$ and $a_{i}(\tau), b_{i}(\tau), c_{i j}(\tau)$ and $d_{i j}(\tau)$ are polynomials of $\tau$ of degree $r, s, t$ and $w$. Therefore all of integer as coefficients of all polynomial $a_{i j}, b_{i j}, a_{i}, b_{i}, c_{i j}, d_{i j}$ is $2 n+(p+1) n^{2}+(q+1) n^{2}+$ $r n+s n+t{ }_{n} C_{2}+w{ }_{n} C_{2} \approx \alpha n^{2}+\beta n+\gamma(\alpha, \beta, \gamma:$ certain natural numbers). Thus we can recognize that the length of the secret key increases linearly to square of the dimension $n$ (i.e. $\mathcal{O}\left(n^{2}\right)$ ).

### 4.4 IND-CPA Security

We prove here the IND-CPA security of FE under the LPD assumption.

To construct the public key PK and the secret key SK of FE needs some parameters. In the case of the example of Appendix, the values ( $a, b, h_{1}, h_{2}, m, t_{0}, t_{1}, \alpha, f_{0}, f_{1}, f_{2}$ ) and the matrix $C(\alpha), \Pi_{c}(\alpha)$ are needed. In addition the energy form $E\left(v_{1}\right)$ is also needed. Especially, for PK, certain methods of rationalization and splitting are essentially needed. The values $\left(a, b, h_{1}, h_{2}, m, t_{0}, t_{1}, \alpha, f_{0}, f_{1}, f_{2}, C(\alpha)\right)$ and the method of splitting of $E\left(v_{1}\right)$ decides PK, and the values ( $a, b, h_{1}, h_{2}, m, t_{0}, t_{1}, \alpha$ ) and the method of rationalization of $t_{2}$ decides $\Pi_{c}(\alpha)$.

Here, we state the LPD assumption([8-10]).

## Computational problem of linear parallel displacement (LPD Problem)

Suppose that each variable is quantized with $\lambda$-bit uniformly. (Note that $\lambda$ is the security parameter. ) Let ( $M, F$ ) be a Finsler space and $p, q$ be points on $M$. For a geodesic $c$ from $p$ to $q$, the problem is stated as the computational problem to find values of the parameters of linear parallel displacement along $c$ from $T_{p} M$ to $T_{q} M$, where $T_{p} M, T_{q} M$ are tangent spaces at $p, q$ respectively. Formally,

- Input: ( $p, q, c$ )
- Output: A matrix $\Pi_{c}(\alpha)$ of linear parallel displacement along $c$ from $T_{p} M$ to $T_{q} M$. LPD Assumption
For a fixed Finsler space with $H_{j}^{i} \neq 0$ (See Appendix (2)), there exists no polynomial time algorithm to solve a random instance of LPD problem.

We will prove the following theorem.
Theorem 41 FE has the IND-CPA security under LPD assumption.
Propositions for Theorem. First we consider the following problem;
Problem Let $\Pi_{c}(\alpha)$ and $\Pi_{c}\left(\alpha^{\prime}\right)$ be the matrices of the linear parallel displacement made by the values ( $a, b, h_{1}, h_{2}, m, t_{0}, t_{1}, \alpha$ ) and ( $a, b, h_{1}, h_{2}, m, t_{0}, t_{1}, \alpha^{\prime}$ ), respectively. Then we distinguish $\Pi_{c}(\alpha)$ and $\Pi_{c}\left(\alpha^{\prime}\right)$, where the method of rationalization is unknown and ( $a, b, h_{1}, h_{2}, m, t_{0}, t_{1}$ ) are same values.

We can state the two matrices in the Problem are indistinguishable under LPD assumption.

Proposition 41 The two matrices in the above Problem are indistinguishable under LPD assumption.

Proof We assume that the two matrices in Problem are capable of being identified. This assumption means that $m$-pieces matrices $\Pi_{c}\left(\alpha_{1}\right), \cdots, \Pi_{c}\left(\alpha_{m}\right)$ which are correspondent to the different $m$ values $\alpha_{1}, \cdots, \alpha_{m}$ are distinguishable.
Now, we have no information of the method of the rationalization of $t_{2}$. Then the general form of $\Pi_{c}(\alpha)$ is put by

$$
\Pi_{c}(\alpha)=\left(\begin{array}{ll}
\frac{a_{11}(\alpha)}{b_{11}(\alpha)} & \frac{a_{12}(\alpha)}{b_{12}(\alpha)} \\
\frac{a_{21}(\alpha)}{b_{21}(\alpha)} & \frac{a_{22}(\alpha)}{b_{22}(\alpha)}
\end{array}\right),
$$

where the forms $a_{11}(\alpha), a_{12}(\alpha), a_{21}(\alpha), a_{22}(\alpha), b_{11}(\alpha), b_{12}(\alpha), b_{21}(\alpha), b_{22}(\alpha)$ are polynomials with respect to unknown value $\alpha$. If the amount of unknown coefficients of $\alpha$ of
all forms $a_{i j}(\alpha), b_{i j}(\alpha)(i, j=1,2)$ are $m$, then all coefficients are solvable under informations of distinguished $m$-pieces matrices $\Pi_{c}\left(\alpha_{1}\right), \cdots, \Pi_{c}\left(\alpha_{m}\right)$. Namely, the general form of $\Pi_{c}(\alpha)$ is obtained. That means that LPD assumption is broken. Therefore this proposition's assertion is true.

Further, we have the following proposition.
Proposition 42 In FE, if parameter values $\left(a, b, h_{1}, h_{2}, m, t_{0}, t_{1}, f_{0}, f_{1}, f_{2}, \alpha\right)$ and the values of entries of the matrix of linear parallel displacement $\Pi_{c}(\alpha)$ are known, then any ciphertext $c t=\left\{c t_{1}, c t_{2}, c t_{3}\right\}$ is solvable. Namely, to decrypt any ciphertext is no need of $E\left(v_{1}\right)$.

Proof Let $c t_{1}=\left(c t_{11}, c t_{12}, c t_{13}\right), c t_{2}=\left(c t_{21}, c t_{22}, c t_{23}\right), c t_{3}=\left(c t_{31}, c t_{32}, c t_{33}\right)$ be the components of the ciphertext $c t$, where all $c t_{i j}(i, j=1,2)$ are rational numbers.
First, respectively, we can obtain $\overline{c t}_{12}, \overline{c t}_{13}, \overline{c t}_{22}, \overline{c t}_{23}, \overline{c t}_{32}, \overline{c t}_{33}$ from $c t_{1}, c t_{2}, c t_{3}$ and $\Pi_{c}(\alpha)$ as follows;
$\binom{\overline{c t}_{12}}{\overline{c t}_{13}}=\Pi_{c}^{-1}(\alpha)\binom{c t_{12}}{c t_{13}},\binom{\overline{c t}}{\overline{c t}_{23}}=\Pi_{c}^{-1}(\alpha)\binom{c t_{22}}{c t_{23}},\binom{\overline{c t}}{\overline{c t}_{33}}=\Pi_{c}^{-1}(\alpha)\binom{c t_{32}}{c t_{33}}$.
Next, we can construct the following simultaneous linear equations of $X_{1}, X_{2}$;

$$
\left\{\begin{aligned}
& c t_{11} f_{0}+\overline{c t}_{12} f_{1} X_{1}+\overline{c t}_{13} f_{2} X_{2}=c t_{31} f_{0}+\overline{c t}_{32} f_{1} X_{1}+\overline{c t}_{33} f_{2} X_{2} \\
& c t_{21} f_{0}+\overline{c t}_{22} f_{1} X_{1}+\overline{c t}_{23} f_{2} X_{2}=c t_{31} f_{0}+\overline{c t}_{32} f_{1} X_{1}+\overline{c t}_{33} f_{2} X_{2}
\end{aligned}\right.
$$

Finally, the solution $X_{1}, X_{2}$ of the above system leads to the plain text $v=\left(v^{1}, v^{2}\right)$. In this algorithm, there is no using of $E\left(v_{1}\right)$.

Proof of Theorem We consider the following game of any given Ppt attacker $\mathbf{A}$ and our FE, (1) to (5), that is in accordance with the experiment $\operatorname{Exp}_{\mathrm{FE}, \mathbf{A}}^{\text {ind-cpa }}\left(1^{\lambda}\right)$.
(1) The challenger sends the public key $\mathbf{P K}$ of FE to the attacker.
(2) The attacker gives two plaintext $m_{0}, m_{1} \in \mathbb{Z}_{+}^{2}$ to the challenger (We denote a message as $m_{i}$ instead of $v_{i}$ to avoid confusion).
(3) The challenger selects $b=0$ or $b=1$ at random.
(4) The challenger selects $\alpha \in \mathbb{Q}_{+}$at random and sends the ciphertext $c t_{b}(\alpha)$ (that is encryption of $m_{b}$ with $\left.\Pi_{c}(\alpha)\right)$ to the attacker.
(5) The attacker returns a guess $b^{\prime}$ to the challenger.

Now we consider another game that is the same as the above procedure (1) to (5) except that a simulated $\operatorname{ct}_{b}\left(\alpha, \alpha^{\prime}\right)$ is used, which is generated using $\Pi_{c}\left(\alpha^{\prime}\right)$ where a random $\alpha^{\prime}$ is sampled independently of $\alpha$, while $E_{0} / f_{0}$ is dependent of $\alpha$. (This is an analogy of the security proof of IND-CPA security of the El Gamal encryption [18, 19]. ) Then $\Pi_{c}(\alpha)$ and $\Pi_{c}\left(\alpha^{\prime}\right)$ are indistinguishable under the LPD assumption because of Proposition 41. Therefore the following relation holds.

$$
\begin{equation*}
\left|\operatorname{Pr}\left[b^{\prime}=b \mid \Pi_{c}(\alpha)\right]-\operatorname{Pr}\left[b^{\prime}=b \mid \Pi_{c}\left(\alpha^{\prime}\right)\right]\right|<\varepsilon \tag{1}
\end{equation*}
$$

On the other hand, $\operatorname{ct}_{b}\left(\alpha, \alpha^{\prime}\right)$ is actually a one-time pad because $\alpha^{\prime}$ is sampled uniformly at random and independently of $\alpha$, and the components of a ciphtertext, except the $E_{0} / f_{0}$, is obtained by multiplying $\Pi_{c}\left(\alpha^{\prime}\right)$. Therefore $\operatorname{Pr}\left[b^{\prime}=b \mid \Pi_{c}\left(\alpha^{\prime}\right)\right]=\frac{1}{2}$ is true. Thus, the following holds.

$$
\begin{equation*}
\operatorname{Adv}_{\mathrm{FE}, \mathbf{A}}^{\text {ind-cpa }}(\lambda)=\left|\operatorname{Pr}\left[b^{\prime}=b \mid \Pi_{c}(\alpha)\right]-\frac{1}{2}\right|<\varepsilon \tag{2}
\end{equation*}
$$

Therefore, the theorem holds.

Remark 41 In the case of the example in Appendix, the determined differential equations (Appendix (4)) is completely solved and the general solution $\Pi_{c}(t)$ is obtained (Appendix (5)).
Therefore there is the polynomial time algorithm to generate key (PK, SK).

### 4.5 Remarks

In this section, we state other strength, for example, splitting method of $E\left(v_{1}\right)$ and transforming method to rational form for parameter $t_{2}$. And any other issues requiring special attention are stated.

1. In Step5 of KeyGen, we treat the splitting $E\left(v_{1}\right)=\sum_{i=0}^{n} E_{i}$. We first use different parameters $t_{2}, t_{3}$ and make the matrix

$$
\widetilde{E}={ }^{t} C(\tau){ }^{t} \Pi_{c}\left(t_{3}\right) G\left(t_{2}\right) \Pi_{c}\left(t_{3}\right) C(\tau)
$$

because of $E={ }^{t} C{ }^{t} \Pi_{c} G \Pi_{c} C$. Next $\widetilde{E}\left(v_{1}\right)={ }^{t} v_{0} \widetilde{E} v_{0}$ is calculated and is splitted to $\widetilde{E}\left(v_{1}\right)=\sum_{i=0}^{n} \widetilde{E}_{i}$. And last, parameter $t_{3}$ of each component $\widetilde{E}_{i}$ is change to $t_{2}$. In this way, we have the splitting of $E\left(v_{1}\right)=\sum_{i=0}^{n} E_{i}$. Therefore, by different splitting of $\widetilde{E}\left(v_{1}\right)$ we have other splitting of $E\left(v_{1}\right)$. The splitting method is arbitrary.
2. In $\S 2.1$, we use the following transformation because of obtaining rational forms of formations in $G, \Pi_{c}$

$$
t_{2}^{2}:=l^{2}-\left(b^{2} h_{1}+a^{2} h_{2} m^{2}\right) t
$$

However, many other transformations exist, for example,

$$
\begin{gathered}
t_{2}^{4}:=l^{2}-\left(b^{2} h_{1}+a^{2} h_{2} m^{2}\right) t \\
\left(t_{2}+1\right)^{2}:=l^{2}-\left(b^{2} h_{1}+a^{2} h_{2} m^{2}\right) t \\
\left(\frac{t_{2}+1}{t_{2}}\right)^{2}:=l^{2}-\left(b^{2} h_{1}+a^{2} h_{2} m^{2}\right) t
\end{gathered}
$$

The transforming method of the parameter $t$ in the solution $\left(B_{1}^{1}, B_{2}^{1}, B_{2}^{1}, B_{2}^{2}\right)$ of the differential equation $(\star)$ is arbitrary. By using above transformations, all equations in PK and SK come to algebraic (or rational), fortunately. However, such thing will not always happen to us. Further, the differential equations(which give geodesics in Appendix (4)) which we must solve and its solutions are always complex.
3. Next, we state the regularity of the simultaneous linear equation ( $I$ ). In Step4 of $\operatorname{Dec}(\mathbf{S K}, c t)$, for the ciphertext $c t=\left(c t_{1}, \cdots, c t_{n+1}\right)$, each inner product $E X_{1}:=<$ $s x, \bar{c} t_{1}>, \cdots, E X_{n+1}:=<s x, \bar{c} t_{n+1}>$ is expressed as follows:

$$
\begin{gathered}
E X_{1}=c t_{11} f_{0}+\bar{c} t_{12} f_{1} X_{1}+\cdots+\overline{c t}_{1(n+1)} f_{n} X_{n} \\
\vdots \\
E X_{n+1}=c t_{(n+1) 1} f_{0}+\bar{c} t_{(n+1) 2} f_{1} X_{1}+\cdots+\bar{c} t_{(n+1)(n+1)} f_{n} X_{n}
\end{gathered}
$$

Then, the determinant Det of (I)

$$
\text { Det }=\left|\begin{array}{ccc}
f_{1}\left(\bar{c} t_{12}-\bar{c} t_{(n+1) 2}\right) & \cdots & f_{n}\left(\overline{c t} t_{1(n+1)}-\bar{c} t_{(n+1)(n+1)}\right) \\
\vdots & \ddots & \vdots \\
f_{1}\left(\bar{c} t_{n 2}-\bar{c} t_{(n+1) 2}\right) & \cdots & f_{n}\left(\overline{c t}_{n(n+1)}-\bar{c} t_{(n+1)(n+1)}\right)
\end{array}\right|
$$

For example, in the case $n=2$ in p.7,
$D e t=\frac{1}{2}\left(c t_{12} c t_{23}-c t_{12} c t_{33}+c t_{13} c t_{32}-c t_{13} c t_{22}+c t_{22} c t_{33}-c t_{23} c t_{32}\right) \tau^{3}$
is satisfied. If $\operatorname{Det}=0$, then we can change $\beta_{i}$ so that $D e t \neq 0$ is satisfied. Therefore the regularity of $(I)$ is recognized from the ciphertext $c t$ only.
4. The encryption map $P K_{\alpha, \beta_{1}, \cdots, \beta_{(n+1) k}}: \mathbb{Z}_{+}^{2} \rightarrow \mathbb{Q}^{9}$ defined by parameters $\left(\alpha, \beta_{1}, \cdots, \beta_{(n+1) k}\right)$ is one to one if $\operatorname{Det} \neq 0$ of $(I)$ is satisfied.Namely, different plaintexts $v, \bar{v}(\neq v)$ don't have the same ciphertext $c t=P K_{\alpha, \beta_{1}, \cdots, \beta_{(n+1) k}}(v)=P K_{\alpha, \beta_{1}, \cdots, \beta_{(n+1) k}}(\bar{v})$. On the other hand, if $\left(\alpha, \beta_{1}, \cdots, \beta_{(n+1) k}\right) \neq\left(\bar{\alpha}, \overline{\beta_{1}}, \cdots, \overline{\beta_{(n+1) k}}\right), P K_{\alpha, \beta_{1}, \cdots, \beta_{(n+1) k}}(v) \neq$ $P K_{\bar{\alpha}, \overline{\beta_{1}}, \cdots, \overline{\beta_{(n+1) k}}}(v)$ will happen for a plaintext $v$.
5. We state the solution of the energy equation

$$
\left.E X_{1}\right|_{X_{1} \leftarrow \bar{X}_{1}, \cdots, X_{n} \leftarrow \bar{X}_{n}}-\left.E\left(v_{1}\right)\right|_{v_{0}^{1} \leftarrow \bar{X}_{1}, \cdots, v_{0}^{n} \leftarrow \bar{X}_{n}}=0 .
$$

This equation is an algebraic equation of a certain degree in $\tau$. Further the real solution's number is two only. In addition, true solution is rational number. Indeed, in Decryption of the case $n=2$ in p.7, this is an algebraic equation of degree 4 in $\tau$. How to solve this equation? It, however, is no problem because we have known the method of finding rational solutions, for example, Newton-Raphson method for an algebraic equation.
The next problem is particularly important.
6. Does the energy equation

$$
\left.E X_{1}\right|_{X_{1} \leftarrow \bar{X}_{1}, \cdots, X_{n} \leftarrow \bar{X}_{n}}-\left.E\left(v_{1}\right)\right|_{v_{0}^{1} \leftarrow \bar{X}_{1}, \cdots, v_{0}^{n} \leftarrow \bar{X}_{n}}=0
$$

have two rational solutions $\alpha_{1}$ and $\alpha_{2}$ ? Further, do $\alpha_{1}$ and $\alpha_{2}$ yield two integer plaintext $v, \bar{v}$ ? This means that different plaintext $v, \bar{v}$ have the same ciphertext with different parameters $\left(\alpha, \beta_{1}, \cdots, \beta_{(n+1) k}\right) \neq\left(\bar{\alpha}, \bar{\beta}_{1}, \cdots, \overline{\beta_{(n+1) k}}\right)$. This is an open problem.
7. In 2 above, we state the transformations about $t$. This is called "coordinate transformation" in differential geometry, in general. Then, the transformation $t=\phi\left(t_{2}\right)$ must satisfy

$$
\frac{d t}{d t_{2}}=\phi^{\prime}\left(t_{2}\right) \neq 0
$$

If there exist a certain $\tilde{t}$ which satisfies $\phi^{\prime}(\tilde{t})=0$, then we omit such $\tilde{t}$.

## 5 Conclusion

Based on a Finsler space, we formalized Finsler encryption.

1. We must choose a Finsler space with the asymmetry property (See Appendix (2)).
2. We must choose a geodesic on the Finsler space.
3. We must obtain the linear parallel displacement on the geodesic.
4. The strength is based on the three following open problems (i),(ii),(iii):
(i). LPD problem(See Appendix (1)),
(ii). Mapping-decomposition problem: To decompose the matrix $E={ }^{t} C{ }^{t} \Pi_{c} G \Pi_{c} C$ is computationally hard (See $\S 4.1,3$ ),
(iii). SUS problem: To solve underdetermined system of equations is very hard(See §4.2),
and further, it owes of arbitrariness of $C(\tau)$, splitting of $E$ and the method of rationalization of forms.
5. In Example(p.4), Finsler space is defined as a single $\left(a, b, h_{1}, h_{2}\right)$, namely, the form $(\star \star)$ expresses the family of Finsler spaces, its amount is about $10^{4 \lambda}(\lambda$ is a security parameter). The parameter $m$ expresses a geodesic, and $t_{0}, t_{1}$ express the start point and the end point. Therefore the amount of $(\mathbf{P K}, \mathbf{S K})$ is about at least $10^{7 \lambda}$.
6. In our Finsler encryption scheme FE, all calculations are over rational number field $\mathbb{Q}$ with $\lambda$-bit quantization.

Key generation, encryption and decryption were given in detail. For intuitive understanding, an example was presented. Then we analyzed the strength of Finsler encryption. Future direction would be a digital signature scheme on a Finsler space.

## Appendix

## (1) LPD problem and LPD assumption

Computational problem for linear parallel displacement (LPD problem) Suppose that each variable is quantized with $\lambda$-bit, uniformly. Let $(M, F)$ be a Finsler space and $p, q$ be points on $M$. For a geodesic $c$ from $p$ to $q$, the problem is stated as the computational problem to find values of the parameters of linear parallel displacement along $c$ from $T_{p} M$ to $T_{q} M$, where $T_{p} M, T_{q} M$ are tangent spaces at $p, q$ respectively.

## LPD assumption

For a fixed Finsler space with $H_{j}^{i} \neq 0$, there exists no polynomial time algorithm to solve a random instance of LPD problem.
(2) Let $M$ be an $n$-dimensional differentiable real manifold. Let $(M, F)$ be a Finsler space with the metric function $F$ which is $2 n$-variable real-valued function on the tangent bundle TM. F plays very important role of which geodesic, linear parallel displacement and norm are determined. Further, we assume that

$$
H_{j}^{i}(x, y):=\sum_{r} F_{r j}^{i}(x, y) y^{r}+\sum_{r} F_{r j}^{i}(x,-y)\left(-y^{r}\right) \neq 0
$$

where

$$
\begin{gathered}
F_{r j}^{i}:=\frac{1}{2} \sum_{k} g^{i k}\left(\frac{\delta g_{r k}}{\delta x^{j}}+\frac{\delta g_{k j}}{\delta x^{r}}-\frac{\delta g_{j r}}{\delta x^{k}}\right) \quad\left(\left(g^{i j}\right)=\left(g_{i j}\right)^{-1}\right) \\
\frac{\delta}{\delta x^{i}}:=\frac{\partial}{\partial x^{i}}-\sum_{r, j} N_{i}^{r}(x, y) \frac{\partial}{\partial y^{r}}
\end{gathered}
$$

16 T. Nagano et al.

Hereafter the indices $h, i, j, \cdots, p, q, r, \cdots$ of $\sum$ run from 1 to $n(=\operatorname{dim} M)$.
(3)

$$
N_{j}^{i}(x, y)=\sum_{r} \gamma_{r j}^{i}(x, y) y^{r}-\sum_{p, q, r} C_{j r}^{i}(x, y) \gamma_{p q}^{r}(x, y) y^{p} y^{q}
$$

where

$$
\begin{aligned}
& \gamma_{p q}^{i}(x, y)= \sum_{h} \frac{1}{2} g^{h i}\left(\frac{\partial g_{p h}}{\partial x^{q}}+\frac{\partial g_{h q}}{\partial x^{p}}-\frac{\partial g_{p q}}{\partial x^{h}}\right), \\
& C_{j r}^{i}(x, y)=\sum_{h} \frac{1}{2} g^{h i} \frac{\partial g_{j h}}{\partial y^{r}}
\end{aligned}
$$

(4) Geodesic is the curve which is minimizing of the distance between two points locally. Then, a geodesic $c(t)=\left(c^{i}(t)\right)$ satisfies the following equation

$$
\frac{d^{2} c^{i}}{d t^{2}}+\sum_{j, r} F_{j r}^{i}(c, \dot{c}) \dot{c}^{j} \dot{c}^{r}=0 \quad\left(\dot{c}=\left(\dot{c}^{i}\right), \dot{c}^{i}=\frac{d c^{i}}{d t}\right)
$$

where $t$ is an affine parameter.
(5)

$$
\begin{aligned}
& B_{1}^{1}=-\frac{1}{\left(a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2}\left(t+t_{0}\right)\right)-b^{2} h_{1}\left(t+t_{0}\right)\right)^{3 / 2}} \times \\
& \left(a ^ { 2 } \left(h_{2} m^{2}\left(t+t_{0}\right) \sqrt{a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2} t_{0}\right)-b^{2} h_{1} t_{0}}\right.\right. \\
& -b^{2}\left(\sqrt{a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2}\left(t+t_{0}\right)\right)-b^{2} h_{1}\left(t+t_{0}\right)}\right. \\
& \left.\left.+m^{2} \sqrt{a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2} t_{0}\right)-b^{2} h_{1} t_{0}}\right)\right) \\
& \left.+b^{2} h_{1} t_{0} \sqrt{a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2}\left(t+t_{0}\right)\right)-b^{2} h_{1}\left(t+t_{0}\right)}\right) \\
& B_{2}^{1}=\frac{1}{\left(a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2}\left(t+t_{0}\right)\right)-b^{2} h_{1}\left(t+t_{0}\right)\right)^{3 / 2}} \times \\
& \left(a b m \left(b ^ { 2 } \left(\sqrt{a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2}\left(t+t_{0}\right)\right)-b^{2} h_{1}\left(t+t_{0}\right)}\right.\right.\right. \\
& \left.-\sqrt{a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2} t_{0}\right)-b^{2} h_{1} t_{0}}\right) \\
& +h_{2}\left(t \sqrt{a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2} t_{0}\right)-b^{2} h_{1} t_{0}}\right. \\
& +t_{0} \sqrt{a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2} t_{0}\right)-b^{2} h_{1} t_{0}} \\
& \left.\left.\left.-t_{0} \sqrt{a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2}\left(t+t_{0}\right)\right)-b^{2} h_{1}\left(t+t_{0}\right)}\right)\right)\right)
\end{aligned}
$$

$$
\begin{aligned}
& B_{1}^{2}=\frac{1}{\left(a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2}\left(t+t_{0}\right)\right)-b^{2} h_{1}\left(t+t_{0}\right)\right)^{3 / 2}} \times \\
& \left(a b m \left(a ^ { 2 } \left(\sqrt{a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2}\left(t+t_{0}\right)\right)-b^{2} h_{1}\left(t+t_{0}\right)}\right.\right.\right. \\
& \left.-\sqrt{a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2} t_{0}\right)-b^{2} h_{1} t_{0}}\right) \\
& +h_{1}\left(t \sqrt{a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2} t_{0}\right)-b^{2} h_{1} t_{0}}\right. \\
& +t_{0} \sqrt{a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2} t_{0}\right)-b^{2} h_{1} t_{0}} \\
& \left.\left.\left.-t_{0} \sqrt{a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2}\left(t+t_{0}\right)\right)-b^{2} h_{1}\left(t+t_{0}\right)}\right)\right)\right)
\end{aligned}
$$

$$
\begin{aligned}
& B_{2}^{2}=-\frac{1}{\left(a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2}\left(t+t_{0}\right)\right)-b^{2} h_{1}\left(t+t_{0}\right)\right)^{3 / 2}} \times \\
& \left(-a^{2} b^{2}\left(m^{2} \sqrt{a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2}\left(t+t_{0}\right)\right)-b^{2} h_{1}\left(t+t_{0}\right)}\right.\right. \\
& \left.+\sqrt{a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2} t_{0}\right)-b^{2} h_{1} t_{0}}\right) \\
& +b^{2} h_{1}\left(t+t_{0}\right) \sqrt{a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2} t_{0}\right)-b^{2} h_{1} t_{0}} \\
& \left.+a^{2} h_{2} m^{2} t_{0} \sqrt{a^{2}\left(b^{2}\left(m^{2}+1\right)-h_{2} m^{2}\left(t+t_{0}\right)\right)-b^{2} h_{1}\left(t+t_{0}\right)}\right)
\end{aligned}
$$

(6)

$$
\begin{aligned}
& \frac{E_{0}}{f_{0}}=\frac{1}{64 t_{2}^{4}} \times \\
& \left(t_{2}^{6}\left(3 \tau^{2}\left(v_{0}^{1}\right)^{2}-6 \tau\left(v_{0}^{1}\right)^{2}+3\left(v_{0}^{1}\right)^{2}-6 \tau^{2} v_{0}^{1} v_{0}^{2}+6 v_{0}^{1} v_{0}^{2}+3 \tau^{2}\left(v_{0}^{2}\right)^{2}+6 \tau\left(v_{0}^{2}\right)^{2}+3\left(v_{0}^{2}\right)^{2}\right)\right. \\
& +t_{2}^{5}\left(-8 \tau^{2}\left(v_{0}^{1}\right)^{2}-8 \tau\left(v_{0}^{1}\right)^{2}+16\left(v_{0}^{1}\right)^{2}-8 \tau^{2} v_{0}^{1} v_{0}^{2}+48 \tau v_{0}^{1} v_{0}^{2}\right. \\
& \left.\quad+8 v_{0}^{1} v_{0}^{2}+16 \tau^{2}\left(v_{0}^{2}\right)^{2}+8 \tau\left(v_{0}^{2}\right)^{2}-8\left(v_{0}^{2}\right)^{2}\right) \\
& +t_{2}^{4}\left(-2 \tau^{2}\left(v_{0}^{1}\right)^{2}+28 \tau\left(v_{0}^{1}\right)^{2}+10\left(v_{0}^{1}\right)^{2}+28 \tau^{2} v_{0}^{1} v_{0}^{2}+24 \tau v_{0}^{1} v_{0}^{2}\right. \\
& \left.\quad-28 v_{0}^{1} v_{0}^{2}+10 \tau^{2}\left(v_{0}^{2}\right)^{2}-28 \tau\left(v_{0}^{2}\right)^{2}-2\left(v_{0}^{2}\right)^{2}\right) \\
& +t_{2}^{3}\left(16 \tau^{2}\left(v_{0}^{1}\right)^{2}+16 \tau\left(v_{0}^{1}\right)^{2}-32\left(v_{0}^{1}\right)^{2}+16 \tau^{2} v_{0}^{1} v_{0}^{2}-96 \tau v_{0}^{1} v_{0}^{2}\right. \\
& \left.\quad-16 v_{0}^{1} v_{0}^{2}-32 \tau^{2}\left(v_{0}^{2}\right)^{2}-16 \tau\left(v_{0}^{2}\right)^{2}+16\left(v_{0}^{2}\right)^{2}\right) \\
& +t_{2}^{2}\left(44 \tau^{2}\left(v_{0}^{1}\right)^{2}-40 \tau\left(v_{0}^{1}\right)^{2}+68\left(v_{0}^{1}\right)^{2}-40 \tau^{2} v_{0}^{1} v_{0}^{2}+48 \tau v_{0}^{1} v_{0}^{2}\right. \\
& \left.\quad+40 v_{0}^{1} v_{0}^{2}+68 \tau^{2}\left(v_{0}^{2}\right)^{2}+40 \tau\left(v_{0}^{2}\right)^{2}+44\left(v_{0}^{2}\right)^{2}\right) \\
& \left.+24 \tau^{2}\left(v_{0}^{1}\right)^{2}-48 \tau\left(v_{0}^{1}\right)^{2}+24\left(v_{0}^{1}\right)^{2}-48 \tau^{2} v_{0}^{1} v_{0}^{2}+48 v_{0}^{1} v_{0}^{2}+24 \tau^{2}\left(v_{0}^{2}\right)^{2}+48 \tau\left(v_{0}^{2}\right)^{2}+24\left(v_{0}^{2}\right)^{2}\right)
\end{aligned}
$$

18 T. Nagano et al.

$$
\begin{aligned}
& V_{3}^{1}=\frac{-1}{64 \tau^{2} t_{2}^{4} v_{0}^{1} v_{0}^{2}} \times \\
& \left(t _ { 2 } ^ { 6 } \left(2 \tau^{3}\left(v_{0}^{1}\right)^{3}-6 \tau^{2}\left(v_{0}^{1}\right)^{3}+6 \tau\left(v_{0}^{1}\right)^{3}-2\left(v_{0}^{1}\right)^{3}+3 \tau^{3}\left(v_{0}^{1}\right)^{2} v_{0}^{2}\right.\right. \\
& -3 \tau^{2}\left(v_{0}^{1}\right)^{2} v_{0}^{2}-3 \tau\left(v_{0}^{1}\right)^{2} v_{0}^{2}+3\left(v_{0}^{1}\right)^{2} v_{0}^{2}-12 \tau^{3} v_{0}^{1}\left(v_{0}^{2}\right)^{2} \\
& \left.-12 \tau^{2} v_{0}^{1}\left(v_{0}^{2}\right)^{2}+12 \tau v_{0}^{1}\left(v_{0}^{2}\right)^{2}+12 v_{0}^{1}\left(v_{0}^{2}\right)^{2}+7 \tau^{3}\left(v_{0}^{2}\right)^{3}+21 \tau^{2}\left(v_{0}^{2}\right)^{3}+21 \tau\left(v_{0}^{2}\right)^{3}+7\left(v_{0}^{2}\right)^{3}\right) \\
& +t_{2}^{5}\left(-4 \tau^{2}\left(v_{0}^{1}\right)^{3}+8 \tau\left(v_{0}^{1}\right)^{3}-4\left(v_{0}^{1}\right)^{3}-12 \tau^{3}\left(v_{0}^{1}\right)^{2} v_{0}^{2}-12 \tau^{2}\left(v_{0}^{1}\right)^{2} v_{0}^{2}\right. \\
& +4 \tau\left(v_{0}^{1}\right)^{2} v_{0}^{2}+20\left(v_{0}^{1}\right)^{2} v_{0}^{2}-12 \tau^{3} v_{0}^{1}\left(v_{0}^{2}\right)^{2}+48 \tau^{2} v_{0}^{1}\left(v_{0}^{2}\right)^{2} \\
& \left.+76 \tau v_{0}^{1}\left(v_{0}^{2}\right)^{2}+16 v_{0}^{1}\left(v_{0}^{2}\right)^{2}+24 \tau^{3}\left(v_{0}^{2}\right)^{3}+40 \tau^{2}\left(v_{0}^{2}\right)^{3}+8 \tau\left(v_{0}^{2}\right)^{3}-8\left(v_{0}^{2}\right)^{3}\right) \\
& +t_{2}^{4}\left(-8 \tau^{3}\left(v_{0}^{1}\right)^{3}+20 \tau^{2}\left(v_{0}^{1}\right)^{3}-24 \tau\left(v_{0}^{1}\right)^{3}+12\left(v_{0}^{1}\right)^{3}-30 \tau^{3}\left(v_{0}^{1}\right)^{2} v_{0}^{2}\right. \\
& +6 \tau^{2}\left(v_{0}^{1}\right)^{2} v_{0}^{2}+26 \tau\left(v_{0}^{1}\right)^{2} v_{0}^{2}-26\left(v_{0}^{1}\right)^{2} v_{0}^{2}+56 \tau^{3} v_{0}^{1}\left(v_{0}^{2}\right)^{2} \\
& +76 \tau^{2} v_{0}^{1}\left(v_{0}^{2}\right)^{2}-56 \tau v_{0}^{1}\left(v_{0}^{2}\right)^{2}-60 v_{0}^{1}\left(v_{0}^{2}\right)^{2}-38 \tau^{3}\left(v_{0}^{2}\right)^{3} \\
& \left.-106 \tau^{2}\left(v_{0}^{2}\right)^{3}-110 \tau\left(v_{0}^{2}\right)^{3}-42\left(v_{0}^{2}\right)^{3}\right) \\
& +t_{2}^{3}\left(8 \tau^{2}\left(v_{0}^{1}\right)^{3}-16 \tau\left(v_{0}^{1}\right)^{3}+8\left(v_{0}^{1}\right)^{3}+24 \tau^{3}\left(v_{0}^{1}\right)^{2} v_{0}^{2}+24 \tau^{2}\left(v_{0}^{1}\right)^{2} v_{0}^{2}\right. \\
& -8 \tau\left(v_{0}^{1}\right)^{2} v_{0}^{2}-40\left(v_{0}^{1}\right)^{2} v_{0}^{2}+24 \tau^{3} v_{0}^{1}\left(v_{0}^{2}\right)^{2}-96 \tau^{2} v_{0}^{1}\left(v_{0}^{2}\right)^{2} \\
& \left.-152 \tau v_{0}^{1}\left(v_{0}^{2}\right)^{2}-32 v_{0}^{1}\left(v_{0}^{2}\right)^{2}-48 \tau^{3}\left(v_{0}^{2}\right)^{3}-80 \tau^{2}\left(v_{0}^{2}\right)^{3}-16 \tau\left(v_{0}^{2}\right)^{3}+16\left(v_{0}^{2}\right)^{3}\right) \\
& +t_{2}^{2}\left(8 \tau^{2}\left(v_{0}^{1}\right)^{3}+8 \tau\left(v_{0}^{1}\right)^{3}-16\left(v_{0}^{1}\right)^{3}+52 \tau^{3}\left(v_{0}^{1}\right)^{2} v_{0}^{2}+44 \tau^{2}\left(v_{0}^{1}\right)^{2} v_{0}^{2}\right. \\
& +36 \tau\left(v_{0}^{1}\right)^{2} v_{0}^{2}+108\left(v_{0}^{1}\right)^{2} v_{0}^{2}-8 \tau^{3} v_{0}^{1}\left(v_{0}^{2}\right)^{2}+64 \tau^{2} v_{0}^{1}\left(v_{0}^{2}\right)^{2} \\
& \left.+144 \tau v_{0}^{1}\left(v_{0}^{2}\right)^{2}+24 v_{0}^{1}\left(v_{0}^{2}\right)^{2}+100 \tau^{3}\left(v_{0}^{2}\right)^{3}+124 \tau^{2}\left(v_{0}^{2}\right)^{3}+68 \tau\left(v_{0}^{2}\right)^{3}+44\left(v_{0}^{2}\right)^{3}\right) \\
& +16 \tau^{3}\left(v_{0}^{1}\right)^{3}-48 \tau^{2}\left(v_{0}^{1}\right)^{3}+48 \tau\left(v_{0}^{1}\right)^{3}-16\left(v_{0}^{1}\right)^{3}+24 \tau^{3}\left(v_{0}^{1}\right)^{2} v_{0}^{2} \\
& -24 \tau^{2}\left(v_{0}^{1}\right)^{2} v_{0}^{2}-24 \tau\left(v_{0}^{1}\right)^{2} v_{0}^{2}+24\left(v_{0}^{1}\right)^{2} v_{0}^{2}-96 \tau^{3} v_{0}^{1}\left(v_{0}^{2}\right)^{2} \\
& -96 \tau^{2} v_{0}^{1}\left(v_{0}^{2}\right)^{2}+96 \tau v_{0}^{1}\left(v_{0}^{2}\right)^{2}+96 v_{0}^{1}\left(v_{0}^{2}\right)^{2}+56 \tau^{3}\left(v_{0}^{2}\right)^{3}+168 \tau^{2}\left(v_{0}^{2}\right)^{3} \\
& \left.+168 \tau\left(v_{0}^{2}\right)^{3}+56\left(v_{0}^{2}\right)^{3}\right),
\end{aligned}
$$

$$
\begin{aligned}
& V_{3}^{2}=\frac{1}{64 \tau^{2} t_{2}^{4} v_{0}^{1} v_{0}^{2}} \times \\
& \left(t _ { 2 } ^ { 6 } \left(2 \tau^{3}\left(v_{0}^{1}\right)^{3}-2 \tau^{2}\left(v_{0}^{1}\right)^{3}-2 \tau\left(v_{0}^{1}\right)^{3}+2\left(v_{0}^{1}\right)^{3}+3 \tau^{3}\left(v_{0}^{1}\right)^{2} v_{0}^{2}\right.\right. \\
& -25 \tau^{2}\left(v_{0}^{1}\right)^{2} v_{0}^{2}+25 \tau\left(v_{0}^{1}\right)^{2} v_{0}^{2}-3\left(v_{0}^{1}\right)^{2} v_{0}^{2}-12 \tau^{3} v_{0}^{1}\left(v_{0}^{2}\right)^{2} \\
& \left.+20 \tau^{2} v_{0}^{1}\left(v_{0}^{2}\right)^{2}+20 \tau v_{0}^{1}\left(v_{0}^{2}\right)^{2}-12 v_{0}^{1}\left(v_{0}^{2}\right)^{2}+7 \tau^{3}\left(v_{0}^{2}\right)^{3}+7 \tau^{2}\left(v_{0}^{2}\right)^{3}-7 \tau\left(v_{0}^{2}\right)^{3}-7\left(v_{0}^{2}\right)^{3}\right) \\
& +t_{2}^{5}\left(-4 \tau^{2}\left(v_{0}^{1}\right)^{3}+4\left(v_{0}^{1}\right)^{3}-12 \tau^{3}\left(v_{0}^{1}\right)^{2} v_{0}^{2}-4 \tau^{2}\left(v_{0}^{1}\right)^{2} v_{0}^{2}\right. \\
& +52 \tau\left(v_{0}^{1}\right)^{2} v_{0}^{2}-20\left(v_{0}^{1}\right)^{2} v_{0}^{2}-12 \tau^{3} v_{0}^{1}\left(v_{0}^{2}\right)^{2}+88 \tau^{2} v_{0}^{1}\left(v_{0}^{2}\right)^{2} \\
& \left.-44 \tau v_{0}^{1}\left(v_{0}^{2}\right)^{2}-16 v_{0}^{1}\left(v_{0}^{2}\right)^{2}+24 \tau^{3}\left(v_{0}^{2}\right)^{3}-8 \tau^{2}\left(v_{0}^{2}\right)^{3}-24 \tau\left(v_{0}^{2}\right)^{3}+8\left(v_{0}^{2}\right)^{3}\right) \\
& +t_{2}^{4}\left(-8 \tau^{3}\left(v_{0}^{1}\right)^{3}+4 \tau^{2}\left(v_{0}^{1}\right)^{3}-12\left(v_{0}^{1}\right)^{3}-30 \tau^{3}\left(v_{0}^{1}\right)^{2} v_{0}^{2}\right. \\
& +114 \tau^{2}\left(v_{0}^{1}\right)^{2} v_{0}^{2}-126 \tau\left(v_{0}^{1}\right)^{2} v_{0}^{2}+26\left(v_{0}^{1}\right)^{2} v_{0}^{2}+56 \tau^{3} v_{0}^{1}\left(v_{0}^{2}\right)^{2} \\
& -84 \tau^{2} v_{0}^{1}\left(v_{0}^{2}\right)^{2}-96 \tau v_{0}^{1}\left(v_{0}^{2}\right)^{2}+60 v_{0}^{1}\left(v_{0}^{2}\right)^{2}-38 \tau^{3}\left(v_{0}^{2}\right)^{3} \\
& \left.-30 \tau^{2}\left(v_{0}^{2}\right)^{3}+26 \tau\left(v_{0}^{2}\right)^{3}+42\left(v_{0}^{2}\right)^{3}\right) \\
& +t_{2}^{3}\left(8 \tau^{2}\left(v_{0}^{1}\right)^{3}-8\left(v_{0}^{1}\right)^{3}+24 \tau^{3}\left(v_{0}^{1}\right)^{2} v_{0}^{2}+8 \tau^{2}\left(v_{0}^{1}\right)^{2} v_{0}^{2}\right. \\
& -104 \tau\left(v_{0}^{1}\right)^{2} v_{0}^{2}+40\left(v_{0}^{1}\right)^{2} v_{0}^{2}+24 \tau^{3} v_{0}^{1}\left(v_{0}^{2}\right)^{2}-176 \tau^{2} v_{0}^{1}\left(v_{0}^{2}\right)^{2} \\
& \left.+88 \tau v_{0}^{1}\left(v_{0}^{2}\right)^{2}+32 v_{0}^{1}\left(v_{0}^{2}\right)^{2}-48 \tau^{3}\left(v_{0}^{2}\right)^{3}+16 \tau^{2}\left(v_{0}^{2}\right)^{3}+48 \tau\left(v_{0}^{2}\right)^{3}-16\left(v_{0}^{2}\right)^{3}\right) \\
& +t_{2}^{2}\left(8 \tau^{2}\left(v_{0}^{1}\right)^{3}+24 \tau\left(v_{0}^{1}\right)^{3}+16\left(v_{0}^{1}\right)^{3}+52 \tau^{3}\left(v_{0}^{1}\right)^{2} v_{0}^{2}-28 \tau^{2}\left(v_{0}^{1}\right)^{2} v_{0}^{2}\right. \\
& +148 \tau\left(v_{0}^{1}\right)^{2} v_{0}^{2}-108\left(v_{0}^{1}\right)^{2} v_{0}^{2}-8 \tau^{3} v_{0}^{1}\left(v_{0}^{2}\right)^{2}+144 \tau^{2} v_{0}^{1}\left(v_{0}^{2}\right)^{2} \\
& \left.-96 \tau v_{0}^{1}\left(v_{0}^{2}\right)^{2}-24 v_{0}^{1}\left(v_{0}^{2}\right)^{2}+100 \tau^{3}\left(v_{0}^{2}\right)^{3}-76 \tau^{2}\left(v_{0}^{2}\right)^{3}+20 \tau\left(v_{0}^{2}\right)^{3}-44\left(v_{0}^{2}\right)^{3}\right) \\
& +16 \tau^{3}\left(v_{0}^{1}\right)^{3}-16 \tau^{2}\left(v_{0}^{1}\right)^{3}-16 \tau\left(v_{0}^{1}\right)^{3}+16\left(v_{0}^{1}\right)^{3}+24 \tau^{3}\left(v_{0}^{1}\right)^{2} v_{0}^{2} \\
& -200 \tau^{2}\left(v_{0}^{1}\right)^{2} v_{0}^{2}+200 \tau\left(v_{0}^{1}\right)^{2} v_{0}^{2}-24\left(v_{0}^{1}\right)^{2} v_{0}^{2}-96 \tau^{3} v_{0}^{1}\left(v_{0}^{2}\right)^{2} \\
& +160 \tau^{2} v_{0}^{1}\left(v_{0}^{2}\right)^{2}+160 \tau v_{0}^{1}\left(v_{0}^{2}\right)^{2}-96 v_{0}^{1}\left(v_{0}^{2}\right)^{2}+56 \tau^{3}\left(v_{0}^{2}\right)^{3} \\
& \left.+56 \tau^{2}\left(v_{0}^{2}\right)^{3}-56 \tau\left(v_{0}^{2}\right)^{3}-56\left(v_{0}^{2}\right)^{3}\right) \text {. }
\end{aligned}
$$

## References

1. T.Aikou and L.Kozma: Global aspects of Finsler geometry. In Handbook of global analysis, pages 1-39, 1211. Elsevier Sci. B. V., Amsterdam (2008).
2. D.Bao, S.-S. Chern and Z.Shen: An Introduction to Riemann-Finsler geometry, 200, Graduate Texts in Math. Springer-Verlag, New York, (2000).
3. S.-S. Chern and Z.Shen: Riemann-Finsler geometry, volume 6 of Nankai Tracts in Mathematics. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ (2005).
4. M. Crampin: Randers spaces with reversible geodesics, Publ. Math.Debrecen, 67(3-4):401-409 (2005).
5. M. Matsumoto: Foundations of Finsler geometry and special Finsler spaces. Kaiseisha Press, Shigaken, 1986.
6. M. Matsumoto: Finsler geometry in the 20th-century. In Handbook of Finsler geometry, Vol. 1, 2, pages 557-966. Kluwer Acad. Publ., Dordrecht, 2003.
7. T. Nagano, N. Innami, Y. Itokawa and K. Shiohama: Notes on reversibility and branching of geodesics in Finsler spaces, Iasi Ploytechic Inst. Bull.-Mathematics. Theoretical Mechanics. Physics Section, pp.9-28, 2019.
8. T.Nagano, H.Anada: Public-Key Encryption Scheme Using Non-symmetry of Finsler Spaces(in Japanese), Proceedings of Computer Security Symposium 2019, Information Processing Society of Japan, pp.415-421,(2019)
9. T. Nagano, H. Anada: One-wayness of Public-Key Encryption Scheme Using Nonsymmetry of Finsler Spaces(in Japanese)(Original title: Indistinguishability of Public-Key Encryption Scheme Using Non-symmetry of Finsler Spaces), Proceedings of 2020 Symposium on Cryptography and Information Security(SCIS 2020), The Institute of Electronics,Information and Communication Engineers, 3A3-1(17), (2020).
10. T.Nagano, H.Anada: Approach to Cryptography from Differential Geometry with Example, Innovative Security Solutions for Information Technology and Communications (SecITC) 2020, Springer, LNCS12596, pp.110-129 (2021).
11. T.Nagano, H.Anada: Mathematical Structure of Finsler Encryption, IPSJ SIG Technical Report, Vol.2021-CSEC-95 No.6(Vol.2021-SPT-45 No.6, Vol.2021-EIP94 No.6), (2021).
12. T.Nagano, H.Anada: Mathematical Structure of Finsler Encryption and Signature, Proceedings of 2022 Symposium on Cryptography and Information Security Osaka(SCIS 2022), Japan \& Online, Jan. 18 - 21, 2022, The Institute of Electronics,Information and Communication Engineers, 2A3-2(1-7), (2022).
13. T.Nagano, H.Anada: Finsler Encryption, 2022 Workshop on interaction between Cryptography, Information Security and Mathematics(CRISMATH 2022), Joint Research Center for Advanced and Fundamental Mathematics-for-Industry, Kyushu Univ. Dec. 20-21, 2022
14. T. Nagano: Notes on the notion of the parallel displacement in Finsler geometry. Tensor (N.S.), 70(3):302-310 (2008).
15. T. Nagano: On the parallel displacement and parallel vector fields in Finsler geometry, Acta Math. Acad. Paedagog. Nyhazi., 26(2):349-358 (2010).
16. T. Nagano: A note on linear parallel displacements in Finsler geometry, Journal of the Faculty of Global Communication, University of Nagasaki, 12:195-205 (2011).
17. T. Nagano: On the quantities W, L, K derived from linear parallel displacements in Finsler geometry, Journal of the Faculty of Global Communication, University of Nagasaki, 14:123-132 (2013).
18. T. El Gamal: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In proc. CRYPTO 84 on Advances in Cryptology, pp.10-18, 1985.
19. J. Katz and Y. Lindell: Introduction to Modern Cryptography, Second Edition, CRC Press, Florida (2014).
20. O. Goldreich: The Foundations of Cryptography - Volume 1, Basic Techniques, Cambridge University Press, Cambridge (2001).
21. O. Goldreich: The Foundations of Cryptography - Volume 2, Basic Applications, Cambridge University Press, Cambridge (2004).

# Experiments and Resource Analysis of Shor's Factorization Using a Quantum Simulator 

Junpei Yamaguchi ${ }^{1}$, Masafumi Yamazaki ${ }^{1}$, Akihiro Tabuchi ${ }^{1}$, Takumi Honda ${ }^{1}$, Tetsuya Izu $^{1}$, and Noboru Kunihiro ${ }^{2}$<br>${ }^{1}$ Fujitsu Limited, Japan<br>\{j-yamaguchi,izu\}@fujitsu.com<br>${ }^{2}$ University of Tsukuba, Japan


#### Abstract

Shor's algorithm on actual quantum computers has succeeded only in factoring small composite numbers such as 15 and 21, and simplified quantum circuits to factor the specific integers are used in these experiments. In this paper, we factor 96 RSA-type composite numbers up to 9 -bit using a quantum computer simulator. The largest composite number $N=511$ was factored in approximately 2 hours on the simulator. In our experiments, we implement Shor's algorithm with basic circuit construction, which does not require complex tricks to reduce the number of qubits, and we give some improvements to reduce the number of gates, including MIX-ADD method. This is a flexible method for selecting the optimal ADD circuit which minimizes the number of gates from the existing ADD circuits for each of the many ADD circuits required in Shor's algorithm. Based on our experiments, we estimate the resources required to factor 2048-bit integers. We estimate that the Shor's basic circuit requires $2.19 \times 10^{12}$ gates and $1.76 \times 10^{12}$ depth when 10241 qubits are available, and $2.37 \times 10^{14}$ gates and $2.00 \times 10^{14}$ depth when 8194 qubits are available.


Keywords: Shor's algorithm • integer factorization • quantum computer - quantum computer simulator.

## 1 Introduction

The security of RSA, one of the standardized public key cryptosystems, is based on the difficulty of the integer factorization problem of large composite numbers. The current factorization record by a classical computer is the factorization of an 829-bit integer [6], so that RSA with larger than 2048-bit integer is considered to be secure for the time being. On the other hand, it is known that the integer factorization problem can be solved in polynomial time by Shor's algorithm by using an ideal quantum computer [18]. Some factorization experiments on quantum computers by using Shor's algorithm have been reported only for $N=15$ and $21[1,13,14,15,16,17]$ because of the difficulty of realizing ideal quantum computers - quantum computers free from the limitation of the number of
quantum bits (qubits) and the noise on the qubits ${ }^{3}$. To make matters worse, these experiments used the simplified Shor's circuits in which qubits and gates are reduced as much as possible by using the properties of the integers to be factored and their factors to be found. Since such experiments do not lead to accurate quantum resource estimation, the implementation of Shor's algorithm which can factor general composite numbers and its resource evaluation based on factoring experiments are required.

Various quantum circuits of Shor's algorithm for general composite numbers have been proposed. Kunihiro summarized basic circuits [12], which use $2 n$ controlling qubits as a 1 st qubit sequence for an $n$-bit composite number. On the other hand, advanced circuits have also been proposed [4,8]. These circuits use a technique to reduce the qubits of the 1 st sequence from $2 n$ to 1 , which requires a complex quantum operation, repeatedly performing observations and quantum gate operations depending on the observation results.

Despite some efforts to estimate circuit resources for factoring 2048-bit integers with noisy qubits $[8,9]$, it is too difficult to give exact estimates for factoring such large integers.

There are two major problems to break the situation. The first problem is the lack of computational resources, specifically, the number of qubits available on quantum computers. Although IBM has recently developed a 433 -qubit processor [10], because of the effects of the noise, it is still difficult to process Shor's algorithm even on such computers. The second problem to be solved is the lack of experimental results for Shor's algorithm. To estimate the circuit resources for factoring 2048-bit integers, more experimental results on the same computing environments are needed.

## Contribution of This Paper

This paper has three contributions. First, we implemented the basic circuits of Shor's algorithm applicable to general composite numbers, and succeeded in factoring 96 RSA-type composite numbers up to 9 -bit on a quantum computer simulator running on a supercomputer. The largest composite number $N=511$ was factorized in 2 hours on the simulator. We used the simulator mpiQulacs developed by Fujitsu [11], a State Vector (SV) type simulator that records all qubit states in memory with no noise and allows to simulate an ideal quantum computation [11]. This paper focuses on the basic circuits because the current large scale quantum simulator cannot handle the advanced circuit due to its complexity. Our implementations are based on the well-known techniques [4,12], but we provide some bug-fixes, improvements (including the second contribution) and comparisons of required resource.

Second, we propose a flexible ADD method, MIX-ADD, to reduce the elementary gates and the depth of the basic circuit. The dominant circuit in Shor

[^21]is Mod-EXP which computes a modular exponentiation $f_{a, N}(x)=a^{x} \bmod N$. Mod-EXP can be constructed from ADD circuits, and there are three well-known ADD circuits: R-ADD, GT-ADD, and Q-ADD [12]. The basic circuit requires $5 n+1$ qubits for R-ADD, and $4 n+2$ qubits for GT/Q/MIX-ADD for $n$-bit composite numbers to be factored. MIX-ADD reduces the gates and the depth by selecting the optimal ADD circuit which minimizes the number of gates from R/GT/Q-ADD for each ADD circuit called multiple times in Mod-EXP. Our analysis shows that R/MIX/Q/GT-ADD require more gates in this order for larger $n$. MIX-ADD can factor larger composite numbers more efficiently in an environment where the number of available qubits is limited like the present.

Finally, we gave estimations of the number of gates and the depth for the Shor's basic circuits. We generated some quantum circuits for $n=8, \ldots, 24$, and evaluated the resources of the circuit. Based on these data, we estimated the circuit resources required to factor 2048 -bit integers. In our estimation, the basic circuit requires 10241 qubits, $2.19 \times 10^{12}$ gates and $1.76 \times 10^{12}$ depth for R-ADD, and 8194 qubits, $2.37 \times 10^{14}$ gates and $2.00 \times 10^{14}$ depth for MIX-ADD.

The rest of the paper is organized as follows: Section 2 describes the construction of Shor's quantum circuit, in particular the modular exponentiation circuit Mod-EXP using ADD circuits. Then, in Section 3, concrete constructions of Mod-EXP from R-ADD, GT-ADD, Q-ADD and MIX-ADD are explained. Section 4 summarizes factoring experiments by Shor's quantum circuit using the quantum computer simulator including the estimation for 2048-bit integers.

## 2 Quantum Circuit of Shor's algorithm

This section describes quantum circuits of Shor's algorithm for general composite numbers based on known techniques [4,12]. In this paper, we consider the following quantum gates as the elementary gates for evaluating the number of gates and the depth: 1-qubit gates including the Hadamard gate, NOT gate, the rotation gate and the phase-shift gate, Controlled NOT (C-NOT) gate, and Toffoli ( $\mathrm{C}^{2}$-NOT) gate.

### 2.1 Shor's algorithm and Factorization

Suppose we want to factor an $n$-bit composite number $N$. For an integer $a$ coprime to $N$, the order of $a$ with regard to $N$ is defined as the smallest positive integer $r$ such that $a^{r} \equiv 1 \bmod N$. In 1994, Shor proposed a quantum algorithm to compute the order $r$ of $a$ with regard to $N$ in polynomial time [18]. The integer $N$ can be factored by using Shor's algorithm in the following way:
i) Choose an integer $a$ from $\{2, \ldots, N-1\}$. If $\operatorname{gcd}(a, N) \neq 1$ then output $\operatorname{gcd}(a, N)$ and terminate (since a factor of $N$ larger than 1 is found).
ii) Compute the order $r$ from $a$ and $N$ by quantum order finding algorithm.
iii) If $r$ is even, $a^{r / 2}+1 \not \equiv 0 \bmod N$ and $\operatorname{gcd}\left(a^{r / 2} \pm 1, N\right) \neq 1$, output $\operatorname{gcd}\left(a^{r / 2} \pm\right.$ $1, N)$ and terminates. Otherwise, return step i).

Note that step i) and iii) can be proceeded by classical computers. On the other hand, step ii) can be computed by the quantum order finding algorithm on a quantum computer in the following way:

1. Generate an initial state $\left|\phi_{0}\right\rangle=|0 \ldots 0\rangle|0 \ldots 01\rangle$, where the 1 st qubit sequence has $m$ qubits, while the 2nd qubit sequence has $n$ qubits.
2. Apply the Hadamard operation $H_{m}$ to the 1st sequence:

$$
\left|\phi_{1}\right\rangle=H_{m}\left(\left|\phi_{0}\right\rangle\right)=\frac{1}{\sqrt{2^{m}}} \sum_{x=0}^{2^{m}-1}|x\rangle|0 \ldots 01\rangle .
$$

3. Apply the operation $U_{f_{a, N}}$ which corresponds to a modular exponentiation $f_{a, N}(x)=a^{x} \bmod N$, to the 2 nd sequence:

$$
\left|\phi_{2}\right\rangle=U_{f_{a, N}}\left(\left|\phi_{1}\right\rangle\right)=\frac{1}{\sqrt{2^{m}}} \sum_{x=0}^{2^{m}-1}|x\rangle\left|f_{a, N}(x)\right\rangle .
$$

4. Apply the Inverse Quantum Fourier Transform to the 1st sequence.
5. Observe the 1st sequence, an approximation of a multiple of $2^{m} / r$ is obtained.
6. Repeat Step 1-5 until $r$ can be estimated.

The parameter $m$ is determined from the approximation precision in Step 5, $m=2 n$ is used usually and in this paper.

### 2.2 Construction of Mod-EXP from ADD

Above steps except Step 3 can be easily realized by elementary gates. On the other hand, Step 3 requires complex circuits called Mod-EXP [12]. This subsection describes how to realize Mod-EXP from elementary gates. In fact, ModEXP can be constructed from ADD circuits, by transforming Mod-EXP to the following circuits step-by-step:
$-\operatorname{Mod}-\operatorname{EXP}(a):|x\rangle|1\rangle \rightarrow|x\rangle\left|a^{x} \bmod N\right\rangle$
$-\operatorname{Mod-MUL}(d):|y\rangle \rightarrow|d y \bmod N\rangle$
$-\operatorname{Mod}-\operatorname{PS}(d):|y\rangle|t\rangle \rightarrow|y\rangle|t+d y \bmod N\rangle$
$-\operatorname{Mod}-\operatorname{ADD}(d):|y\rangle \rightarrow|y+d \bmod N\rangle$
$-\operatorname{ADD}(d):|y\rangle \rightarrow|y+d\rangle$

Construction of Mod-EXP from Mod-MUL For an exponent $x$ represented in binary, namely, $x=\sum_{i=0}^{m-1} 2^{i} x_{i}$, a modular exponentiation $\operatorname{Mod}-\operatorname{EXP}(a)$ is computed by a repetition of multiplying $d_{i}=a^{2^{i}} \bmod N$ and taking modulus by $N$, since $a^{x} \bmod N=a^{\sum_{i=0}^{m-1} 2^{i} x_{i}} \bmod N=\prod_{i=0}^{m-1} a^{2^{i} x_{i}} \bmod N$. In other words, $\operatorname{Mod}-\operatorname{EXP}(a)$ can be computed by a repetition of the modular multiplication $\operatorname{Mod}-\operatorname{MUL}\left(d_{i}\right)$ controlled by $\left|x_{i}\right\rangle$, so that $\operatorname{Mod-EXP}(a)$ requires $m$ controlled-Mod-MULs, which is denoted by $C\left(x_{i}\right)$-Mod-MUL.

Construction of Mod-MUL from Mod-PS The modular multiplication $\operatorname{Mod}-\operatorname{MUL}(d)$ for an integer $0 \leq d<N$ and an $n$-bit integer $y$ can be computed by using modular product-sums Mod-PSs in the following way:

$$
\left.\begin{array}{rl}
|y\rangle|\underbrace{0 \ldots 0}_{n}\rangle & \xrightarrow{\operatorname{Mod}-\mathrm{PS}(d)}|y\rangle|0+d y \bmod N\rangle
\end{array} \xrightarrow{\operatorname{SWAP}}|d y \bmod N\rangle|y\rangle \begin{array}{rl}
\operatorname{Mod}-\mathrm{PS}\left(-d^{-1}\right)
\end{array} d y \bmod N\right\rangle\left|y+\left(-d^{-1}\right)(d y \bmod N) \bmod N\right\rangle .
$$

Since $d_{i}=a^{2^{i}} \bmod N$ and $\operatorname{gcd}(a, N)=1$, there always exists the inverse $d^{-1} \bmod$ $N$. Thus, Mod-MUL can be computed by two Mod-PSs and one $n$-qubit SWAP with $n$ auxiliary qubits $\left|R_{2}\right\rangle=|0 \ldots 0\rangle$. Especially, $C\left(x_{i}\right)$-Mod-MUL requires two $C\left(x_{i}\right)$-Mod-PSs and one $n$-qubit C-SWAP. Moreover, an $n$-qubit C-SWAP can be realized by $n 1$-qubit C-SWAPs, and one 1-qubit C-SWAP can be realized by two C-NOTs and one Toffoli gate.

Construction of Mod-PS from Mod-ADD When the 2nd sequence is represented as $|y\rangle=\left|y_{n-1} \ldots y_{0}\right\rangle$, for an integer $0 \leq d<N$, we have $d y=\sum_{j=0}^{n-1} d 2^{j} y_{j}$. Thus, a modular product-sum $\operatorname{Mod}-\mathrm{PS}(d)$ on a bit sequence $\left|R_{2}\right\rangle$ can be computed by a repetition $R_{2} \leftarrow R_{2}+d 2^{j} \bmod N$ if $y_{j}=1$ for $j=0,1, \ldots, n-1$, which is equivalent to $C\left(y_{j}\right)-\operatorname{Mod}-\mathrm{ADD}\left(d 2^{j} \bmod N\right)$. That is, Mod-PS can be realized by $n 1$-controlled Mod-ADDs, and $C\left(x_{i}\right)$-Mod-PS can be realized by $n$ 2-controlled Mod-ADDs, namely, $C\left(x_{i}, y_{j}\right)$-Mod-ADDs.

Construction of Mod-ADD from ADD There are two constructions, Type 1 and Type 2 for realizing $C\left(x_{i}, y_{j}\right)$-Mod-ADD [12]. From the efficiency point of view, Type 2 is optimal for R-ADD and Q-ADD, while Type 1 for GT-ADD. Due to space limitation, we omit describing the details.

### 2.3 Construction of ADD

This subsection describes how to construct ADD circuits from the elementary gates in three ways: R-ADD, GT-ADD, and Q-ADD. Here, we consider the circuit to add an $n$-bit integer $p=p_{n-1} \ldots p_{0}$ to an $n$-qubit register $\left|R_{2}\right\rangle=$ $\left|R_{2, n-1} \ldots R_{2,0}\right\rangle$. Considering the carry-over, the result is represented by $\left|R_{1} R_{2}\right\rangle$ with 1-qubit register $\left|R_{1}\right\rangle$. All ADD circuits use another 1-qubit register $\left|R_{3}\right\rangle$, and R-ADD uses further $(n-1)$-qubit sequence $|c\rangle$. In total, GT-ADD and QADD require $m+n+1+n+1=m+2 n+2=4 n+2$ qubits, while R-ADD requires $m+2 n+2+(n-1)=m+3 n+1=5 n+1$ qubits. On the other hand, the number of elementary gates is estimated by $270 n^{3}$ for R-ADD, $16 / 3 n^{5}$ for GT-ADD, and $97 n^{4}$ for $\mathrm{Q}-\mathrm{ADD}$ [12].

(a) $p_{k}=0$

Fig. 1: CARRY Circuit

(a) $p_{k}=0$

(b) $p_{k}=1$

Fig. 2: SUM Circuit

Construction of R-ADD R-ADD is a ripple carry adder [5,20], which computes $R_{2}+p$ by using the following addition table:

$$
\begin{array}{ccccc}
c_{n-1} & c_{n-2} & \ldots & c_{1} \\
& & \\
R_{2, n-1} & R_{2, n-2} & \ldots & R_{2,1} & R_{2,0} \\
+) & p_{n-1} & p_{n-2} & \ldots & p_{1}
\end{array} p_{0} .
$$

Here, $c=c_{n-1} \ldots c_{1}$ is an auxiliary $(n-1)$-bit register with initial value 0 , which is used for storing carry-overs. R-ADD consists of three circuits, CARRY (for computing carry bits), SUM (for computing additions), and CARRY ${ }^{-1}$ (for resetting carry bits). As in Figure 2 of Vedral, Barenco and Ekert's paper [20], R-ADD firstly computes all carry-overs by using CARRY circuit described in Figure 1 for $k=0,1, \ldots, n-1$ (set $c_{n}=R_{1}$ when $k=n-1$ ). When $p_{n-1}=1$, apply the NOT gate to $R_{2, n-1}$. Finally, for $k=n-1, \ldots, 0$, update $R_{2, k}$ by using SUM circuit described in Figure 2 and reset $c_{k}$ by using CARRY ${ }^{-1}$ circuit, which is reverse circuit of CARRY. When $k=0$, CARRY $^{-1}$ is omitted. Thus, R-ADD can be constructed from Toffoli gates, C-NOT gates, and NOT gates.

Type 2 Mod-ADD requires 1-controlled R-ADD and 2-controlled R-ADD, which require not only C-NOT gate and Toffoli gate, but 3 -controlled NOT and 4 -controlled NOT gates. Barenco et al. showed two conversions from a $C^{k}$-NOT gate to Toffoli gates [3]. The first conversion converts a $C^{k}$-NOT gate to $2 k-3$ Toffoli gates by using $k-2$ clean auxiliary qubits (qubits with their state known to be $|0\rangle$ ). The second converts a $C^{k}$-NOT gate to $4 k-8$ Toffoli gates by using $k-2$ dirty (unclean) auxiliary qubits. Both auxiliary qubits return to their initial state after the usage. According to Kunihiro's paper [12], the first conversion is used for all $C^{k}$-NOT gates.

Construction of GT-ADD For $k=0,1, \ldots, n-1$, GT-ADD adds $p$ by repeatedly computing $R_{2} \leftarrow R_{2}+2^{k}$ when $p_{k}=1$. An addition by $2^{k}$ can be realized by $C^{\ell}$-NOT gates $(1 \leq \ell \leq n-k)$ and one NOT gate as in Figure 3. Type 1 Mod-ADD requires, in addition to GT-ADD, $1 / 2 / 3$-controlled GTADD, which consists of NOT gates, C-NOT gates, Toffoli gates, and C ${ }^{\ell}$-NOT gates ( $3 \leq \ell \leq n+3$ ). Both conversions described in Section 2.3 can be used in GT-ADD, however, since it is difficult to allocate clean qubits, Kunihiro used the second conversion for all $\mathrm{C}^{\ell}$-NOT gates [12].

Construction of Q-ADD Q-ADD is an adder using the Quantum Fourier Transform (QFT) [4,7]. For simplicity, we set $\left|R_{2, n}\right\rangle:=\left|R_{1}\right\rangle$ and assume that


Fig. 3: Adder $2^{k}$ to $\left|R_{2}\right\rangle$


Fig. 4: Conversion of 1-controlled $R_{k}$
$\left|R_{2}\right\rangle$ has $n+1$ qubits in this subsection. Also set $p_{n}=0$. Unlike R/GT-ADD, Q-ADD computes $\left|R_{2}\right\rangle \leftarrow\left|R_{2}+p \bmod 2^{n+1}\right\rangle$. Denote the state after applying QFT to the register $\left|R_{2}\right\rangle$ (Figure 9 in [4]) as $\phi\left(\left|R_{2}\right\rangle\right)$. Then, Q-ADD computes in the following way: for $j=n, n-1, \ldots, 0$, and for $k=1,2, \ldots, j+1$, apply the $Z$-rotation gate $R_{k}=\left(1,0 ; 0, e^{2 \pi i / 2^{k}}\right)$ to $\phi\left(\left|R_{2, j}\right\rangle\right)$ when $p_{j-k+1}=1$. Inverse QFT $\left(\mathrm{QFT}^{-1}\right)$ is required to obtain the result of the addition. Thus, Q-ADD can be realized by rotation gates except QFT/ $\mathrm{QFT}^{-1}$.

Type 2 Mod-Add requires $1 / 2$-controlled Q-ADDs, that is, $1 / 2$-controlled $R_{k}$ gates are required. Here, 1 -controlled $R_{k}$ gate can be realized by 2 C-NOTs and 41 -qubit gates, and 2 -controlled $R_{k}$ gate can be realized by 6 C -NOTs and 8 1 -qubit gates [3].

Construction of $1 / 2$-controlled $R_{k}$ is as follows. Arbitrary unitary matrix $W$ can be represented by

$$
\begin{equation*}
W=\Phi(\delta) R z(\alpha) R y(\theta) R z(\beta) \tag{1}
\end{equation*}
$$

for parameters $\alpha, \beta, \theta, \delta \in[0,2 \pi]$, where

$$
\Phi(x)=\left(\begin{array}{cc}
e^{i x} & 0 \\
0 & e^{i x}
\end{array}\right), R y(x)=\left(\begin{array}{cc}
\cos x / 2 & \sin x / 2 \\
-\sin x / 2 & \cos x / 2
\end{array}\right), \quad R z(x)=\left(\begin{array}{cc}
e^{i x / 2} & 0 \\
0 & e^{-i x / 2}
\end{array}\right)
$$

Then 1-controlled $W$ gate can be represented as in Figure 4, where

$$
\begin{aligned}
& A=R z(\alpha) R y(\theta / 2), B=R y(-\theta / 2) R z(-(\alpha+\beta) / 2), \\
& C=R z((\beta-\alpha) / 2), E=R z(-\delta) \Phi(\delta / 2)
\end{aligned}
$$

Thus, 1-controlled $R_{k}$ can be realized by 2 C-NOTs and 41 -qubit gates as in Figure 4 by determining parameters $\alpha, \beta, \theta, \delta$. Similarly, 2-controlled $R_{k}$ gate can be realized by 6 C -NOTs and 81 -qubit gates from Lemma 6.1 in [3].

### 2.4 Required Resources

This section summarizes the resources required in Shor's circuit to factor an $n$-bit integer.

Shor's circuit has three main circuits, Hadamard, Mod-EXP, and QFT ${ }^{-1}$. Required number of gates for each of Hadamard and $\mathrm{QFT}^{-1}$ is $O\left(n^{2}\right)$, while ModEXP requires $G_{\text {ModExP }}(\mathrm{R}-\mathrm{ADD})=270 n^{3}$ with R-ADD, $G_{\text {ModEXP }}(\mathrm{GT}-\mathrm{ADD})=$ $16 / 3 n^{5}$ with GT-ADD, and $G_{\text {ModEXP }}(\mathrm{Q}-\mathrm{ADD})=97 n^{4}$ with Q-ADD. Therefore,
required number of gates for Shor's circuit can be identified by these numbers: $G_{\text {Shor }}(\mathrm{R}-\mathrm{ADD})=270 n^{3}, G_{\text {Shor }}(\mathrm{GT}-\mathrm{ADD})=16 / 3 n^{5}$, and $G_{\text {Shor }}(\mathrm{Q}-\mathrm{ADD})=$ $97 n^{4}$. Unfortunately, no estimation for the depth is known. Required numbers of qubits are $Q_{\text {Shor }}(\mathrm{R}-\mathrm{ADD})=5 n+1$ with R-ADD, and $Q_{\text {Shor }}($ GT-ADD $)=$ $Q_{\text {Shor }}(\mathrm{Q}-\mathrm{ADD})=4 n+2$ with GT-ADD and Q-ADD.

## 3 Implementation of Shor's Quantum Circuit

This section describes how to implement Mod-EXP with each of R-ADD, GTADD, and Q-ADD, respectively, based on Kunihiro's paper [12]. We also show bug-fixes and improvements from them. Moreover, we propose Mod-EXP with MIX-ADD method. This requires $4 n+2$ qubits same as the case of GT/Q-ADD, but consists of fewer gates compared with GT/Q-ADD.

### 3.1 Mod-EXP with R-ADD

We use Type 2 Mod-ADD in order to minimize the number of gates. We also apply the following bug-fixes and improvements.

Bug-fix on Converting $\mathbf{C}^{\mathbf{3}}$-NOT, $\mathbf{C}^{4}$-NOT to Toffoli Gate The first conversion described in Section 2.3 is used in [12] for all $\mathrm{C}^{k}$-NOT ( $k=3,4$ ) gates in $1 / 2$-controlled $\mathrm{R}-\mathrm{ADD}$, however, $k-2$ clean qubits are not available in some cases. In such cases we propose to take the following procedures. When $k=3$ and no clean qubit is available, then use the second conversion described in Section 2.3. When $k=4$, use the second conversion if no qubit is available, and use the conversion described in Figure 6 if 1-qubit is available, which is given by greedy method described later. Compared to the first conversion, 1 Toffoli gate is increased when $k=3$, and $3 / 1$ Toffoli gates are increased when $k=4$ with $0 / 1$ clean qubit. Though this increases the number of gates in Mod-EXP, it does not affect the order since it is at most $O(n)$ (explained later).

Greedy Method Suppose $1 \leq c \leq k-3$ clean qubits and sufficient dirty qubits are available. Our greedy method converts a $\mathrm{C}^{k}$-NOT to Toffoli gates as follows.

1. Generate a null circuit circ.
2. Let $X$ be a set of indices of $k$ control qubits.
3. Select two indices from $X$, and delete these indices from $X$.
4. Select one clean qubit with changing its status as 'dirty' in clean qubit management, and add its index to $X$.
5. Generate a Toffoli gate, controlled by selected indexed-qubits and targeted to the selected clean qubit.
6. Add the generated Toffoli gate to circ.
7. Repeat Step 2-6 ctimes.
8. Generate a $\mathrm{C}^{k-c}$-NOT gate controlled by $(k-c)$ indices in $X$, and targeted to the same qubit as the original $\mathrm{C}^{k}$-NOT gate, and convert to $4(k-c)-8$ Toffoli gates by using the 2nd conversion, and add to circ.
9. Add all Toffoli gates generated in Step 2-7 in the reversed order to circ.
10. Output circ.

The number of Toffoli gates generated by the greedy method is $c+4(k-c)-8+c=$ $4 k-8-2 c$. See Appendix 1 and 2 for examples of our greedy method.

Clean Qubits Management It is difficult to figure out which qubit is clean or not manually when $\mathrm{C}^{k}$-NOT conversion is required. So we implemented the management function to automatically list the status of auxiliary qubits.

- When a quantum gate is added to the circuit, set the status of the target qubit of the gate as 'dirty' (not clean). If the gate makes the status clean (such as CARRY ${ }^{-1}$ ), set 'clean'.
- Use 'clean' qubits in $\mathrm{C}^{k}$-NOT conversion.

This management minimizes the number of gates of Mod-EXP.

The Number of Gates after Bug-fix In Step 2 of Shor's algorithm, we apply the Hadamard gate to the $m$-qubit sequence. By changing this operation to applying the Hadamard gate to $x_{i}$ just before each $C\left(x_{i}\right)$-Mod-MUL, $x_{i+1}, \ldots, x_{m-1}$ can be used as clean qubits in $C\left(x_{i}\right)$-Mod-MUL. Thus, for $i=$ $0, \ldots, m-3, x_{i+1}, x_{i+2}$ can be used as clean qubits and there is no increase on the number of gates because the first conversion can be applied same as in the Kunihiro's paper [12]. On the other hand, when $i=m-2, m-1$, available clean qubits are less than 2 , and additional circuits are required.

The number of gates for Bug-fix Mod-EXP with R-ADD is given as follows. $C\left(x_{i}, y_{j}\right)$-Mod-ADD in Mod-EXP consists of $3 C^{2}$-R-ADDs, $1 C$-R-ADD, $1 C^{2}$ NOT, $2 C$-NOTs and 3 NOTs. In the case for $i=0, \ldots, m-3, C^{4} / C^{3}$-NOT is converted to $5 / 3$ Toffoli gates because two clean qubits are available. Hence, $C\left(x_{i}, y_{j}\right)$-Mod-ADD consists of $135 / 2 n-155 / 2$ elementary gates. For $i=m-$ $2, m-1$, we count the number of gates to be added from this number. For $i=m-2$, there is no additional gates in the first and last $C^{2}-\mathrm{R}-\mathrm{ADD}(d)$ in Type $1 C\left(x_{i}, y_{j}\right)$-Mod-ADD, because two clean qubits ( $x_{m-1}$ and $R_{3}$ ) are available. On the other hand, additional gates are required in $C^{2}-\mathrm{R}-\operatorname{ADD}\left(2^{n}-d\right)$ due to lack of clean qubits. Specifically, two clean qubits $x_{m-1}$ and $c_{n-1}$ are available in $C^{2}$-CARRY for $c_{j}$ and $C^{2}$-CARRY ${ }^{-1}$ for $c_{j}$ for $j=1, \ldots, n-2$, but just one in $C^{2}$-CARRY for $c_{n-1}, c_{n}$ and $C^{2}$-CARRY ${ }^{-1}$ for $c_{n-1}$. Each $C^{4}$-NOT gate in these three CARRYs is converted to 6 Toffoli gates by the greedy method for $k=4, c=1$. This leads to the addition of 3 elementary gates compared to the case for $i=0, \ldots, m-3$. Hence, $C\left(x_{i}, y_{j}\right)$-Mod-ADD consists of $135 / 2 n-149 / 2$ elementary gates. For $i=m-1$, additional gates are required as shown in Table 1, then $C\left(x_{i}, y_{j}\right)$-Mod-ADD consists of $135 / 2 n-55$ elementary gates. Therefore,

|  | $C^{2}-\mathrm{ADD}(d)$ at Step 1-1 and $7 \mid$ |  |  | $C$-ADD at Step 3 |  |  | $C^{2}-\operatorname{ADD}\left(2^{n}-d\right)$ at Step 5 |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | required | available | \#gates | required | available | \#gates | required | available | \#gates |
| CARRY $c_{n-2}$ | 2 | $R_{3}, c_{n-1}$ | +0 | 1 | $c_{n-1}$ | +0 | 2 | $c_{n-1}$ | +1 |
| CARRY $c_{n-1}$ | 2 | $R_{3}$ | +1 | 1 | - | +1 | 2 | - | +7/2 |
| $\text { CARRY } R_{1}$ | 2 | $R_{3}$ | +1 | 1 | - | +1 | 2 | - | +7/2 |
| SUM $R_{2, n-1}$ | 2 | $R_{3}$ | +0 | 0 | - | +0 | 1 | - | +1 |
| $\mathrm{CARRY}^{-1} c_{n-1}$ | 2 | $R_{3}$ | +1 | 1 | - | +1 | 2 | - | +7/2 |
| $\mathrm{CARRY}^{-1} c_{n-2}$ | 2 | $R_{3}, c_{n-1} \mid$ | +0 | 1 | $c_{n-1}$ | +0 | 2 | $c_{n-1}$ | +1 |

Table 1: The number of required clean qubits, available clean qubits and the number of additional gates in each controlled ADD in Mod-ADD Type-2 [12] with R-ADD for $i=m-1$

Bug-fix Mod-EXP with R-ADD consists of

$$
\begin{aligned}
G_{\text {ModEXP }}(\mathrm{R}-\mathrm{ADD})= & 2 n(m-2)(135 / 2 n-155 / 2)+2 n(135 / 2 n-149 / 2) \\
& +2 n(135 / 2 n-55)+3 m n \\
= & 270 n^{3}-304 n^{2}+51 n
\end{aligned}
$$

elementary gates, where $3 m n$ is the number of elementary gates for C-SWAPs in Mod-MUL. The gates increased by the lack of clean qubits is at most $O(n)$.

### 3.2 Mod-EXP with GT-ADD

For implementing Mod-EXP with GT-ADD, Type 1 Mod-ADD is used to minimize the number of gates. Kunihiro used the second conversion described in Section 2.3 for converting $\mathrm{C}^{k}$-NOT gates (for $3 \leq k \leq n+3$ ) to Toffoli gates. This paper proposes to use clean qubits as much as possible by the greedy method to decrease the number of gates.

Greedy Method in Mod-EXP For all conversions from $\mathrm{C}^{k}$-NOT gates $(3 \leq$ $k \leq n+3$ ) to Toffoli gates appeared in Mod-EXP with GT-ADD, we use the 1st conversion described in Section 2.3 when more than or equal to $k-2$ clean qubits are available, the greedy method when 1 to $k-3$ clean qubits are available, and the 2nd conversion when no clean qubit is available. We also use the clean qubit management in the greedy method.

The Number of Gates with Greedy Method The number of gates for Mod-EXP with GT-ADD with the greedy method is given as follows. Type $1 C\left(x_{i}, y_{j}\right)$-Mod-ADD consists of the following four gates. Each gate can be converted to elementary gates as shown in Case 1-4.

1. $C^{3}$-GT-ADD with $m-i-1$ cleans,
2. $2 C^{3}$-GT-ADDs with $m-i$ cleans,
3. $C^{2}$-GT-ADD with $m-i-1$ cleans,
4. $2 C^{3}$-NOTs and $4 C^{2}$-NOTs.

Case 1. $C^{3}$-GT-ADD consists of $(n-k+4) / 2 C^{k}$-NOTs $(4 \leq k \leq n+3)$ and $n / 2$ $C^{3}$-NOTs on average. In the case for $0 \leq i \leq n-2$, all $C^{k}$-NOTs can be converted to Toffoli gates by the 1 st conversion because $n+1$ clean qubits are available. Hence, the number of gates is given as $n_{1}(i)=1 / 2 \sum_{k=4}^{n+3}(n-k+4)(2 k-3)+3 / 2 n$. For $n-1 \leq i \leq m-2$, we convert $C^{k}$-NOT to Toffoli gates by the 1st conversion for $3 \leq k \leq m-i+1$ and the greedy method for $m-i+2 \leq k \leq n+3$. Hence, the number of gates is $n_{1}(i)=1 / 2 \sum_{k=m-i+2}^{n+3}(n-k+4)(4 k-8-2(m-i-1))+$ $1 / 2 \sum_{k=4}^{m-i+1}(n-k+4)(2 k-3)+3 / 2 n$. For $i=m-1$, we use the 2 nd conversion, then the number of gates is $n_{1}(i)=1 / 2 \sum_{k=4}^{n+3}(n-k+4)(4 k-8)+3 / 2 n$.

Case 2. In the same way as Case 1, the number of gates is $n_{2}(i)=1 / 2 \sum_{k=3}^{n+2}(n-$ $k+3)(2 k-3)+n / 2$ for $0 \leq i \leq n, n_{2}(i)=1 / 2 \sum_{k=m-i+3}^{n+2}(n-k+3)(4 k-8-$ $2(m-i))+1 / 2 \sum_{k=3}^{m-i+2}(n-k+3)(2 k-3)+n / 2$ for $n+1 \leq i \leq m-1$.

Case 3. In the same way as Case 1, the number of gates is $n_{3}(i)=1 / 2 \sum_{k=3}^{n+2}(n-$ $k+3)(2 k-3)+n / 2$ for $0 \leq i \leq n-1, n_{3}(i)=1 / 2 \sum_{k=m-i+2}^{n+2}(n-k+3)(4 k-$ $8-2(m-i-1))+1 / 2 \sum_{k=3}^{m-i+1}(n-k+3)(2 k-3)+n / 2$ for $n \leq i \leq m-2$, and $n_{3}(i)=1 / 2 \sum_{k=3}^{n+2}(n-k+3)(4 k-8)+n / 2$ for $i=m-1$.

Case 4. Each $C^{3}$-NOT can be converted to 3 Toffoli gates for $0 \leq i \leq m-2$, and 4 for $i=m-1$.

Mod-EXP with GT-ADD with the greedy method consists of

$$
\begin{aligned}
G_{\mathrm{ModEXP}}(\mathrm{GT}-\mathrm{ADD}) & =2 n\left\{\sum_{i=0}^{m-1}\left(n_{1}(i)+n_{2}(i)+n_{3}(i)+4\right)+6(2 n-1)+8\right\}+3 m n \\
& =3 n^{5}+15 n^{4}+\frac{51}{2} n^{3}+\frac{103}{2} n^{2}+8 n
\end{aligned}
$$

elementary gates. The greedy method reduces the fifth-order coefficient from $16 / 3$ to 3 .

### 3.3 Mod-EXP with Q-ADD

This subsection describes how to implement Mod-EXP with Q-ADD.

Bug-fix in Q-ADD Since Q-ADD requires to apply QFT to the registers $\left|R_{1} R_{2}\right\rangle$, QFT just before $C\left(x_{0}\right)$-Mod-MUL in Q-ADD (Figure 2 in [4]), and QFT $^{-1}$ just before C-SWAP and QFT just before C-SWAP in $C\left(x_{i}\right)$-Mod-MUL should be added. Thus the number of gates are increased to $4 n+2$ QFTs for Mod-EXP from the original [12]. Furthermore, the original number of gates did not consider C-SWAP, so that $m n$ Toffoli gates and $2 m n$ C-NOTs should be added. However, since these increase is at most $O\left(n^{3}\right)$, it does not effect on the total number of Mod-EXP at all.


Fig. 5: Conversion of 2-controlled Rotation Gate

Change of Mod-ADD When Type 2 Mod-ADD is used for Q-ADD, 4 QFTs and $4 \mathrm{QFT}^{-1}$ s are required, and the number of gates of Mod-EXP will be increased (the order is same, but the coefficient becomes larger). So, we propose to use Beauregard's Mod-ADD which requires 2 QFTs and $2 \mathrm{QFT}^{-1} \mathrm{~s}$ [4].

Gate Reduction of Controlled Rotation Gate Conversion When 1/2controlled $R_{k}$ gates are converted to elementary gates, one 1-qubit gate can be reduced by setting parameters properly. In fact, set $\alpha=\beta=-\pi / 2^{k}, \theta=0, \delta=$ $\pi / 2^{k}$ in (1) for $W=R_{k}$, then $C$ becomes an identity matrix and can be omitted. Similarly, setting $\alpha=\beta=-\pi / 2^{k-1}, \theta=0, \delta=\pi / 2^{k-1}$ for 2-controlled $R_{k}$ gates reduces one 1-qubit gate as in Figure 5, where ${ }^{\dagger}$ denotes an inversion.

The Number of Gates after Bug-fix Mod-EXP consists of $2 m n$ Beauregard's Mod-ADDs, $2 m+2$ QFTs (or QFT ${ }^{-1}$ ) and $m n$ C-SWAPs. Mod-ADD also consists of as follows.
$-3 C^{2}$-Q-ADDs, each of which consists of $(n+2-k) / 2 C^{2}-R_{k}$ for $1 \leq k \leq n+1$,

- C-Q-ADD, which consists of $(n+2-k) / 2$ of $C-R_{k}$ for $1 \leq k \leq n+1$,
- Q-ADD, which consists of $(n+2-k) / 2$ of $R_{k}$ for $1 \leq k \leq k$,
-4 QFTs, each QFT consists of $n+1 \mathrm{H}$ gates and $n+2-k C-R_{k}$ for $2 \leq k \leq n+1$,
- $2 C$-NOTs and 2 NOTs.

Hence, Mod-ADD consists of $G_{\text {ModADD }}(\mathrm{Q}-\mathrm{ADD})=21 n^{2} / 4+47 n / 4+21 / 2$ elementary gates because $C^{2} / C-R_{k}$ can be converted to $13 / 5$ elementary gates. And QFT consists of $G_{\mathrm{QFT}}(n+1)=5 / 2 n^{2}+7 n / 2+1$ elementary gates. Therefore, Mod-EXP with Q-ADD consists of

$$
\begin{aligned}
G_{\mathrm{ModEXP}}(\mathrm{Q}-\mathrm{ADD}) & =2 m n \times G_{\mathrm{ModADD}}(\mathrm{Q}-\mathrm{ADD})+(2 m+2) \times G_{\mathrm{QFT}}(n+1)+3 m n \\
& =85 n^{4}+201 n^{3}+147 n^{2}+11 n+2
\end{aligned}
$$

elementary gates. The fourth-order coefficient is reduced from 97 to 85 by the gate reduction.

### 3.4 Mod-EXP with MIX-ADD

This subsection proposes a MIX-ADD method, which uses different ADD methods in Mod-EXP depending on the number of available clean qubits to minimize the number of elementary gates.

Definition of MIX-ADD The original Mod-EXP uses just one ADD circuit such as $\mathrm{R} / \mathrm{GT} / \mathrm{Q}-\mathrm{ADD}$, but in the case of $4 n+2$ qubits circuit, the number of gates for Mod-EXP can be reduced by selecting the optimal ADD for each ADD in Mod-EXP. We call this construction Mod-EXP with MIX-ADD. Considering the order of the number of gates for each ADD, R-ADD is top priority, next QADD, then GT-ADD. However, R-ADD is available only if $n-1$ clean auxiliary qubits are available as carry qubits. In $C\left(x_{i}\right)$-Mod-MUL in Mod-EXP, we can use R-ADD for $0 \leq i \leq n$ because $m-i+1$ clean qubits ( $x_{i+1}, \ldots, x_{m-1}$ ) are available. On the other hand, we use Q-ADD for $n+1 \leq i \leq m-1$ to minimize the number of gates. In applying Q-ADD from the middle of Mod-EXP, QFT is added in the following three points. The first is after $C\left(x_{n-2}\right)$-Mod-MUL, the second is QFT ${ }^{-1}$ before C-SWAP and QFT after C-SWAP in Mod-MUL for $n+1 \leq i \leq m-1$, and the third is after $C\left(x_{m-1}\right)$-Mod-MUL.

The Number of Gates The number of gates for Mod-EXP with MIX-ADD is computed in the same way as in Section 3.1 and Section 3.3. Therefore, we have

$$
\begin{aligned}
& G_{\text {ModEXP }}(\text { MIX-ADD })=2 n(n-1)(135 / 2 n-155 / 2)+2 n(135 / 2 n-149 / 2) \\
&+2 n(135 / 2 n-55)+2 n(n-1) \times G_{\mathrm{ModADD}}(\mathrm{Q}-\mathrm{ADD}) \\
&+2 n \times G_{\mathrm{QFT}}(n+1)+3 m n \\
&=\frac{85}{2} n^{4}+193 n^{3}-\frac{83}{2} n^{2}-163 n,
\end{aligned}
$$

which is about half the number of gates for Mod-EXP with Q-ADD.

## 4 Experimental Results

This section reports our factorization results based on our implementation described in Section 3 by using the quantum computer simulator mpiQulacs [11], a distributed version of the quantum simulator Qulacs [19]. We used an A64FXbased cluster system similar to Todoroki [11] with 512 nodes, which enables 39 -qubit operations. A64FX is an ARM-based CPU that is also equipped in the world's top Fugaku supercomputer.

The experiments were conducted by the following steps:

1. For an $n$-bit RSA-type composite number (a product of two different odd primes) $N$, choose $a$ which induces the factorization (for efficiency reason).
2. Generate the quantum circuit for factoring $N$ by Shor's algorithm. Here we have four choices for ADD circuit.
3. Input the quantum circuit to the simulator.
4. Observe the 1st bit sequence 10,000 times to estimate the order $r$.
5. Output $\operatorname{gcd}\left(a^{r / 2} \pm 1, N\right)$.

Note that, since the observation in Step 4 does not destroy the quantum state, it is sufficient to run each quantum circuit once in the experiments.

|  |  | R-ADD |  |  |  | GT-ADD |  |  |  | Q-ADD |  |  |  | MIX-ADD |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $N$ |  | $a \mid Q$ | $Q \quad G$ | D | $T$ | $Q$ | G | D | T | $Q$ | G | D | $T$ | $Q$ | $G$ | D | $T$ |
| 15 | 42 | 221 | 2112937 | 10507 | 2.4 | 18 | 12595 | 9838 | 0.91 | 18 | 38967 | 20208 | 3.5 | 18 | 22815 | 14273 | 1.6 |
| 21 | 5 | 226 | 2626155 | 20779 | 89.9 | 22 | 25325 | 18824 | 5.2 | 22 | 78334 | 40409 | 18 | 22 | 47273 | 28866 | 10.4 |
| 33 | 6 | 531 | 3146935 | 36870 |  | 26 | 44461 | 31436 | 92 | 26 | 145620 | 76578 | 404 | 26 | 87251 | 53343 | 228 |
| 35 | 6 | 231 | 3147662 | 37775 |  | 26 | 55387 | 38869 | 115 | 26 | 155329 | 79693 | 426 | 26 | 93174 | 55541 | 241 |
| 39 | 6 | 231 | 3147843 | 38214 |  | 26 | 61941 | 43483 | 129 | 26 | 160315 | 81152 | 441 | 26 | 95233 | 56408 | 6 |
| 51 | 6 | 231 | 3146991 | 37413 |  | 26 | 55755 | 39348 | 116 | 26 | 152468 | 78285 | 421 | 26 | 90995 | 54677 | 237 |
| 55 | 6 | 231 | 3147845 | 38513 |  | 26 | 61899 | 43507 | 129 | 26 | 160613 | 80877 | 441 | 26 | 95368 | 56384 | 246 |
| 57 | 6 | 531 | 3147555 | 38028 |  | 26 | 51360 | 36346 | 107 | 26 | 154085 | 78686 | 431 | 26 | 91616 | 55062 | 238 |
| 65 | 73 | 336 | 3676341 | 59902 |  | 30 | 82676 | 56199 | 2430 | 30 | 251424 | 132329 | 10545 | 30 | 150521 | 90940 | 5915 |
| 6 | 72 | 236 | 3678035 | 61939 |  | 30 | 98774 | 66690 | 2866 | 30 | 271832 | 138888 | 11329 | 30 | 162705 | 95730 | 6362 |
| 7 | 72 | 236 | 3677066 | 61391 |  | 30 | 104285 | 70616 | 3033 | 30 | 267042 | 135177 | 11125 | 30 | 159450 | 93522 | 6275 |
| 8 | 72 | 236 | 3675704 | 60041 |  | 30 | 99407 | 67570 | 2906 | 30 | 256625 | 132179 | 10719 | 30 | 153316 | 91241 | 6011 |
| 8 | 72 | 236 | 3678196 | 62751 |  | 30 | 120027 | 80999 | 3485 | 30 | 284083 | 142164 | 11792 | 30 | 167554 | 97300 | 6524 |
| 9 | 72 | 236 | 3677819 | 62369 |  | 30 | 116234 | 78729 | 3369 | 30 | 279204 | 141000 | 11594 | 30 | 165151 | 96642 | 6435 |
| 93 | 72 | 236 | 3677659 | 62319 |  | 30 | 108070 | 73227 | 3150 | 30 | 276912 | 140313 | 11516 | 30 | 163710 | 96243 | 6380 |
| 95 | 72 | 236 | 3678550 | 63480 |  | 30 | 125960 | 85061 | 3664 | 30 | 289797 | 144364 | 12098 | 30 | 169991 | 98446 | 6610 |
| 111 | 7 | 236 | 3678692 | 63633 |  | 30 | 124959 | 84533 | 3646 | 30 | 289793 | 144261 | 12020 | 30 | 170163 | 98552 | 6648 |
| 115 | 72 | 236 | 3678591 | 63151 |  | 30 | 109922 | 74503 | 3188 | 30 | 282238 | 141557 | 11809 | 30 | 168210 | 97753 | 6568 |
| 119 | 72 | $2{ }^{2} 36$ | 3678563 | 63477 |  | 30 | 122960 | 83264 | 3577 | 30 | 287020 | 142555 | 11978 | 30 | 170386 | 98332 | 6620 |
| 123 | 72 | 236 | 3678691 | 63672 |  | 30 | 118337 | 80519 | 3452 | 30 | 286730 | 143475 | 11899 | 30 | 170798 | 99083 | 6643 |

Table 2: Factorization of $N$ up to 7 -bit (with 1-node).

### 4.1 Naive Circuit

Firstly, we factored small RSA-type composite numbers up to 7 -bit with 1 -node by using Shor's quantum circuits generated by our implementation. Table 2 shows the required resources and timings for factorization, where $Q, G, D, T$ denote the number of required qubits, the number of elementary gates, the depth of Shor's circuit, and the timing data in seconds. Since we used 1-node only, 30 qubits are available for factorization. Thus, circuits with R-ADD for 6-bit and 7 -bit integers cannot be proceeded (denoted by '-' in the table).

As in the table, required resources depend on the parameters $N$ and $n$, but on $n$ mainly. The ratio $D / G$ seems to be a constant depending on the features of R-ADD, GT-ADD, Q-ADD, and MIX-ADD. Since Q-ADD has many 1-qubit operations and is easy to parallelize, so that the ratio $D / G$ is smaller ( $0.50-$ 0.53 for $\mathrm{Q}-\mathrm{ADD}$ and $0.57-0.63$ for MIX-ADD) compared to other ADDs ( 0.79 to 0.81 for R-ADD, 0.68-0.79 for GT-ADD). Though $G$ and $D$ are expected in the following order, $O\left(n^{3}\right)$ for R-ADD, $O\left(n^{4}\right)$ for Q-ADD and MIX-ADD, and $O\left(n^{5}\right)$ for GT-ADD, the results differ from expected ones. The reason is that the composite numbers are so small that other terms rather than the dominant term affect. The difference may be smaller for larger parameters.

### 4.2 Optimized Circuit

Then, we factor 8 -bit and 9 -bit integers with 512 -nodes. GT-ADD is used for the experiment because it requires less number of qubits and gates compared to other ADDs in the case of these small integers. In order to decrease the number of gates and the depth as much as possible, we used optimize_light option of

| $N$ | $n$ | $a$ | $Q$ | G | D | $T$ | $N$ | a | a $Q$ | Q G | D | $T$ | $N$ | $n$ | a Q | G | D | $T$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 129 | 8 |  | 734 | 152780 | 100141 | 256 | 259 | 92 | 238 | 8288684 | 183065 | 6143 | 395 | 9 | 238 | 319088 | 203494 | 7307 |
| 133 | 8 | 2 | 234 | 169108 | 111205 | 247 | 265 | 9 | 638 | 8272685 | 173346 | 5620 | 403 | 9 | 238 | 307506 | 195485 | 6271 |
| 141 | 8 |  | 234 | 183453 | 120170 | 287 | 267 | 9 | 238 | 8309270 | 196137 | 6572 | 407 | 9 | 238 | 338095 | 214301 | 7907 |
| 143 | 8 |  | 234 | 207514 | 135907 | 311 | 287 | 9 | 238 | 8359003 | 228259 | 7511 | 411 | 9 | 238 | 335319 | 214006 | 7404 |
| 145 | 8 |  | 634 | 158918 | 105271 | 262 | 291 | 9 | 238 | 8308155 | 195603 | 6542 | 413 | 9 | 238 | 327370 | 208569 | 6648 |
| 15 | 8 |  | 234 | 198473 | 130150 | 311 | 295 | 9 | 238 | 8334848 | 212590 | 6370 | 415 | 9 | 238 | 359587 | 228199 | 7723 |
| 15 | 8 |  | 234 | 217743 | 142924 | 335 | 299 | 9 | 238 | 8321523 | 204402 | 7094 | 417 | 9 | 538 | 267426 | 171328 | 5940 |
| 161 | 8 |  | 334 | 155238 | 103030 | 238 | 301 | 9 | 238 | 8317493 | 202575 | 6461 | 427 | 9 | 238 | 324243 | 207582 | 6862 |
| 177 | 8 |  | 534 | 168876 | 111997 | 259 | 303 | 9 | 238 | 8353151 | 224856 | 7559 | 437 | 9 | 238 | 314856 | 200771 | 5925 |
| 183 | 8 |  | 234 | 207468 | 136410 | 297 | 305 | 9 | 338 | 8285798 | 182560 | 6350 | 445 | 9 | 238 | 339458 | 216426 | 6572 |
| 185 | 8 |  | 334 | 180752 | 119593 | 282 | 309 | 9 | 238 | 8309354 | 196737 | 6358 | 447 | 9 | 238 | 373035 | 237421 | 7448 |
| 187 | 8 |  | 234 | 208281 | 137192 | 328 | 319 | 9 | 238 | 8367923 | 233944 | 7419 | 451 | 9 | 238 | 306484 | 195876 | 5999 |
| 201 | 8 |  | 734 | 170050 | 112064 | 244 | 321 | 9 | 738 | 8260877 | 166496 | 5899 | 453 | 9 | 238 | 286538 | 183164 | 6146 |
| 203 | 8 |  | 234 | 193163 | 126762 | 285 | 323 | 9 | 238 | 8304490 | 193554 | 5956 | 469 | 9 | 238 | 303229 | 193946 | 6246 |
| 205 | 8 |  | 334 | 178117 | 117326 | 276 | 327 | 9 | 238 | 8322336 | 204745 | 6115 | 471 | 9 | 238 | 343707 | 219148 | 7473 |
| 209 | 8 |  | 334 | 165014 | 109327 | 243 | 329 | 9 |  | 8285506 | 182113 | 6099 | 473 | 9 | 338 | 303975 | 194528 | 6933 |
| 213 | 8 |  | 234 | 184210 | 121450 | 272 | 335 | 9 | 238 | 8349246 | 222013 | 8104 | 481 | 9 | 338 | 281077 | 180267 | 6815 |
| 215 | 8 |  | 234 | 204621 | 134697 | 327 | 339 | 9 | 238 | 8317273 | 201779 | 7109 | 485 |  | 238 | 305606 | 195586 | 6502 |
| 217 | 8 |  | 534 | 178741 | 118044 | 255 | 341 | 9 | 238 | 8291468 | 186213 | 6363 | 489 |  | 738 | 302012 | 193333 | 7218 |
| 219 | 8 |  | 234 | 204160 | 134522 | 299 | 355 | 9 | 238 | 8310783 | 197410 | 7491 | 493 |  | 238 | 329162 | 210756 | 6188 |
| 221 | 8 |  | 234 | 200121 | 131790 | 283 | 365 | 9 | 238 | 8322926 | 206125 | 6346 | 497 | 9 | 338 | 296472 | 189877 | 5750 |
| 235 | 8 |  | 234 | 198443 | 130597 | 285 | 371 | 9 | 238 | 8324641 | 206674 | 6287 | 501 |  | 238 | 322414 | 207063 | 6335 |
| 237 | 8 |  | 234 | 193348 | 127347 | 286 | 377 | 9 | 338 | 8316691 | 202612 | 6676 | 505 |  | 638 | 313370 | 200596 | 6811 |
| 247 | 8 |  | 234 | 208086 | 136900 | 289 | 381 | 9 | 238 | 8321134 | 204686 | 5860 | 511 | 9 | 338 | 395310 | 252188 | 8226 |
| 249 | 8 | 11 | 134 | 186487 | 123502 | 292 | 391 | 9 | 238 | 8326281 | 207709 | 6697 |  |  |  |  |  |  |
| 253 | 8 |  | 234 | 202159 | 133987 | 306 | 393 | 9 | 538 | 8281956 | 17987 | 6014 |  |  |  |  |  |  |

Table 3: Factorization of $N$ up to 9 -bit with GT-ADD (with 512-nodes).

Qulacs which unifies successive 1-qubit gates to one gate. However, the effect was very limited: it reduce the number of gates by only 1 percent.

Since factorization of 9 -bit integers require 38 -qubits, and 256 -nodes are sufficient for the computation, other 256 -nodes can be used for the speed-up. To do so, we used the fused_swap_option option of mpiQulacs which enables to distribute tasks to identified nodes for efficient computation.

Table 3 summarizes the factorization results. As in the table, we have succeeded factoring all RSA-type integers up to 9 -bit. The largest integer we factored here was $N=511$, which requires 8226 seconds ( 2.3 hours). On the other hand, optimize_light option works very well for $\mathrm{Q}-\mathrm{ADD}$, since $\mathrm{Q}-\mathrm{ADD}$ uses a lot of successive 1-qubit gates. In fact, the optimized quantum circuit for factoring $N=511$ with Q-ADD requires 225523 gates and 187618 depth, and it factors $N=511$ in 7050 seconds ( 1.96 hours) in the experiment.

### 4.3 Resource Estimation of Basic Circuit

Finally, we estimated the quantum circuit resources for factoring 1024 -bit and 2048 -bit integers. For each $8 \leq n \leq 24$, we generated 10 composite numbers $N$ randomly ( 170 composite numbers in total). Then, we generated the quantum circuit for each $N$ with the optimize_light option, and evaluated the number of elementary gates and the depth. Here, we used R-ADD since resources become smaller than others for larger $N$ 's. Next, we computed the average of resources for each $n$. See Appendix 3 for the detailed values from this experiment.

|  | $n=1024$ |  |  | $n=2048$ |  |  |
| ---: | ---: | ---: | ---: | ---: | ---: | ---: |
|  | qubits | gates | depth | qubits | gates | depth |
| Kunihiro [12] | 3074 | $2.90 \times 10^{11}$ | - | 6146 | $2.32 \times 10^{12}$ | - |
| R-ADD | 5121 | $2.74 \times 10^{11}$ | $2.20 \times 10^{11}$ | 10241 | $2.19 \times 10^{12}$ | $1.76 \times 10^{12}$ |
| GT-ADD | 4098 | $3.33 \times 10^{15}$ | $1.02 \times 10^{15}$ | 8194 | $1.07 \times 10^{17}$ | $3.23 \times 10^{16}$ |
| Q-ADD | 4098 | $2.87 \times 10^{13}$ | $2.43 \times 10^{13}$ | 8194 | $4.58 \times 10^{14}$ | $3.88 \times 10^{14}$ |
| MIX-ADD | 4098 | $1.49 \times 10^{13}$ | $1.26 \times 10^{13}$ | 8194 | $2.37 \times 10^{14}$ | $2.00 \times 10^{14}$ |

Table 4: Circuit estimation for factoring 1024/2048-bit integers

From average values for $8 \leq n \leq 24$, we obtain approximation polynomials

$$
\begin{aligned}
& G_{\mathrm{R}-\mathrm{ADD}}(n)=254.84981 n^{3}-338.63513 n^{2}-177.31878 n+3112.36316, \\
& D_{\mathrm{R}-\mathrm{ADD}}(n)=204.72160 n^{3}-265.74807 n^{2}-515.61678 n+5232.47162,
\end{aligned}
$$

using least squares method with assuming that $G(n)=O\left(n^{3}\right)$ and $D(n)=$ $O\left(n^{3}\right)$. Then, by substituting $n=1024$ and $n=2048$ to these polynomials, we obtain approximations as in Table 4. Compared to the estimation by Kunihiro, our estimation decreases by about $5.6 \%$ for the number of gates. We do not discuss the feasibility of such a huge quantum computer, however, if the quantum circuit for factoring a 2048 -bit integer is proceeded by an ideal quantum computer which can proceed the operation in the same speed as Google's Sycamore [2], that took 200 seconds to sample $10^{6}$ times with a circuit with depth 40 , factoring requires about 101.70 days, which seems infeasible by the current quantum technology.

As in the R-ADD case, we obtain the approximation polynomials for GTADD, Q-ADD and MIX-ADD

$$
\begin{aligned}
G_{\mathrm{GT}-\mathrm{ADD}}(n) & =2.931 n^{5}+20.169 n^{4}, & D_{\mathrm{GT}-\mathrm{ADD}}(n) & =0.883 n^{5}+21.875 n^{4}, \\
G_{\mathrm{Q}-\mathrm{ADD}}(n) & =25.983 n^{4}+59.060 n^{3}, & D_{\mathrm{Q}-\mathrm{ADD}}(n) & =21.993 n^{4}+44.503 n^{3}, \\
G_{\mathrm{MIX-ADD}}(n) & =13.378 n^{4}+136.287 n^{3}, & D_{\mathrm{MIX}-\mathrm{ADD}}(n) & =11.309 n^{4}+107.630 n^{3},
\end{aligned}
$$

with assuming that $G(n)=O\left(n^{k}\right)$ and $D(n)=O\left(n^{k}\right)$ for $k=5,4,4$, respectively. Since $k$ is large, we compute the approximation polynomials only in the upper two degrees. We obtain approximations for $n=1024$ and 2048 as in Table 4. Factoring a 2048-bit composite number requires about 5107, 61.4 and 31.7 years (GT, Q and MIX-ADD, respectively). MIX-ADD requires less time than GT/QADD, but more time than R-ADD. However, MIX-ADD is useful in environments where the number of available qubits is limited since MIX-ADD requires fewer qubits than R-ADD.

## 5 Concluding Remarks

In this paper, we have proposed the MIX-ADD method that can flexibly select the optimal ADD circuit for each of the ADD circuits in the Mod-EXP.

This method reduces the number of elementary gates and the depth in Shor's quantum circuit while maintaining a lower qubit requirement compared to R ADD. Next, we have implemented Shor's algorithm for factoring general composite numbers using 4 different ADD (R-ADD, GT-ADD, Q-ADD and MIX-ADD), and successfully factored 96 RSA-type composite numbers up to 9 -bit using the quantum computer simulator developed by Fujitsu. Finally, we have estimated the number of gates and depth required of Shor's quantum circuit for larger composite numbers by actually generating quantum circuits, and gave the estimation for 1024 and 2048-bit integers.

A new finding obtained from our experiments is that the required resources related to Shor's algorithm can be evaluated based on actual implementation rather than theoretical analysis, at least for small parameters, by using the quantum simulator. The effectiveness of improvements can be assessed through actual implementation and experiments on quantum simulators.

Our implementations are based on the basic construction of Shor's quantum circuit. Future work will involve experiments and resource estimation using advanced circuits that apply complex techniques to reduce the number of qubits, as well as under noisy conditions.

## Appendix 1. Examples of Greedy Method

Figure 6 shows an example of our greedy method for $k=4, c=1$, and Figure 7 for $k=5, c=1,2$. The number of Toffoli gates is 6 for $k=4, c=1,8$ for $k=5, c=2$, and 10 for $k=5, c=1$, which matches $4 k-8-2 c$.

## Appendix 2. Effectiveness of Greedy Method

In order to show the superiority of our greedy method, we factored RSA-type composite numbers up to 7 -bit with 1-node, without and with the greedy method for GT-ADD. Results are summarized in Table 5, where results in the 'Greedy' column coincide with the results shown at 'GT-ADD' column in Table 2. As shown in the table, the greedy method reduces the number of gates to about $66-71 \%$, and the depth to about $45-56 \%$. Since the generated Toffoli gates by the greedy method can be parallelized easily, the effect on the depth is much larger than that on the number of gates. Our analysis in Section 3.2 shows that the greedy method reduces the number of gates to about $56.25 \%$ (calculated as $3 /(16 / 3) \times 100)$ when $n$ is sufficiently large.

## Appendix 3. Data for circuit estimation in Section 4.3

Figure 8 shows the average values and the approximation polynomials described in Section 4.3. Table 6 summarizes the average values, lowest values, and highest values for the R-ADD case. There is virtually no difference between them.


Fig. 6: Conversion from a $\mathrm{C}^{4}$-NOT to $\mathrm{C}^{2}$-NOTs with 1 clean qubit


Fig. 7: Conversion from a $\mathrm{C}^{5}$-NOT to $\mathrm{C}^{2}$-NOTs

## Acknowledgments

We would like to thank Shintaro Sato, Hirotaka Oshima, Kazunori Maruyama, Masayoshi Hashima and Kohta Nakashima from Fujitsu for their efforts in using the quantum computer simulator. The sixth author conducted the research supported by JST CREST JPMJCR2113 and JPSP KAKENHI Grant Number JP21H03440.

## References

1. Amico, M., Saleem, Z.H., Kumph, M.: Experimental study of Shor's factoring algorithm using the IBM Q Experience. Physical Review A 100(1), 012305 (2019). https://doi.org/10.1103/PhysRevA.100.012305
2. Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J.C., Barends, R., et al.: Quantum supremacy using a programmable superconducting processor. Nature 574(7779), 505-510 (2019). https://doi.org/10.1038/s41586-019-1666-5
3. Barenco, A., Bennett, C.H., Cleve, R., DiVincenzo, D.P., Margolus, N., Shor, P., et al.: Elementary gates for quantum computation. Physical review A 52(5), 3457 (1995). https://doi.org/10.1103/PhysRevA.52.3457
4. Beauregard, S.: Circuit for shor's algorithm using $2 \mathrm{n}+3$ qubits. arXiv preprint quant-ph/0205095 (2002). https://doi.org/10.48550/arXiv.quant-ph/0205095
5. Beckman, D., Chari, A.N., Devabhaktuni, S., Preskill, J.: Efficient networks for quantum factoring. Physical Review A 54(2), 1034 (1996). https://doi.org/10. 1103/PhysRevA.54.1034
6. Boudot, F., Gaudry, P., Guillevic, A., Heninger, N., Thomé, E., Zimmermann, P.: Factorization of RSA-250. https://web.archive.org/web/20200228234716/https: //lists.gforge.inria.fr/\pipermail/cado-nfs-discuss/2020-February/001166.html (2020)
7. Draper, T.G.: Addition on a quantum computer. arXiv preprint quant-ph/0008033 (2000). https://doi.org/10.48550/arXiv.quant-ph/0008033

| GT-ADD |  |  | No Greedy |  |  | Greedy |  |  | Ratio |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $N$ | $n \mid a$ | a $Q$ | $G_{0}$ | $D_{0}$ | $T_{0}$ |  | $D_{1}$ | $T_{1}$ |  | $D_{1} / D_{0}$ | $T_{1} / T_{0}$ |
| 15 | 4 | 218 | 17881 | 17763 | 1.5 | 12595 | 9838 | 0.91 | 0.71 | 0.56 | 0.61 |
| 21 | 5 | 222 | 37044 | 36867 | 10.1 | 25325 | 18824 | 5.2 | 0.69 | 0.52 | . 2 |
| 33 | 6 | 526 | 66679 | 6643 | 2 | 44461 | 31436 | 92 | 0.67 | 0.48 | 0.41 |
| 35 | 6 | 226 | 83216 | 82966 | 282 | 55387 | 38869 | 115 | 0.67 | 0.47 | 0.41 |
| 39 | 6 | 226 | 931 | 928 | 3 | 1941 | 43483 | 129 | 0.67 | 0.47 | 0. |
|  | 6 | 226 | 83790 | 83541 | 285 | 55755 | 39348 | 116 | 0.67 | 0.48 | 0.41 |
| 55 | 6 | 226 | 931 | 92 | 315 | 61899 | 43 | 129 | 0.67 | 7 | 0.41 |
|  | 6 | 526 | 77400 | 77151 | 262 | 51360 | 36346 | 107 | 0.67 | 0.48 | 0.41 |
| 65 | 73 | 330 | 126462 | 126133 | 6814 | 82676 | 56199 | 2430 | 0. | . 45 | 36 |
| 69 | 7 | 230 | 151490 | 151157 | 8121 | 98774 | 66690 | 2866 | 0.66 | 0.45 | 0.36 |
| 77 | 72 | 230 | 1 | 159509 | 8546 | 104285 | 70616 | 3033 | 0.66 | 5 | 36 |
| 85 | 7 | 230 | 152208 | 151875 | 816 | 99407 | 67570 | 2906 | 0.66 | 0.45 | 0.36 |
| 87 | 7 | 230 | 183909 | 183575 | 98 | 120027 | 80999 | 3 | 0.66 | 0.45 | 0.36 |
| 91 | 72 | 230 | 178045 | 177711 | 9537 | 116234 | 78729 | 3369 | 0.66 | 0.45 | 0.36 |
|  | 7 | 230 | 165750 | 165417 | 8857 | 108070 | 73227 | 3150 | 0.66 | 0.45 | 0.36 |
| 95 | 72 | 230 | 193219 | 192885 | 10358 | 125960 | 85061 | 3664 | 0.66 | 0.45 | 0.36 |
| 111 | 7 | 230 | 191313 | 190979 | 10257 | 124959 | 84533 | 3646 | 0.66 | 0.45 | 0.36 |
| 115 | 7 | 230 | 168479 | 168145 | 9048 | 109922 | 74503 | 3188 | 0.66 | 0.45 | 0.36 |
| 119 | 7 | 230 | 188369 | 188035 | 10112 | 122960 | 83264 | 3577 | 0.66 | 0.45 | 0.36 |
| 123 | 72 | 230 | 181029 | 180695 | 9692 | 118337 | 80519 | 3452 | 0.66 | 0.45 | 0.36 |

Table 5: Factorization of $N$ with GT-ADD without and with the greedy method
8. Gidney, C., Ekerå, M.: How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. Quantum 5, 433 (2021). https://doi.org/10.22331/ q-2021-04-15-433
9. Gouzien, E., Sangouard, N.: Factoring 2048-bit RSA integers in 177 days with 13 436 qubits and a multimode memory. Physical Review Letters 127(14), 140503 (2021). https://doi.org/10.1103/PhysRevLett.127.140503
10. IBM: 433-qubits quantum processor, Osprey. https://research.ibm.com/blog/ next-wave-quantum-centric-supercomputing
11. Imamura, S., Yamazaki, M., Honda, T., Kasagi, A., Tabuchi, A., Nakao, H., et al.: mpiqulacs: A distributed quantum computer simulator for A64FX-based cluster systems. arXiv preprint arXiv:2203.16044 (2022). https://doi.org/10.48550/arXiv. 2203.16044
12. Kunihiro, N.: Exact analyses of computational time for factoring in quantum computers. IEICE transactions on fundamentals of electronics, communications and computer sciences 88(1), 105-111 (2005). https://doi.org/10.1093/ietfec/e88-a.1. 105
13. Lanyon, B.P., Weinhold, T.J., Langford, N.K., Barbieri, M., James, D.F., Gilchrist, A., et al.: Experimental demonstration of a compiled version of Shor's algorithm with quantum entanglement. Physical Review Letters 99(25), 250505 (2007). https: //doi.org/10.1103/PhysRevLett.99.250505
14. Lu, C.Y., Browne, D.E., Yang, T., Pan, J.W.: Demonstration of a compiled version of Shor's quantum factoring algorithm using photonic qubits. Physical Review Letters 99(25), 250504 (2007). https://doi.org/10.1103/PhysRevLett.99.250504
15. Lucero, E., Barends, R., Chen, Y., Kelly, J., Mariantoni, M., Megrant, A., et al.: Computing prime factors with a Josephson phase qubit quantum processor. Nature Physics 8(10), 719-723 (2012). https://doi.org/10.1038/nphys2385

## 20 J. Yamaguchi et al.



Fig. 8: Average values of the number of gates and the depth of Shor's circuit for $n$-bit integers. The dashed lines represent approximation polynomials.

|  |  |  |  | Average |  | Lowest |  | Highest |  |
| ---: | ---: | ---: | ---: | ---: | ---: | ---: | :---: | :---: | :---: |
| $n$ | gates | depth | gates | depth | gates |  |  |  |  |
| depth |  |  |  |  |  |  |  |  |  |
| 8 | 109654 | 87762.8 | 107372 | 85241 | 111100 | 89648 |  |  |  |
| 9 | 159835.8 | 128291.1 | 158641 | 126288 | 162018 | 130662 |  |  |  |
| 10 | 223161.5 | 179300.9 | 218662 | 173506 | 225187 | 182354 |  |  |  |
| 11 | 299715.2 | 240530.5 | 297074 | 238494 | 302068 | 243438 |  |  |  |
| 12 | 393551.7 | 315677.3 | 387423 | 309689 | 397390 | 320203 |  |  |  |
| 13 | 503860 | 403660.2 | 497511 | 396630 | 508398 | 409619 |  |  |  |
| 14 | 633082.9 | 507259.3 | 627831 | 500757 | 641738 | 517706 |  |  |  |
| 15 | 783469.7 | 627229.6 | 779900 | 622474 | 788900 | 634906 |  |  |  |
| 16 | 958542.6 | 769055.5 | 953155 | 761892 | 967134 | 781641 |  |  |  |
| 17 | 1152195 | 92632.6 | 1146738 | 915406 | 1157066 | 929076 |  |  |  |
| 18 | 1373845.4 | 1100267 | 1366213 | 1087255 | 1384917 | 1115100 |  |  |  |
| 19 | 1628078.6 | 1307037.4 | 1618600 | 1293395 | 1644736 | 1331018 |  |  |  |
| 20 | 1901953.5 | 1525742.6 | 1886368 | 1503536 | 1909138 | 1535936 |  |  |  |
| 21 | 2213048.9 | 1776710.5 | 2203974 | 1764214 | 222441 | 1789865 |  |  |  |
| 22 | 2549491.8 | 2045255.8 | 2532631 | 2023007 | 2562329 | 2062360 |  |  |  |
| 23 | 2919664.5 | 2342540.3 | 2907098 | 2320299 | 2936593 | 2366786 |  |  |  |
| 24 | 3326305.5 | 2669232.6 | 3295857 | 2629337 | 3349921 | 2701801 |  |  |  |

Table 6: Resources of optimized Shor's circuit with R-ADD
16. Martin-Lopez, E., Laing, A., Lawson, T., Alvarez, R., Zhou, X.Q., O'brien, J.L.: Experimental realization of Shor's quantum factoring algorithm using qubit recycling. Nature photonics 6(11), 773-776 (2012). https://doi.org/10.1038/nphoton. 2012.259
17. Politi, A., Matthews, J.C., O'brien, J.L.: Shor's quantum factoring algorithm on a photonic chip. Science 325(5945), 1221-1221 (2009). https://doi.org/10.1126/ science. 1173731
18. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th FOCS. pp. 124-134. IEEE Computer Society Press (Nov 1994). https://doi.org/10.1109/SFCS.1994.365700
19. Suzuki, Y., Kawase, Y., Masumura, Y., Hiraga, Y., Nakadai, M., Chen, J., et al.: Qulacs: a fast and versatile quantum circuit simulator for research purpose. Quantum 5, 559 (2021). https://doi.org/10.22331/q-2021-10-06-559
20. Vedral, V., Barenco, A., Ekert, A.: Quantum networks for elementary arithmetic operations. Physical Review A 54(1), 147 (1996). https://doi.org/10.1103/ PhysRevA.54.147
21. Yan, B., Tan, Z., Wei, S., Jiang, H., Wang, W., Wang, H., et al.: Factoring integers with sublinear resources on a superconducting quantum processor. arXiv preprint arXiv:2212.12372 (2022). https://doi.org/10.48550/arXiv.2212.12372

# Quantum Circuits for High-degree and half-Multiplication For Post-Quantum Analysis 

Rini Wisnu Wardhani ${ }^{1[0000-0003-0565-6458]}$ *, Dedy Septono Catur Putranto ${ }^{2,3[0000-0002-1246-7877]}$, and Howon Kim ${ }^{1[0000-0001-8475-7294]}$<br>${ }^{1}$ School of Computer Science and Engineering, Pusan National University, Busan 609735, South Korea<br>${ }^{2}$ IoT Research Center, Pusan National University, Busan 609735, South Korea<br>${ }^{3}$ Blockchain Platform Research Center, Pusan National University, Busan 609735, South Korea<br>\{rini.wisnu, dedy.septono, howonkim\}@pusan.ac.kr


#### Abstract

Along with the possibility of accelerated polynomial multiplication, the ToomCook $k$-way multiplication technique has drawn significant interest in the field of postquantum cryptography due to its ability to serve as a part of the lattice-based algorithm. In contrast, the growing likelihood of attacks based on multiplication, specifically correlation power analysis attacks, has heightened vulnerability and emphasized the need to examine the feasibility of employing the polynomial multiplication method as a potential alternative in the era of post-quantum. This study examines thoroughly an elaborate mathematical procedure designated as high-degree and half-multiplication, focusing on the design of an efficient multiplication technique. The proposed polynomial multiplication is intended to be enhanced in terms of asymptotic performance analysis and quantum resource utilization. Through the utilization of the Toom-Cook 8.5 -way method, we reach the lowest asymptotic performance and quantum resources usage for multiplication operation in comparison to the existing Toom-Cook-based multiplication designs with $186 n^{\log _{9} 17}-202 n$ Toffoli count  of $n\left(\frac{17}{9}\right)^{\frac{1 \log 17}{\left(2 \log 17 \log _{9}\right)} \log _{9} n}$, or approximately $n^{1.236}$. We further compare its asymptotic performance and quantum resource efficiency to other Toom-Cook-based multiplications to determine its efficacy.


Keywords: High-degree and half-multiplication • Toom-Cook • Post-Quantum Cryptography • Correlation Power Analysis • Quantum

## 1 Introduction

The Toom-Cook, a method based on [34], [11], is widely acknowledged as an effective approach for solving large number multiplication algorithms. The approach being referred to is a mathematical method employed for the efficient multiplication of polynomials. This method involves breaking down the multiplication process into smaller multiplications (sub-multiplications) and

[^22]additions, thereby minimizing the overall computing complexity. The use of this technique is prevalent throughout diverse domains, including computer algebra systems, cryptography, and signal processing, with the aim of enhancing the efficiency of polynomial multiplication processes.

Besides the number theoretic transform (NTT)-based polynomial multiplication, the Toom-Cook-based or Karatsuba-based polynomial multiplication algorithms have experienced a resurgence in popularity after the commencement of the National Institute of Standards and Technology's (NIST) post-quantum standardization program [26], [23]. Several studies (i.e., [14], [23], and [26]) have put forth a new approach to Toom-Cook multiplication, taking into account the NIST adoption of the module learning with errors (MLWE) algorithm, which forms the basis of many lattice-based cryptography schemes, as the forthcoming standard.

In terms of Toom-Cook multiplication implementation, to optimize performance and reduce implementation costs, Putranto et al. [32] propose employing a Toom-Cook-based multiplier based on several Toom-Cook calculation strategies, including [7], [35], [13], [21]. The analysis of the asymptotic performance of multiplication algorithms and the corresponding costs associated with their quantum implementation offers effectiveness in multiplication operations and valuable perspectives on the importance of multiplication algorithms within the realm of post-quantum cryptography (PQC) and mitigating the risk of side-channel attacks (SCA). Meanwhile, Mera et al. [26], provide a proposition consisting of two innovative strategies aimed at enhancing the efficiency of polynomial multiplications based on the Toom-Cook algorithm. These techniques are then implemented within the Saber post-quantum key encapsulation mechanism.

Recently, the present study [23] investigates the vulnerabilities of the Toom-Cook algorithm in the reference implementation of the Saber cryptographic scheme. It introduces a novel approach by conducting a single-trace attack on Toom-Cook, utilizing the soft-analytical side-channel attack technique. In accordance with this, Mujdei et al. [28] undertook a comparative examination of the complexity associated with attacking various multiplication schemes, multiplication algorithms, and parameter selections. This study utilized the correlation power analysis (CPA) technique, which was first introduced by Brier et al. in their influential paper released in 2004 [10], to prove the existing Toom-Cook vulnerability, particularly the Toom-Cook 4-way PQC algorithm, against the attacks.

The examination of the feasibility of polynomial multiplication as a prospective alternative within the context of PQC holds significant importance. Lattice-based cryptographic systems commonly employ either the NTT with time complexity of $(\mathcal{O}(n \log n))$ [30] or the Toom-Cook/Karatsuba algorithm with time complexity of $\left(\mathcal{O}\left(n^{1+\epsilon}\right)\right.$, where $\left.0<\epsilon<1\right)$, [34], [11], [17], to achieve efficient polynomial multiplication involving $n$ coefficients [28]. In this paper, we will explore the utilization of a new and advantageous multiplication operation derived from Toom's approach, considering that Toom-Cook-based multiplication, especially degrees up to 4 , is part of the lattice-based postquantum algorithm approach, which is also associated with attacks. Further, the proposed multiplication is intended to be integrated into a quantum cryptanalysis circuit with the aim of facilitating an evaluation of post-quantum security.

In this study, we refer to Bodrato's research on high-degree Toom'n'half balanced and unbalanced multiplication [8] to elucidate the functioning of Toom's method for polynomials. To the best of our knowledge, this study is the first to utilize high-degree and half-multiplication compounds in quantum circuits, specifically Toom-Cook-based multiplication exceeding 8 degrees. The primary objective in the design of high-degree and half-multiplication quantum circuits is to reach lower asymptotic performance analyses and minimize the utilization of quantum resources during the execution of multiplication operations. The contributions of this paper can be succinctly summarized as follows:

1. We elaborate a comprehensive analysis of multiplication strategies (i.e., [35], [22], and [32]), with a specific emphasis on the high-degree and half-multiplication technique, the Toom-Cook 8.5way method. Referring to [8], we conduct computation steps like splitting, evaluation, recursive multiplication, interpolation, and recomposition in a certain order, to reach the goal of yielding the best asymptotic performance analysis and the lowest amount of quantum resource use.
2. We design the Toom-Cook 8.5 -way multiplier in a quantum environment, yielding the lowest asymptotic performance analysis for the multiplier and the minimum quantum resource utilization with qubit count $n\left(\frac{17}{9}\right)^{\frac{\log 17}{(2 \log (7)-\log 9)} \log _{9} n} \approx n^{1.236}, 186 n^{\log _{9} 17}-202 n$ Toffoli count, and $n\left(\frac{17}{9}\right)^{1-\frac{\log 17}{(2 \log 177 \log 9)} \log _{9} n} \approx$ $n^{1.053}$ Toffoli depth.
3. We then investigate the asymptotic performance and quantum resource use of various multiplication algorithms, namely the naïve schoolbook method, the Karatsuba algorithm, and existing Toom-Cook-based multiplication up to 8.5 degrees. Additionally, we provide a thorough analysis and evaluation of various factors, including qubit count, Toffoli count, and Toffoli depth, for the purpose of assessing the space-time complexity and drawing up a comprehensive comparison metric to the multiplication operation.

The organization of the paper is as follows: Section 1 provides an overview of the background insights relevant to our work. Section 2 provides a brief overview of high-degree and half-multiplication, particularly in the context of Toom-Cook-based multiplication. Section 3 outlines a detailed procedure for designing the proposed high-degree and half-multiplication, the Toom-Cook 8.5 -way. In Section 4 , we provide a concise insight into the utilization and underlying principles of multiplicationbased attacks with CPA and address multiplication usage in cryptanalysis circuits that led to a post-quantum security evaluation. In Section 5, we analyze and compare the computational complexity in terms of space and time for designs involving proposed multiplication. Future work discussion and conclusions are formulated in Section 6 and Section 7.

## 2 High-degree and half-Multiplication

The Schoolbook Multiplication algorithm, which has a time complexity of $\mathcal{O}\left(n^{2}\right)$, is considered the most basic and straightforward approach for multiplying polynomials of degree $n$, which is equivalent to a variant of the Toom-Cook 1-way algorithm. Meanwhile, the Karatsuba algorithm can be considered a variant of the Toom-Cook 2-way algorithm, in which the original number is divided into two smaller sub-numbers. The reduction of four multiplications to three results in the Karatsuba method yield efficiency compared to naive with a complexity value of $\left(n^{\log (3) / \log (2)}\right) \equiv \mathcal{O}\left(n^{1.58}\right)$.

The Toom-Cook algorithm, specifically the Toom-Cook $k$-way algorithm for multiplication, is a divide-and-conquer approach that bears resemblance to Karatsuba multiplication. However, unlike Karatsuba multiplication which divides each polynomial into two equal parts during each recursive step, the Toom-Cook $k$-way multiplication divides two large integers $f$ and $g$ into $k$ smaller parts, each with a length of $l$. In general, the time complexity of the Toom-Cook $k$-way algorithm can be expressed as $\mathcal{O}\left(c(k) n^{e}\right)$, where $e$ is calculated as the logarithm of $(2 k-1)$ divided by the logarithm of $k$. The term $n^{e}$ represents the time spent on sub-multiplications, while $c$ denotes the time spent on additions and multiplication by small constants.

The computational procedures encompass many steps such as splitting, evaluation, recursive multiplication, interpolation, and recomposition, which have already received extensive study in other works ( $[8,13,21,32,35])$. This study concentrates its attention on effective multiplication,
specifically exploring its complexity before delving into the realm of quantum circuits for high-degree and half-multiplication in quantum architecture.

In the first step in Toom's splitting step, in order to divide a given quantity into $k$ segments using Toom's $k$-way algorithm, it is necessary to choose a base $B=b^{i}$ that satisfies the condition where the number of integer digits both $m$ and $n$ when expressed in base $B$ does not exceed $k$. A commonly selected option for the variable $i$ is provided by Equation 1, then, the variables $m$ and $n$ are partitioned into their respective base $B$ digits, denoted as $m_{i}$ and $n_{i}$.

$$
\begin{equation*}
i=\max \left\{\left\lfloor\frac{\left\lceil\log _{b} m\right\rceil}{k_{m}}\right\rfloor,\left\lfloor\frac{\left\lceil\log _{b} n\right\rceil}{k_{n}}\right\rfloor\right\}+1 \tag{1}
\end{equation*}
$$

Subsequently, the aforementioned digits are employed as coefficients in polynomials $p$ and $q$ of degree $(k-1)$, satisfying the condition that $p(B)$ equals $m$ and $q(B)$ equals $n$. The rationale for the defining of these polynomials lies in the fact that by calculating their product, denoted as $r(x)=p(x) q(x)$, the resulting value $r(B)$ will correspond to the multiplication of $m \mathrm{x} n$.

In the case where the multiplicands have different magnitudes, it is advantageous to employ different values of $k$ for $m$ and $n$, denoted as $k_{m}$ and $k_{n}$. An instance in this condition is the highdegree and half-multiplication Toom-Cook $k$-way ; for example (using terminology, high-degree and half-multiplication), Toom-Cook 8.5 -way corresponds to the Toom-Cook algorithm with the specific values of $k_{m}=9$ and $k_{n}=8$. In this particular scenario, the selection of the variable $i$ in the equation $B=b^{i}$ is commonly determined by Equation 1 .

## 3 Quantum Toom-Cook 8.5-way Multiplier Design

Zanoni et al. [35] introduce a conventional computational implementation of a balanced Toom-Cook 8 -way algorithm for the purpose of integer multiplication and squaring. The authors successfully achieved a degree of 7 in their Toom-Cook-based multiplication version. In their comprehensive study, Dutta et al. [13] provide an in-depth elucidation of the Toom-Cook 2.5-way technique employed in the realm of quantum computing. The authors primarily concentrate on the identification of the maximum count of Toffoli gates and qubits attainable by means of a rigorous examination of the recursive tree inherent to the algorithm.

The research undertaken by Larasati et al. [21] shows findings that demonstrate the possibility of the $k$-way Toom-Cook method, which employs higher-order polynomial interpolation, to exhibit lower asymptotic complexity in comparison to alternative approaches such as Toom-Cook 2.5-way. In their study, Larasati et al. [21] expound upon the Toom-Cook 3-way algorithm by incorporating the division gate. They augment their analysis by drawing upon the research conducted by Bodrato et al. [7], resulting in a singular instance of accurate division by three circuits in every iteration. Moreover, the cost related to the remaining division was reduced by the usage of the circuit's unique properties. The aforementioned accomplishment was attained through the use of a circuit that employs a constant multiplication by reciprocal technique, complemented with the requisite swap operations [21].

Referring to [32], the following part provides a detailed description of the sequential procedure for implementing our quantum Toom-Cook 8.5-way Multiplication algorithm, while also highlighting the distinctions between this approach and the Toom-Cook 8-way multiplication method for the purpose of clarification. The comparison between the recursion tree structures of Toom-Cook 8way and Toom-Cook 8.5-way is depicted in Figure 1. In the present context, Figure 2 draws a


Fig. 1: The Toom-Cook 8-way and 8.5-way Multiplication Recursion Tree Structure, where $T$ represents the Toom-Cook $k$-way Multiplication and $n$ and $N$ represent the bit length for each level and the overall depth of the tree, respectively.
comparative analysis of quantum circuits pertaining to the multiplications of Toom-Cook 8-way and Toom-Cook 8.5-way.

### 3.1 Computation Steps

Focusing on the Toom-Cook 8.5-way strategy design, this work explains and undertakes a thorough investigation of high-degree and half-multiplication methods based on the Toom-Cook algorithm within the context of polynomial multiplication. We incorporate several prior research findings, including [32], and [8]. The processes of computation include splitting, evaluation, recursive multiplication, interpolation, and recomposition, as discussed in previous studies [35], [8], [22], [32]. To offer a succinct explanation of the approach, the quantities to be multiplied, referred to as the input operands, are represented by the variables $x$ and $y$. The variable $x$ is used to represent the complete numerical input. The subscripts $x_{0}, x_{1}, x_{-1}, x_{-2}, \ldots$ are used to signify the individual components of the input. On the other hand, the notations $x(0), x(1), x(-1), x(-2), \ldots$ are employed to indicate the results obtained by evaluating the variable $x$ at certain places.

Splitting. As shown by Equations 2 and 3, the specified inputs, denoted as $x$ and $y$, are divided into eight smaller pieces of length $\frac{n}{8}$. The radix $j$ in the equations can be determined in advance through the calculation of Equation 4.

$$
\begin{gather*}
x=x_{7} s^{7 j}+x_{6} s^{6 j}+x_{5} s^{5 j}+x_{4} s^{4 j}+x_{3} s^{3 j}+x_{2} s^{2 j}+x_{1} s^{j}+x_{0}  \tag{2}\\
y=y_{8} s^{8 j}+y_{7} s^{7 j}+y_{6} s^{6 j}+y_{5} s^{5 j}+y_{4} s^{4 j}+y_{3} s^{3 j}+y_{2} s^{2 j}+y_{1} s^{j}+y_{0}  \tag{3}\\
\left.j=\max \left\{\left\lfloor\frac{\left\lceil\log _{2} x\right\rceil}{9}\right\rfloor, \frac{\left\lceil\log _{2} y\right\rceil}{8}\right\rfloor\right\}+1 \tag{4}
\end{gather*}
$$



Toom Cook 8


Toom Cook 8.5

Fig. 2: Quantum Circuits Comparison for the Toom-Cook 8-way and Toom-Cook 8.5-way Multiplication Algorithms. The function block boxes serve as representations of the individual steps involved in constructing the Toom-Cook quantum circuit. The quantum circuit utilized in the multiplication algorithm uses red triangles to denote the input and output of each respective operation within the function blocks. A notation symbol is employed to denote the quantum state of the input, with each line representing a required register in the quantum circuit. The presence of triangles positioned on the left side of a block serves to highlight the location of its input entry point. The output location on the right side is symbolized by triangles. To maintain simplicity, the ancilla registers are omitted from the display.

```
\(F=x_{0} y_{0}\)
\(G=\left(x_{7}+x_{6}+x_{5}+x_{4}+x_{3}+x_{2}+x_{1}+x_{0}\right)\left(y_{8}+y_{7}+y_{6}+y_{5}+y_{4}+y_{3}+y_{2}+y_{1}+y_{0}\right)\)
\(H=\left(-x_{7}+x_{6}-x_{5}+x_{4}-x_{3}+x_{2}-x_{1}+x_{0}\right)\left(y_{8}+-y_{7}+y_{6}-y_{5}+y_{4}-y_{3}+y_{2}-y_{1}+y_{0}\right)\)
\(I=\left(128 x_{7}+64 x_{6}+32 x_{5}+16 x_{4}+8 x_{3}+4 x_{2}+2 x_{1}+x_{0}\right)\left(256 y_{8}+128 y_{7}+64 y_{6}+32 y_{5}+16 y_{4}+8 y_{3}+4 y_{2}+2 y_{1}+y_{0}\right)\)
\(J=\left(-128 x_{7}+64 x_{6}-32 x_{5}+16 x_{4}-8 x_{3}+4 x_{2}-2 x_{1}+x_{0}\right)\left(256 y_{8}+-128 y_{7}+64 y_{6}-32 y_{5}+16 y_{4}-8 y_{3}+4 y_{2}-2 y_{1}+y_{0}\right)\)
\(K=\left(16384 x_{7}+4096 x_{6}+1024 x_{5}+256 x_{4}+64 x_{3}+16 x_{2}+4 x_{1}+x_{0}\right)\)
\(\left(65536 y_{8}+16384 y_{7}+4096 y_{6}+1024 y_{5}+256 y_{4}+64 y_{3}+16 y_{2}+4 y_{1}+x_{0}\right)\)
\(L=\left(-16384 x_{7}+4096 x_{6}-1024 x_{5}+256 x_{4}-64 x_{3}+16 x_{2}-4 x_{1}+x_{0}\right)\)
\(\left(65536 y_{8}-16384 y_{7}+4096 y_{6}-1024 y_{5}+256 y_{4}-64 y_{3}+16 y_{2}-4 y_{1}+x_{0}\right)\)
\(M=\left(2097152 x_{7}+262144 x_{6}+32768 x_{5}+4096 x_{4}+512 x_{3}+64 x_{2}+8 x_{1}+x_{0}\right)\)
\(\left(16777216 y_{8}+2097152 y_{7}+262144 y_{6}+32768 y_{5}+4096 y_{4}+512 y_{3}+64 y_{2}+8 y_{1}+y_{0}\right)\)
\(N=\left(-2097152 x_{7}+262144 x_{6}-32768 x_{5}+4096 x_{4}-512 x_{3}+64 x_{2}-8 x_{1}+x_{0}\right)\)
\(\left(16777216 y_{8}+-2097152 y_{7}+262144 y_{6}-32768 y_{5}+4096 y_{4}-512 y_{3}+64 y_{2}-8 y_{1}+y_{0}\right)\)
\(O=\left(268435456 x_{7}+16777216 x_{6}+1048576 x_{5}+65536 x_{4}+4096 x_{3}+256 x_{2}+16 x_{1}+x_{0}\right)\)
\(\left(4294967296 y_{8}+268435456 y_{7}+16777216 y_{6}+1048576 y_{5}+65536 y_{4}+4096 y_{3}+256 y_{2}+16 y_{1}+y_{0}\right)\)
\(P=\left(-268435456 x_{7}+16777216 x_{6}-1048576 x_{5}+65536 x_{4}-4096 x_{3}+256 x_{2}-16 x_{1}+x_{0}\right)\)
\(\left(4294967296 y_{8}-268435456 y_{7}+16777216 y_{6}-1048576 y_{5}+65536 y_{4}-4096 y_{3}+256 y_{2}-16 y_{1}+y_{0}\right)\)
\(Q=\left(0.0078125 x_{7}+0.015625 x_{6}+0.03125 x_{5}+0.0625 x_{4}+0.125 x_{3}+0.25 x_{2}+0.5 x_{1}+x_{0}\right)\)
\(\left(0.00390625 y_{8}+0.0078125 y_{7}+0.015625 y_{6}+0.03125 y_{5}+0.0625 y_{4}+0.125 y_{3}+0.25 y_{2}+0.5 y_{1}+y_{0}\right)\)
\(R=\left(-0.0078125 x_{7}+0.015625 x_{6}-0.03125 x_{5}+0.0625 x_{4}-0.125 x_{3}+0.25 x_{2}-0.5 x_{1}+x_{0}\right)\)
\(\left(0.00390625 y_{8}-0.0078125 y_{7}+0.015625 y_{6}-0.03125 y_{5}+0.0625 y_{4}-0.125 y_{3}+0.25 y_{2}-0.5 y_{1}+y_{0}\right)\)
\(S=\left(0.00006103515625 x_{7}+0.000244140625 x_{6}+0.0009765625 x_{5}+0.00390625 x_{4}+0.015625 x_{3}+0.0625 x_{2}+0.25 x_{1}+x_{0}\right)\)
\(\left(0.0000152587890625 y_{8}+0.00006103515625 y_{7}+0.000244140625 y_{6}+0.0009765625 y_{5}+0.00390625 y_{4}+0.015625 y_{3}+0.0625 y_{2}\right.\)
\(\left.+0.25 y_{1}+y_{0}\right)\)
\(T=\left(-0.00006103515625 x_{7}+0.000244140625 x_{6}-0.0009765625 x_{5}+0.00390625 x_{4}-0.015625 x_{3}+0.0625 x_{2}-0.25 x_{1}+x_{0}\right)\)
\(\left(0.0000152587890625 y_{8}-0.00006103515625 y_{7}+0.000244140625 y_{6}-0.0009765625 y_{5}+0.00390625 y_{4}-0.015625 y_{3}+0.0625 y_{2}\right.\)
\(\left.-0.25 y_{1}+y_{0}\right)\)
\(U=\left(0.000000476837158203125 x_{7}+0.000003814697265625 x_{6}+0.000030517578125 x_{5}+0.000244140625 x_{4}+0.001953125 x_{3}\right.\)
\(\left.+0.015625 x_{2}+0.125 x_{1}+x_{0}\right)\left(0.0000000596046447753906 y_{8}+0.000000476837158203125 y_{7}+0.000003814697265625 y_{6}\right.\)
\(\left.+0.000030517578125 y_{5}+0.000244140625 y_{4}+0.001953125 y_{3}+0.015625 y_{2}+0.125 y_{1}+y_{0}\right)\)
\(V=\left(-0.000000476837158203125 x_{7}+0.000003814697265625 x_{6}-0.000030517578125 x_{5}+0.000244140625 x_{4}-0.001953125 x_{3}\right.\)
\(\left.+0.015625 x_{2}-0.125 x_{1}+x_{0}\right)\left(0.0000000596046447753906 y_{8}-0.000000476837158203125 y_{7}+0.000003814697265625 y 6\right.\)
\(\left.-0.000030517578125 y_{5}+0.000244140625 y_{4}-0.001953125 y_{3}+0.015625 y_{2}-0.125 y_{1}+y_{0}\right)\)
```

Evaluation. We employ $x_{1}=0, x_{2}=1, x_{3}=-1, x_{4}=2, x_{5}=-2, x_{6}=4, x_{7}=-4, x_{8}=8$, $x_{9}=-8, x_{10}=16, x_{11}=-16, x_{12}=0.5, x_{13}=-0.5, x_{14}=0.25, x_{15}=-0.25, x_{16}=-0.125$, and
$x_{17}=-0.125$ to obtain $x(0), x(1), x(-1), x(2), x(-2), x(4), x(-4), x(8), x(-8), x(16), x(-16), x(0.5)$, $x(-0.5), x(0.25), x(-0.25), x(0.125)$ and $x(-0.125)$ for the evaluating points $x$ and $y$, each of the 17 predefined evaluation points. Figure 3 and Figure 4 illustrate the evaluation points x and y for the evaluation stage in the Toom-Cook 8.5 -way multiplications design. The exact equation for the evaluation points $x(0), x(1), x(-1), x(2), x(-2), x(4), x(-4), x(8), x(-8), x(16), x(-16), x(0.5), x(-0.5)$, $x(0.25), x(-0.25), x(0.125)$ and $x(-0.125)$ is not included in this work. However, it can be inferred from the evaluation multiplication equation, Equation 5.

Recursive Multiplication. A single iteration of non-recursive point-wise multiplication for ToomCook 8.5 -way multiplication utilizes a total of 17 multiplications, each with smaller bit lengths. To multiply each component of $x(0), x(1), x(-1), x(2), x(-2), x(4), x(-4), x(8), x(-8), x(16), x(-16)$, $x(0.5), x(-0.5), x(0.25), x(-0.25), x(0.125)$ and $x(-0.125)$, the result is expressed in Equation 5 , denoted as $F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U$, and $V$, respectively.

Interpolation. The process of interpolation can be represented mathematically using a matrix, which is the opposite process of multiplying a point, as demonstrated in Equation 6. It needs to be noticed that, in the aforementioned procedure, an inverse matrix derived from the sub-multiplication of coefficients ( $k_{0} \ldots k_{16}$ ) in Equation 5 is employed. To facilitate comprehension, the inverse matrix is represented as described in Equation 6.

Recomposition The recomposition from the interpolation result is indicated as $V V, U U, T T, S S$, $R R, Q Q, P P, O O, N N, M M, L L, K K, J J, I I, H H, G G$, and $F F$ in Equation 7 below. The final product of Toom-Cook 8.5 -way multiplication is the $x y$ Equation.


$$
\begin{align*}
& x y=F F 2^{16 j}+G G 2^{15 j}+H H 2^{14 j}+I I 2^{13 j}+J J 2^{12 j}+K K 2^{11 j}+L L 2^{10 j}+M M 2^{9 j} \\
& +N N 2^{8 j}+O O 2^{7 j}+P P 2^{6 j}+Q Q 2^{5 j}+R R 2^{4 j}+S S 2^{3 j}+T T 2^{2 j}+U U 2^{j}+V V \tag{7}
\end{align*}
$$

Fig. 3: Evaluation point x


Session 7-4 $\quad$ The $26^{\text {th }}$ Annual International Conference on Information Security and Cryptology

10 Rini Wisnu Wardhani, Dedy Septono Catur Putranto, and Howon Kim


## 4 Toom-Cook-Based Polynomial Multiplication in the Post-Quantum

Numerous investigations have been conducted pertaining to the enhancement of public-key cryptosystems, aiming to protect against potential attacks deriving from both classical and quantum computing paradigms. The period characterized by the need for quantum-resistant encryption is commonly denoted as the PQC era, as elucidated in [1]. According to the NIST PQC standardization process, the two main algorithms that are suggested for a range of applications, including digital signatures, are Crystals-Kyber [9] for public-key setup and Crystals-Dilithium [25] Latticebased encryption is expected to exhibit optimal efficiency and resilience against quantum attacks, rendering it a feasible solution within the domain of PQC and appears to be the most rapid implementation as in [27] [24] [6] [5]. Dilithium, Falcon, FrodoKEM, Kyber, NTRU, NTRU Prime, and Saber are seven of the fifteen candidates in the NIST third round that use lattice-based cryptography [1]. In this subsection, we present a brief example of the usage and implementation of Toom-Cook-based multiplication in the Saber and Kyber PQC algorithm, as well as the potential vulnerability that arises from the utilization of lower-degree multiplication.

The primary focus of public key cryptography (PKC) implementation is on compactness, power efficiency, and energy consumption, with a secondary consideration given to throughput or delay [14]. This is due to its main purpose of generating shared secret keys. While the majority of other research concentrates on optimizing NTT-based multiplications, [14] research optimizes a Toom-Cook-based multiplier to an exceptional degree. A memory-efficient striding Toom-Cook with delayed interpolation yields a highly compact, low-power implementation that allows for a very regular memory access scheme. They demonstrate the multiplier's effectiveness and integrate it into one of the four NIST finalists, the Saber post-quantum accelerator. The results of the runtime analysis for a post-quantum lattice-based cryptographic algorithm, specifically a key encapsulation mechanism, are displayed in Figure 5. In this figure, our focus is solely on the Kyber algorithm. The analysis is conducted by comparing the algorithm's runtime behavior and memory consumption statistics, as documented in the work by Mujdei et al. [28].

Polynomial multiplications, such as Toom-Cook and NTT, play a crucial role in lattice-based post-quantum encryption by serving as the essential constituents. Lattice-based cryptographic systems commonly employ either the NTT with time complexity of $(\mathcal{O}(n \log n))$ [30] or the ToomCook/Karatsuba algorithm with time complexity of $\left(\mathcal{O}\left(n^{1+\epsilon}\right)\right.$, where $\left.0<\epsilon<1\right)$, [34] [11] [17], to achieve efficient polynomial multiplication involving $n$ coefficients [28]. These multiplications facilitate the division of the resultant sub-polynomial, as highlighted in [28]. The Saber algorithm employs an additional division of the resultant sub-polynomials into two Karatsuba layers, followed by the execution of a 16 -coefficient schoolbook operation [28]. Figure 6 displays an image that portrays an occurrence of Toom-Cook-based multiplication executed within the Saber structures. We redraw from the work of Mera et al. [26] to demonstrate the application of the Toom-Cook 4 -way method in the implementation of the Saber post-quantum cryptography algorithm.

The exploitation of side-channel information, such as power consumption, electromagnetic radiation, and execution time, has been shown to be a method for gaining unauthorized access to sensitive data [19]. CPA is widely recognized as a very effective technique that leverages the correlation between a device's power consumption and the data it is processing. This approach exploits power fluctuations that are caused by mathematical processes such as multiplication. Hence, the evaluation of potential risks associated with multiplication exploitation in side-channel analysis attacks, particularly when utilizing the CPA approach, is crucial during the construction of cryp-


Fig. 5: Runtime analysis of Open Quantum Safe Lattice-based Cryptographic algorithms (Key Encapsulation Mechanisms)


Fig. 6: The Toom-Cook 4-way and Karatsuba Multiplication used in Saber Post-Quantum Cryptography Algorithm
tographic algorithms. This concern arises due to the frequent use of arithmetic multiplication as a sub-operation multiplier in real implementations.

The architectural design of all NTRU versions exhibits a common structure, characterized by the presence of four Karatsuba layers, with the exception of ntruhps2048509, which features three layers [28]. Further, variations in the schoolbook thresholds are observed [28]. Mujdei et al. conducted an experimental analysis to investigate the potential occurrence of CPA peaks when employing the schoolbook sub-operation in the processing of 3 -way and 4 -way Toom-Cook within the lattice-based PQC algorithm. The post-quantum algorithm ntruhps 4096821 elaborated in [28], can be subjected to a multiplication-based attack utilizing side-channel measurements. Mujdei et al. study encompasses an examination of the variance plot of 500 instances of schoolbook multiplication, wherein a comprehensive analysis reveals the identification of a total of 72 apparent peaks. These peaks are specifically associated with the targeted algorithm as described in the work by [28].

PQC refers to a collection of cryptographic methods, specifically algorithms developed for the purpose of public key encapsulation, that are widely acknowledged for their ability to withstand possible attacks from quantum computers. The main goal of PQC is to strengthen and optimize mathematical methods and standards in anticipation of the emergence of quantum computing. Proficiency in mathematical approaches is essential for the development of PQC algorithms that can effectively withstand SCA. Furthermore, the utilization of effective mathematical techniques is imperative in the construction of quantum circuits, which can be employed for the creation of cryptanalysis circuits. The primary function of these cryptanalysis circuits is to evaluate the resilience of a method.

Efficient arithmetic operations, particularly multiplication, play a vital role in conducting comprehensive investigations within the domain of quantum-based cryptanalysis. According to Roche [33], Parent et al. [29], Gidney [15], Banegas et al. [3], and Putranto et al. [32], [31], the development of a fundamental arithmetic constructor that demonstrates efficiency in terms of space use and time consumption is crucial for expediting the cryptanalysis process. The primary objective of these investigations is to reduce the complexity that is typically encountered during the execution of quantum cryptanalysis. The efficacy of basic mathematical operations, particularly multiplication, can significantly impact the predictive analysis of the utilization of multiplication inside the lattice-based PQC algorithm, as well as the quantum computer's ability to solve conventional public key cryptography through cryptanalysis, which further leads to post-quantum security evaluation.

## 5 Complexity Analysis of High-degree and half-Multiplication

### 5.1 Toffoli Gate Count

The variable $T_{n}$ is used to represent the cost incurred when performing multiplication on two larger $n$-bit quantities utilizing the Toom-Cook multiplier. Thus, $A_{n}$ denotes the cost associated with the addition or substracting of $n$ bits. To implement a $n$-bit Toom-Cook 8.5 -way multiplication, it is necessary to perform a total of 17 operations involving $\frac{n}{9}$ submultiplications and three types of adders with different lengths. These adders consist of 46 operations for $\frac{n}{9}$-bit adders, 272 operations for $\frac{2 n}{9}$-bit adders. The Toffoli cost of an n-bit Toom-Cook 8.5 -way multiplication can be determined by employing the equation referenced as Equation 8. Furthermore, the cost increases to 9 for recursive implementations, and Equation 10 becomes equivalent when the Toffoli cost of $A_{n}=2 n$ is substituted.

$$
\begin{equation*}
T_{n}=17 T_{\frac{n}{9}}+46 A_{\frac{n}{9}}+272 A_{\frac{n}{9}} \tag{8}
\end{equation*}
$$

$$
\begin{gather*}
T_{n}=17^{\log _{9} n} T_{1}+46\left(A \frac{n}{9}+23 A \frac{n}{81}+\cdots+23^{\log _{9}(n)-1} A_{1}\right) \\
+272\left(A \frac{2 n}{9}+136 A \frac{2 n}{81}+\cdots+95^{\log _{9}(n)-1} A_{2}\right)  \tag{9}\\
T_{n}=17^{\log _{9} n}+\sum_{i=0}^{\log _{9}(n)-1}\left[92 n\left(\frac{17}{9}\right)^{i}\right] \tag{10}
\end{gather*}
$$

By utilizing the geometric series calculation $\sum_{i=0}^{m-1} r^{i}=\frac{1-r^{m}}{1-r}$, it is possible to determine the Toffoli cost of a recursive implementation, as denoted by Equation 11. The result obtained from Equation 11 does not consider the typical uncomputation procedure carried out in a quantum environment. The strategy mentioned in this study is also discussed in previous research conducted by [29], [13], [21], and Putranto et al [32]. Equation 12 in this study incorporates the concept of uncomputed process to prevent a significant increase in the previously determined cost. It is important to acknowledge that the definition of "clean cost" used in the subsequent equation aligns with Larasati et al.'s [21]and Putranto et al.'s [32] definitions.

$$
\begin{gather*}
T_{n}=17^{\log _{9} n}+92 n\left(\frac{1-\left(\frac{17}{9}\right)^{\log _{9} n}}{1-\left(\frac{17}{9}\right)}\right) \\
=n^{\log _{9} 17}+92 n\left(\frac{1-n^{\left.\log _{9} \frac{17}{9}\right)}}{1-\left(\frac{17}{9}\right)}\right)  \tag{11}\\
=93 n^{\log _{9} 17}-101 n \\
T_{n(\text { clean })}=186 n^{\log _{9} 17}-202 n \tag{12}
\end{gather*}
$$

### 5.2 Space-Time Complexity Analysis

Bennett in [4] introduced the technique for measuring asymptotic performance improvements in the context of space consumption in the context of space-time complexity analysis. This technique is utilized extensively in reversible computing, which makes time and space complexity analysis possible and enables time-efficient finite-space computing [20]. This method will allow us to evaluate the difference in the cost of the successfully optimized multiplication and compare it to the results of previous studies. We determined the optimal cost of multiplication by following the procedures outlined in [29], [13], [21], and [32].

In the Toom-Cook 8.5 -way algorithm, 17 simultaneous multiplications were done in a recursive way to make a quinary eight structure. There are $17^{l}$ nodes of size $9^{-l} n$ for an input of size $n$ at level $l$, and this input has a total circuit cost of $n\left(\frac{15}{9}\right)^{l}$. Equations $13-15$ depict the total price of the quinary tree. For determining the optimal tree height $k$ for optimal performance, use Equation 15.

$$
\begin{equation*}
n \sum_{i=0}^{N}\left(\frac{17}{9}\right)^{i}, \quad N=\log _{9} n \tag{13}
\end{equation*}
$$

$$
\begin{equation*}
n \sum_{i=0}^{N-k-1}\left(\frac{17}{9}\right)^{i}=\frac{1}{9^{N-k}} \sum_{i=0}^{k-1}\left(\frac{17}{9}\right)^{i} \tag{14}
\end{equation*}
$$

In a pattern similar to Equation 12, the identity of the geometric series enables us to locate the boundaries indicated by Equation 15. Thus, the space can be reduced, as shown in qubit count Equation 16. The obtained result from Equation 16, approximately equal to $\mathcal{O}\left(n^{1.245}\right)$, is lower than the initially required space assessed with Equation 17, which is confined to the value $\mathcal{O}\left(n^{\log _{9} 15}\right) \approx \mathcal{O}\left(n^{n^{1.3029}}\right)$.

$$
\left.\left.\begin{array}{c}
k \leq \frac{N}{2-\frac{\log 9}{\log 17}} \approx 0.8167 N \\
Q C=\mathcal{O}\left(n \left(\frac{17}{9}\left(\frac{\log 17}{2 \log 1772 \log 9}\right) \log _{9} n\right.\right. \\
\hline \tag{17}
\end{array}\right)\right) \approx \mathcal{O}\left(n^{1.236}\right), ~ \$ \sum_{k=0}^{\log _{9} n-1}\left(\frac{17}{9}\right)^{k}=n\left(\frac{1-\left(\frac{17}{9}\right)^{\log _{9} n}}{1-\frac{17}{9}}\right)
$$

The Toffoli depth of a circuit is a prevalent way to describe its time complexity [13], [2]. It can be calculated by multiplying the number of subtrees $S_{k}$ at the $k-t h$ level by the corresponding depth $D_{k}$. Consequently, we can express the Toffoli depth $T_{d}$ as in Equation 18.

$$
\begin{align*}
& S_{k}=17^{\left(1-\frac{\log 17}{2 \log _{17-\log 9}}\right) \log _{9} n} \\
& D_{k}=\frac{n}{\left.{ }_{9}^{\left(1-\frac{\log _{1} 17}{2 \log 77 \log _{9}}\right)}\right) \log _{9} n}  \tag{18}\\
& T_{d}=S_{k} D_{k}=n\left(\frac{17}{9}\right)^{\left(1-\frac{\log _{17}}{2 \log 15-\log _{9}}\right) \log _{9} n} \approx n^{1.0530}
\end{align*}
$$

### 5.3 Complexity Analysis Comparison

The naïve multiplication, which is equivalent to the Toom-Cook 1-way, exhibits a time complexity of $\mathcal{O}\left(n^{2}\right)$, where $n$ represents the size of the input. The Toffoli depth of Naive is also of the order $\mathcal{O}(n \log n)$, according to a more in-depth study done in [12]. In the context of asymptotic performance analysis in quantum implementation, it is observed that the schoolbook technique necessitates a qubit count of $\mathcal{O}(n)$, as well as a Toffoli count and depth values of $\mathcal{O}\left(n^{2}\right)$. The costs associated with quantum multiplication are characterized by a qubit count of $(4 n+1)$, a Toffoli depth of $\left(4 n^{2}-4 n+1\right)$, and a Toffoli count of $\left(4 n^{2}-3 n\right)$ [13] [21].

Karatsuba multiplication, a multiplication equivalency with the Toom-Cook 2-way approach, resulted in a qubit count of $\mathcal{O}\left(n^{\log _{2}(3)}\right)$ for both the qubit count and Toffoli count. The improvement study reveals asymptotic values for qubit count $\left(\mathcal{O}\left(n^{1.427}\right)\right)$, Toffoli count $\left(\mathcal{O}\left(n^{\log _{2}(3)}\right)\right)$, and Toffoli depth $\left(\mathcal{O}\left(n^{1.158}\right)\right)$ [29] [13] [21]. Parent et al. [29] determined the values of the qubit count, denoted

Table 1: Asymptotic Performance and Quantum Implementation Cost Multipliers Comparison. In order to provide a comprehensive analysis of the advancements in complexity multiplication research, specifically focusing on the Karatsuba and Toom-Cook-based approaches, we provide our results pertaining to cost evaluation. This evaluation is conducted utilizing the Toffoli count, qubit count, and Toffoli depth as metrics to assess the space-time complexity.

| No | Reference | Multiplication Algorithm | Asymptotic Performance Analysis |  |  | Cost of Quantum Implementation of Multiplication |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  | Qubit Count | Toffoli Count | Toffoli Depth | Qubit Count | Toffoli Count | Toffoli Depth | CNOT |
| 1 | Kepley and Steinwandt (2015, [18]) | Karatsuba | $\mathcal{O}\left(n^{\lg _{2}{ }^{3}}\right)$ | $\mathcal{O}\left(n^{\text {bog }}\right.$ ) | - | - | - | - | $\mathcal{O}\left({ }^{\log _{2} 3}\right)$ |
| 2 | Parent et al. (2017, [29]) | Karatsuba | $\mathcal{O}\left(n^{1.4 Z}\right)$ | $\mathcal{O}\left(n^{\log _{2} 3}\right)$ | $\mathcal{O}\left({ }^{1.158}\right)$ |  | $42{ }^{\text {log }{ }_{2} 3}$ |  | - |
| 3 | Dutta et al. (2018, [13]) | Toom-Cook 2.5-way | $\mathcal{O}\left({ }^{1.404}\right)$ | $\mathcal{O}\left({ }^{\text {logis }}\right.$ ) | $\mathcal{O}\left({ }^{1.143}\right)$ |  | $49 n^{\log _{6} 16}$ |  | - |
| 4 | Larasati et al.(2021, [21]) | Toom-Cook 3-way | $\mathcal{O}\left(n^{135}\right)$ | $O\left(n^{2}\right)$ | $\mathcal{O}\left(n^{1.112}\right)$ |  | $8 n^{2}+66 n^{\log _{5} 5}-72$ |  | - |
| 5 | Van Hoof (2020, [16]) | Karatsuba | $3 n$ | $\mathcal{O}\left(n^{\operatorname{tog}_{8} 3}\right)$ | - | - | - | - | $\mathcal{O}\left(n^{2}\right)$ |
| 6 | Putranto et al. (2023), [31]) | Karatsuba | $3 n$ | $\mathcal{O}\left(n^{\operatorname{tog} 3}\right)$ | - | - | - | - | $\mathcal{O}\left(n^{\text {log } 3}\right)$ |
| 7 | Putranto et al. (2023, [32]) | Toom Cook 2-way | $\mathcal{O}\left({ }^{1589}\right)$ | $\mathcal{O}\left(n^{\text {bg }_{2}{ }^{3}}\right)$ | $\mathcal{O}\left(n^{1217}\right)$ |  | $34 n^{\log _{23} 3}-32 n$ |  | - |
| 8 | Putranto et al. (2023, [32]) | Toom Cook 4-way | $\mathcal{O}\left({ }^{1313}\right)$ | $\mathcal{O}\left(n^{\text {bos }}{ }_{4}{ }^{7}\right)$ | $\mathcal{O}\left(n^{109}\right)$ |  | $122 \mathrm{n}^{\log _{4} 7}-160 n$ |  | - |
| 9 | Putranto et al. (2023, [32]) | Toom Cook 8-way | $\mathcal{O}\left({ }^{1245}\right)$ | $\mathcal{O}\left(n^{\log _{4} 15}\right)$ | $\mathcal{O}\left(n^{1.056 \%}\right)$ |  | $112 n^{\text {bog }} 15-128 n n$ |  | - |
| 10 | our | Toom-Cook 8.5-way | $\mathcal{O}\left(n^{1236}\right)$ | $\mathcal{O}\left(n^{\log _{g} 17}\right)$ | $\mathcal{O}\left(n^{1.053}\right)$ |  | $186 n^{\log _{9} 17}-202 n$ |  | - |

as $n^{1.427}$, the Toffoli count, denoted as $\mathcal{O}\left(n^{\log _{2} 3}\right)$, and the Toffoli depth, denoted as $n^{1.158}$ for Karatsuba. Recently, the Karatsuba variant proposed by Putranto et al. [31] demonstrates a reduction in CNOT usage, changing the $\mathcal{O}\left(n^{2}\right)$ CNOT in the prior work to $\mathcal{O}\left(n^{\log _{2}(3)}\right)$.

According to Dutta et al. [13], the Toom-Cook 2.5-way algorithm offers a potential approach for reducing the cost of developing quantum systems by achieving the qubit count ( $n^{1.404}$ ), Toffoli count $\left(49 n^{\log _{6} 16}\right)$, and Toffoli depth $\left(n^{1.143}\right)$. Later, Larasati et al. [21] present a comprehensive examination of the asymptotic performance metrics for qubit count, Toffoli count, and Toffoli depth. They report an estimated value of $n^{1.353}$ for the qubit count, $\mathcal{O}\left(n^{2}\right)$ for the Toffoli count, and $n^{1.112}$ for the Toffoli depth.

Recently, from Putranto et al. [32] elaboration, they exhibit a better asymptotic performance analysis in terms of qubit count for the Toom-Cook 8 -way approach. Specifically, it is approximated by qubit count with $n\left(\frac{15}{8}\right)^{\frac{\log 15}{\left(2 \log 15-\log _{88}\right.} \log _{8} n}$, which is of the order $\mathcal{O}\left(n^{1.245}\right)$. In the context of Toffoli depth, which is relevant to efficient computation, the Toom-Cook 8 -way design results in a lower bound on logical depth of $\mathcal{O}\left(n^{1.0569}\right)$ and a Toffoli count of $\mathcal{O}\left(n^{\log _{8} 15}\right)$.

In the present study, as presented in Table 1, a comparative analysis of various multiplication methods reveals that the Toom-Cook high-degree and half-multiplier, established in this research, demonstrates the lowest desired asymptotic performance in terms of qubit count, Toffoli count, and Toffoli depth when compared to other approaches. In terms of cost, the proposed multiplication in quantum implementation demonstrates lower quantum resources when compared to the alternative Toom-Cook strategy. The high-degree and half-multiplication, specifically the Toom-Cook 8.5 -way approach, involves a qubit count of $\mathcal{O}\left(n^{1.236}\right)$, a logical Toffoli depth of $n\left(\frac{17}{9}\right)^{1-\frac{\log 17}{\left(2 \log 17 \log _{9}\right)} \log _{9} n} \approx n^{1.053}$, and a Toffoli count of $186 n^{\log _{9} 17}-202 n$.

## 6 Discussion

Empirical research has provided evidence indicating that while higher-order procedures may exhibit superior efficiency, the incorporation of the division operation, a crucial component of the
$k$-way Toom-Cook method, can provide difficulties in terms of identifying an effective strategy. In the current research, as shown in Table 1, using the Toom-Cook-8.5 approach and yielding complexity analysis $\left(\mathcal{O}\left(n^{1.236}\right)\right.$ qubit Count, $\mathcal{O}\left(n^{\log _{9}{ }^{17}}\right)$ Toffoli Count, and Toffoli Depth of $\left.\mathcal{O}\left(n^{1.053}\right)\right)$, we established the optimal utilization of resources for multiplication operations. Nevertheless, the design multiplication was not incorporated into the PQC algorithm, and the notable cryptanalysis using the Shor algorithm technique was also not performed. In later stages, it is imperative to also enhance the implementation of a higher degree in the PQC algorithm and provide a more comprehensive examination of multiplication-based attacks employing SCA, or correlation power analysis, methodologies.

Further, it should be noted that the efficiency of the recently developed Toom-Cook method exceeds that of the currently employed Toom-Cook-based multiplication techniques, Karatsuba, and naive schoolbooks. This demonstrates a higher level of efficiency in comparison to existing multipliers based on the Toom-Cook method currently utilized as part of the lattice-based algorithm, the Toom-Cook 4 -way approach. In this work, the multiplication is also designed in a quantum environment, facilitating its integration into quantum circuits for cryptanalysis (e.g., [3], [31]). This integration will thereafter enable the evaluation of security in the post-quantum era.

## 7 Conclusions

The present study undertook a thorough examination of high-degree and half-multiplication, focusing particularly on the Toom-Cook 8.5 -way algorithm. The study demonstrated the achievement of the lowest or most optimal multiplication, which is distinguished by its lower asymptotic performance and fewer demands on quantum resources compared to other multiplications. The proposed multiplication was subjected to asymptotic performance analysis, resulting in a qubit count of $n\left(\frac{17}{9}\right)^{\frac{\log 17}{(2 \log 717-\log 9)} \log _{9} n} \approx n^{1.236}$, approximately $\mathcal{O}\left(n^{1.236}\right)$. Additionaly, the Toom-Cook 8.5 -way has a Toffoli count of $186 n^{\log _{9} 17}-202 n$ and a Toffoli depth of $n\left(\frac{17}{9}\right)^{1-\frac{\log 17}{\left(2 \log 77 \log _{9)}\right.} \log _{9} n} \approx n^{1.053}$ for multiplication.

The alternative methods that have been proposed have the potential to reduce the computational resources needed and can result in efficient multiplication with high degrees of multiplication. As part of planned future research, the suggested multiplication operation could be used as an alternative to constructing lattice-based post-quantum algorithms while lowering the risks of attacks that use multiplication. Furthermore, the multiplication technique is intended to be incorporated into a quantum cryptanalysis circuit in order to enhance the efficiency of evaluating post-quantum security.

## References

1. Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Miller, C., Moody, D., Peralta, R., et al.: Status report on the third round of the nist post-quantum cryptography standardization process. US Department of Commerce, NIST (2022)
2. Amy, M.: Algorithms for the optimization of quantum circuits. Master's thesis, University of Waterloo (2013)
3. Banegas, G., Bernstein, D.J., van Hoof, I., Lange, T.: Concrete quantum cryptanalysis of binary elliptic curves. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 451-472 (2021)
4. Bennett, C.H.: Time/space trade-offs for reversible computation. SIAM Journal on Computing 18(4), 766-776 (1989)
5. Bisheh-Niasar, M., Azarderakhsh, R., Mozaffari-Kermani, M.: High-speed ntt-based polynomial multiplication accelerator for post-quantum cryptography. In: 2021 IEEE 28th Symposium on Computer Arithmetic (ARITH). pp. 94-101. IEEE (2021)
6. Bisheh-Niasar, M., Azarderakhsh, R., Mozaffari-Kermani, M.: Instruction-set accelerated implementation of crystals-kyber. IEEE Transactions on Circuits and Systems I: Regular Papers 68(11), 4648-4659 (2021)
7. Bodrato, M.: Towards optimal toom-cook multiplication for univariate and multivariate polynomials in characteristic 2 and 0. In: International Workshop on the Arithmetic of Finite Fields. pp. 116-133. Springer (2007)
8. Bodrato, M.: High degree toom'n'half for balanced and unbalanced multiplication. In: 2011 IEEE 20th Symposium on Computer Arithmetic. pp. 15-22. IEEE (2011)
9. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-kyber: a cca-secure module-lattice-based kem. In: 2018 IEEE European Symposium on Security and Privacy (EuroS\&P). pp. 353-367. IEEE (2018)
10. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: International workshop on cryptographic hardware and embedded systems. pp. 16-29. Springer (2004)
11. Cook, S.A., Aanderaa, S.O.: On the minimum computation time of functions. Transactions of the American Mathematical Society 142, 291-314 (1969)
12. Draper, T.G., Kutin, S.A., Rains, E.M., Svore, K.M.: A logarithmic-depth quantum carry-lookahead adder. Quantum Information and Computation 6(4) (2006)
13. Dutta, S., Bhattacharjee, D., Chattopadhyay, A.: Quantum circuits for toom-cook multiplication. Physical Review A 98(1), 012311 (2018)
14. Ghosh, A., Mera, J.M.B., Karmakar, A., Das, D., Ghosh, S., Verbauwhede, I., Sen, S.: A 334 microwatt 0.158 mm 2 asic for post-quantum key-encapsulation mechanism saber with low-latency striding toomcook multiplication authors version. arXiv preprint arXiv:2305.10368 (2023)
15. Gidney, C.: Asymptotically efficient quantum karatsuba multiplication. arXiv preprint arXiv:1904.07356 (2019)
16. van Hoof, I.: Space-efficient quantum multiplication polynomials for binary finite fields with sub-quadratic toffoli gate count. Quantum Inf.Comput. 20(9\&10), 721-735 (2020). https://doi.org/10.26421/QIC20.9-10-1, https://doi.org/10.26421/QIC20.9-10-1
17. Karatsuba, A.A., Ofman, Y.P.: Multiplication of many-digital numbers by automatic computers. In: Doklady Akademii Nauk. vol. 145, pp. 293-294. Russian Academy of Sciences (1962)
18. Kepley, S., Steinwandt, R.: Quantum circuits for f2n-multiplication with subquadratic gate count. Quantum Information Processing 14(7), 2373-2386 (2015)
19. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Annual international cryptology conference. pp. 388-397. Springer (1999)
20. Král'ovič, R.: Time and space complexity of reversible pebbling. In: International Conference on Current Trends in Theory and Practice of Computer Science. pp. 292-303. Springer (2001)
21. Larasati, H.T., Awaludin, A.M., Ji, J., Kim, H.: Quantum circuit design of toom 3-way multiplication. Applied Sciences 11(9), 3752 (2021)
22. Larasati, H.T., Awaludin, A.M., Ji, J., Kim, H.: Quantum circuit design of toom 3-way multiplication. Applied Sciences 11(9), 3752 (2021)
23. Li, Y., Zhu, J., Huang, Y., Liu, Z., Tang, M.: Single-trace side-channel attacks on the toom-cook: The case study of saber. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 285-310 (2022)
24. Liu, Z., Choo, K.K.R., Grossschadl, J.: Securing edge devices in the post-quantum internet of things using lattice-based cryptography. IEEE Communications Magazine 56(2), 158-162 (2018)
25. Lyubashevsky, V.: Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 598-616. Springer (2009)
26. Mera, J.M.B., Karmakar, A., Verbauwhede, I.: Time-memory trade-off in toom-cook multiplication: an application to module-lattice based cryptography. Cryptology ePrint Archive (2020)
27. Micciancio, D., Regev, O.: Post-quantum cryptography, chapter lattice-based cryptography. Computing 85(1-2), 105-125 (2008)
28. Mujdei, C., Wouters, L., Karmakar, A., Beckers, A., Mera, J.M.B., Verbauwhede, I.: Side-channel analysis of lattice-based post-quantum cryptography: Exploiting polynomial multiplication. ACM Transactions on Embedded Computing Systems (2022)
29. Parent, A., Roetteler, M., Mosca, M.: Improved reversible and quantum circuits for karatsuba-based integer multiplication. In: 12th Conference on the Theory of Quantum Computation, Communication, and Cryptography (TQC 2017). pp. 7:1-7:15. Springer (2017)
30. Pollard, J.M.: The fast fourier transform in a finite field. Mathematics of computation 25(114), 365-374 (1971)
31. Putranto, D.S.C., Wardhani, R.W., Larasati, H.T., Ji, J., Kim, H.: Depth-optimization of quantum cryptanalysis on binary elliptic curves. IEEE Access (2023)
32. Putranto, D.S.C., Wardhani, R.W., Larasati, H.T., Kim, H.: Space and time-efficient quantum multiplier in post quantum cryptography era. IEEE Access 11, 21848-21862 (2023)
33. Roche, D.S.: Space-and time-efficient polynomial multiplication. In: Proceedings of the 2009 international symposium on Symbolic and algebraic computation. pp. 295-302 (2009)
34. Toom, A.L.: The complexity of a scheme of functional elements realizing the multiplication of integers. In: Soviet Mathematics Doklady. vol. 3, pp. 714-716 (1963)
35. Zanoni, A.: Toom-cook 8-way for long integers multiplication. In: 2009 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing. pp. 54-57. IEEE (2009)

# Theoretical and Empirical Analysis of FALCON and SOLMAE using their Python Implementation 

Kwangjo Kim<br>Korea Advanced Institute of Science and Technology(KAIST) and International Research Institute for Cyber Security(IRCS), Korea kkj@kaist.ac.kr


#### Abstract

Since NIST has recently selected FALCON as one of quantum-resistant digital signatures which uses the hash-and-sign paradigm in the style of Gentry-Peikert-Vaikuntanathan framework and instantiated over NTRU lattices, SOLMAE as a variant of FALCON was submitted to KpqC standard competition by taking all the pros of FALCON and Mitaka and reducing their cons as much as possible. In this paper, we suggest the asymptotic computational complexity of FALCON and SOLMAE take $\Theta(n \log n)$ in their KeyGen, Sign and Verif procedures simultaneously, but our computer experiments using their Python implementation exhibit empirically that KeyGen of FALCON-512 takes longer time than that of SOLMAE- 512 by about a second while the other two procedures are running almost the same time. We show a sample execution of FALCON-512 and SOLMAE-512 with their real value are described in detail for the educational purpose to understand FALCON and SOLMAE easily. We also checked the Gaussian randomness of $\mathcal{N}$-Sampler and UnifCrown samplers used in SOLMAE only.


Keywords: Lattice-based cryptography • Hash-and-sign paradigm • NTRU trapdoors • Discrete Gaussian sampling • Python implementation

## 1 Introduction

When Shor [16] has proposed an efficient randomized algorithm on a hypothetical quantum computer in 1999 to integer factorization and discrete logarithm problems in a polynomial time, it was beyond our imagination building for the powerful computing environment at that time. Currently the threat of attacking the current (or classical) secure system by using the quantum computer is expected to be right at our fingertips due to the aggressive road map by IBM quantum computing. We are very concerned about so called Harvest now, decrypt later attack [17] which is a surveillance strategy that relies on the acquisition and long-term storage of currently unreadable encrypted data awaiting possible breakthroughs in decryption technology that would render it readable in the future.

Due to the substantial amount of research on quantum computers, large-scale quantum computers if built, can break many public-key cryptosystems based on the number-theoretic hard problems in use. In 2016, NIST [14] has initiated Post Quantum Cryptography (PQC) project to solicit, evaluate, and standardize one or more quantum-resistant cryptographic algorithms for Key Encapsulation Mechanism(KEM) and Digital Signature(DS) worldwide. After several rounds, NIST has finally selected CRYSTALS-Kyber for KEM and CRYSTALS-Dilithium, FALCON, and SPHINCS+ for DS in 2022.

Influenced by this NIST PQC project, Korean cryptographic society led by KpqC task force [11] has called for soliciting Korean PQC standard candidates by the end of Oct. in 2022. By the due of submission, 7 candidates KEM and 8 candidates DS for KpqC competition were submitted and their details are available at https://kpqc.or.kr/.

SOLMAE which stands for an acronym of quantum-Secure algOrithm for Long-term Message Authentication and Encryption was submitted to KpqC Competition as one of DS candidate algorithms which is a lattice-based signature scheme inspired by several pioneering works based on the hash-then-sign signature paradigm proposed by Gentry, Peikert and Vaikuntanathan [6].

SOLMAE is inspired from FALCON's design. Some of the new theoretical foundations were laid out in the presentation of Mitaka [1] while keeping the security level of FALCON with 5 NIST levels of security I to V. At a high level, SOLMAE removes the inherent technicality of the
sampling procedure, and most of its induced complexity from an implementation standpoint, for free, that is with no loss of efficiency. This theoretical simplicity translates into faster operations while preserving signatures and verification key sizes, on top of allowing for additional features absent from FALCON, such as enjoying cheaper masking and being parallelizable. We need to evaluate this features with our Python implementation which all the readers can easily understand and compare them.

To the best of our knowledge, there is no the open literature to compare FALCON and SOLMAE directly from the point of their asymptotic complexity and performance. In this paper, after giving a brief description from the specification of FALCON and SOLMAE, we discuss their asymptotic computational complexity of KeyGen, Sign and Verif procedures and evaluate their performance empirically using their Python implementation including Gaussian samplers used in SOLMAE.

The organization of this paper is as follows: In Section 2, we define our notations and definition used in this paper. In Sections 3 and 4, we describe how FALCON and SOLMAE work summarized from their specification, respectively. In Section 5, we discuss the asymptotic computational complexity of FALCON and SOLMAE. In Section 6, we analyse the $\mathcal{N}$-Sampler and UnifCrown sampler used in SOLMAE only and verify its function by the experiment. In Section 7, we suggest the practical execution time of KeyGen, Sign and Verif procedures running 3,000 times for FALCON-512 and SOLMAE-512 by their Python implementation. Finally, we will give concluding remarks and challenging issues.

## 2 Notations and Definition

To keep the consistency to understand FALCON and SOLMAE correctly, we will use the following notations and definitions used their specification throughout this paper.

## Matrices, vectors, and scalars

Matrices will usually be in bold uppercase (e.g. B), vectors in bold lowercase (e.g. v), and scalars which include polynomials - in italic (e.g. $s$ ). We use the row convention for vectors. The transpose of a matrix $\mathbf{B}$ may be noted $\mathbf{B}^{\mathrm{t}}$. It is to be noted that for a polynomial $f$, we do not use $f^{\prime}$ to denote its derivative in this document.

## Quotient rings

Let $\mathbb{Z}$ and $\mathbb{N}$ denote a set of integers and a set of all numbers starting from 1 , respectively. $\mathbb{Q}$ and $\mathbb{R}$ denote a set of rational numbers and a set of real numbers, respectively. For $q \in \mathbb{N}^{\times}$, we denote by $\mathbb{Z}_{q}$ the quotient ring $\mathbb{Z} / q \mathbb{Z}$. In FALCON and SOLMAE, an integer modulus $q=12,289$ is prime, so $\mathbb{Z}_{q}$ is also a finite field. We denote by $\mathbb{Z}_{q}^{\times}$the group of invertible elements of $\mathbb{Z}_{q}$, and by $\varphi$ Euler's totient function: $\varphi(q)=\left|\mathbb{Z}_{q}^{\times}\right|=q-1=3 \cdot 2^{12}$ since $q$ is prime. The rings $\mathbb{Q}[x] /(\phi)$, $\mathbb{Z}[x] /(\phi)$, and $\mathbb{R}[x] /(\phi)$ where $\phi$ is a monic minimal polynomial will be interchangeably written as $\mathcal{Q}, \mathcal{Z}$, and $K_{\mathbb{R}}$, respectively for the sake of our convenience.

## DFT representation

For $d=2^{n}$, we use $\phi(x)=x^{d}+1$. It is a monic polynomial of $\mathbb{Z}[x]$, irreducible in $\mathbb{Q}[x]$ and with distinct roots over $\mathbb{C}$. Then $\zeta_{j}=\exp (i(2 j-1) \pi / d)$ for $j=1,2, \cdots d$ are roots of $\phi(x)$. For $f=\Sigma f_{i} x^{i} \in K_{\mathbb{R}}$, we define the coefficient representation as $\mathbf{f}=\left(f_{0}, f_{1}, \cdots f_{d-1}\right)$ and Discrete Fourier Transform(DFT) representation $\varphi(f)=\left(\varphi_{1}(f), \cdots, \varphi_{d}(f)\right)$.

## Number fields

Let $a=\sum_{i=0}^{d-1} a_{i} x^{i}$ and $b=\sum_{i=0}^{d-1} b_{i} x^{i}$ be arbitrary elements of the number field $\mathcal{Q}=\mathbb{Q}[x] /(\phi)$. We note $a^{*}$ and call (Hermitian) adjoint of $a$ the unique element of $\mathcal{Q}$ such that for any root $\zeta$ of
$\phi, a^{*}(\zeta)=\overline{a(\zeta)}$, where ${ }^{-}$is the usual complex conjugation over $\mathbb{C}$. For $\phi=x^{d}+1$, the Hermitian adjoint $a^{*}$ can be expressed simply:

$$
\begin{equation*}
a^{*}=a_{0}-\sum_{i=1}^{d-1} a_{i} x^{d-i} \tag{1}
\end{equation*}
$$

We extend this definition to vectors and matrices: the adjoint $\mathbf{B}^{*}$ of a matrix $\mathbf{B} \in \mathcal{Q}^{n \times m}$ (resp. a vector $\mathbf{v}$ ) is the component-wise adjoint of the transpose of $\mathbf{B}$ (resp. $\mathbf{v}$ ):

$$
\mathbf{B}=\left[\begin{array}{ll}
a & b  \tag{2}\\
c & d
\end{array}\right] \quad \Leftrightarrow \quad \mathbf{B}^{*}=\left[\begin{array}{ll}
a^{*} & c^{*} \\
b^{*} & d^{*}
\end{array}\right]
$$

## Inner product

The inner product $\langle\cdot, \cdot\rangle$ over $\mathcal{Q}$ and its associated norm $\|\cdot\|$ are defined as:

$$
\begin{gather*}
\langle a, b\rangle=\frac{1}{\operatorname{deg}(\phi)} \sum_{0<i \leq d} \varphi_{i}(a) \cdot \overline{\varphi_{i}(b)}  \tag{3}\\
\|a\|=\sqrt{\langle a, a\rangle} \tag{4}
\end{gather*}
$$

These definitions can be extended to vectors: for $u=\left(u_{i}\right)$ and $v=\left(v_{i}\right)$ in $\mathcal{Q}^{m},\langle u, v\rangle=\sum_{i}\left\langle u_{i}, v_{i}\right\rangle$. For our choice of $\phi$, the inner product coincides with the usual coefficient-wise inner product:

$$
\begin{equation*}
\langle a, b\rangle=\sum_{0 \leq i<d} a_{i} b_{i} \tag{5}
\end{equation*}
$$

From an algorithmic point of view, computing the inner product or the norm is most easily done using Eq.(3) if polynomials are in FFT representation, and using Eq.(5) if they are in coefficient representation. By substituting $b=a$ in Eqs (3) and (5), we get

$$
\begin{equation*}
\|\varphi(a)\|=\sqrt{d} \cdot\|a\| \tag{6}
\end{equation*}
$$

where $\|\cdot\|$ is Euclidean norm. Since we know that

$$
\begin{equation*}
\|\varphi(a)\|=\sqrt{2} \cdot\left\|\left(\operatorname{Re}\left(\varphi_{1}(a)\right), \operatorname{Im}\left(\varphi_{1}(a)\right), \cdots \operatorname{Re}\left(\varphi_{d / 2}(a)\right), \operatorname{Im}\left(\varphi_{d / 2}(a)\right)\right)\right\| \tag{7}
\end{equation*}
$$

we get

$$
\begin{equation*}
\left\|\left(\operatorname{Re}\left(\varphi_{1}(a)\right), \operatorname{Im}\left(\varphi_{1}(a)\right), \cdots \operatorname{Re}\left(\varphi_{d / 2}(a)\right), \operatorname{Im}\left(\varphi_{d / 2}(a)\right)\right)\right\|=\sqrt{\frac{d}{2}} \cdot\|a\| \tag{8}
\end{equation*}
$$

If $a \in K_{\mathbb{R}}$ follows the $d$-dimensional standard normal distribution, it is known that

$$
\begin{equation*}
\left(\operatorname{Re}\left(\varphi_{1}(a)\right), \operatorname{Im}\left(\varphi_{1}(a)\right), \cdots \operatorname{Re}\left(\varphi_{d / 2}(a)\right), \operatorname{Im}\left(\varphi_{d / 2}(a)\right)\right) \text { follows } \mathcal{N}_{d / 2} \tag{9}
\end{equation*}
$$

where $\mathcal{N}_{d / 2}$ denotes continuous Gaussian distribution with zero mean and $\frac{d}{2} \cdot I_{d}$ (i.e., Identity matrix) variance.

## Ring lattices

For the rings $\mathcal{Q}=\mathbb{Q}[x] /(\phi)$ and $\mathcal{Z}=\mathbb{Z}[x] /(\phi)$, positive integers $m \geq n$, and a full-rank matrix $\mathbf{B} \in \mathcal{Q}^{n \times m}$, we denote by $\Lambda(\mathbf{B})$ and call lattice generated by $\mathbf{B}$, the set $\mathcal{Z}^{n} \cdot \mathbf{B}=\left\{z \mathbf{B} \mid z \in \mathcal{Z}^{n}\right\}$. By extension, a set $\Lambda$ is a lattice if there exists a matrix $\mathbf{B}$ such that $\Lambda=\Lambda(\mathbf{B})$. We may say that $\Lambda \subseteq \mathcal{Z}^{m}$ is a $q$-ary lattice if $q \mathcal{Z}^{m} \subseteq \Lambda$.

## NTRU lattices

Let $q$ be an integer, and $f \in \mathbb{Z}[x] /\left(x^{d}+1\right)$ such that $f$ is invertible modulo $q$ (equivalently, $\operatorname{det}[f]$ is coprime to $q$ ). Let $h=g / f \bmod q$ and consider the NTRU module associated to $h$ :

$$
\mathcal{M}_{\mathrm{NTRU}}=\left\{(u, v) \in K_{\mathbb{R}}^{2}: h u-v=0 \bmod q\right\}
$$

and its lattice version

$$
\mathcal{L}_{\mathrm{NTRU}}=\left\{(\mathbf{u}, \mathbf{v}) \in \mathbb{Z}^{2 d}:[h] \mathbf{u}-\mathbf{v}=0 \bmod q\right\} .
$$

This lattice has volume $q^{d}$. Over $K_{\mathbb{R}}$, it is generated by $(f, g)$ and any $(F, G)$ such that $f G-g F=q$. For such a pair $(f, g),(F, G)$, this means that $\mathcal{L}_{\text {NTRU }}$ has a basis of the form

$$
\mathbf{B}_{f, g}=\left[\begin{array}{c}
{[f][F]} \\
{[g][G]}
\end{array}\right] .
$$

One checks that $\left([h],-\operatorname{Id}_{d}\right) \cdot \mathbf{B}_{f, g}=0 \bmod q$, so the verification key is $h$. The NTRU-search problem is : given $h=g / f \bmod q$, find any $\left(f^{\prime}=x^{i} f, g^{\prime}=x^{i} g\right)$. In its decision variant, one must distinguish $h=g / f \bmod q$ from a uniformly random $h \in R_{q}:=\mathbb{Z}[x] /\left(q, x^{d}+1\right)=(\mathbb{Z} / q \mathbb{Z})[x] /\left(x^{d}+1\right)$. These problems are assumed to be intractable for large $d$.

## Discrete Gaussians

For $\sigma, \mu \in \mathbb{R}$ with $\sigma>0$, we define the Gaussian function $\rho_{\sigma, \mu}$ as $\rho_{\sigma, \mu}(x)=\exp \left(-|x-\mu|^{2} / 2 \sigma^{2}\right)$, and the discrete Gaussian distribution $D_{\mathbb{Z}, \sigma, \mu}$ over the integers as:

$$
\begin{equation*}
D_{\mathbb{Z}, \sigma, \mu}(x)=\frac{\rho_{\sigma, \mu}(x)}{\sum_{z \in \mathbb{Z}} \rho_{\sigma, \mu}(z)} \tag{10}
\end{equation*}
$$

The parameter $\mu$ may be omitted when it is equal to zero.

## Gram-Schmidt orthogonalization

Any matrix $\mathbf{B} \in \mathcal{Q}^{n \times m}$ can be decomposed as follows:

$$
\begin{equation*}
\mathbf{B}=\mathbf{L} \times \tilde{\mathbf{B}} \tag{11}
\end{equation*}
$$

where $\mathbf{L}$ is lower triangular with 1's on the diagonal, and the rows $\tilde{b_{i}}$ 's of $\tilde{\mathbf{B}}$ verify $\left\langle\tilde{b_{i}}, \tilde{b_{j}}\right\rangle=0$ for $i \neq j$. When $\mathbf{B}$ is full-rank, this decomposition is unique, and it is called the Gram-Schmidt orthogonalization (or GSO). We also call the Gram-Schmidt norm of $\mathbf{B}$ the following value:

$$
\begin{equation*}
\|\mathbf{B}\|_{G S}=\max _{\mathbf{b}_{\mathbf{i}} \in \tilde{\mathbf{B}}}\left\|\tilde{\mathbf{b}_{\mathbf{i}}}\right\| \tag{12}
\end{equation*}
$$

## The LDL* decomposition

The LDL* decomposition writes any full-rank Gram matrix as a product LDL* where $\mathbf{L} \in \mathcal{Q}^{n \times n}$ is lower triangular with 1's on the diagonal, and $\mathbf{D} \in \mathcal{Q}^{n \times n}$ is diagonal. The LDL* decomposition and the GSO are closely related as for a basis $\mathbf{B}$, there exists a unique GSO $\mathbf{B}=\mathbf{L} \cdot \tilde{\mathbf{B}}$, and for a full-rank Gram matrix $\mathbf{G}$, there exists a unique $\mathbf{L D L}^{*}$ decomposition $\mathbf{G}=\mathbf{L D L}^{*}$. If $\mathbf{G}=\mathbf{B B}^{*}$, then $\mathbf{G}=\mathbf{L} \cdot\left(\tilde{\mathbf{B}} \tilde{\mathbf{B}}^{*}\right) \cdot \mathbf{L}^{*}$ is a valid $\mathrm{LDL}^{*}$ decomposition of $\mathbf{G}$. As both decompositions are unique, the matrices $\mathbf{L}$ in both cases are actually the same. In a nutshell:

$$
\begin{equation*}
[\mathbf{L} \cdot \tilde{\mathbf{B}} \text { is the GSO of } \mathbf{B}] \Leftrightarrow\left[\mathbf{L} \cdot\left(\mathbf{B} \tilde{\mathbf{B}}^{*}\right) \cdot \mathbf{L}^{*} \text { is the } \mathrm{LDL}^{*} \text { decomposition of }\left(\mathbf{B B}^{*}\right)\right] \tag{13}
\end{equation*}
$$

The reason why we present both equivalent decompositions is that the GSO is a more familiar concept in lattice-based cryptography, whereas the use of LDL* decomposition is faster and therefore makes more sense from an algorithmic point of view.

## 3 How FALCON works

A group of top-notch cryptographers, Hoffstein, Pipher and Silverman [8] suggested new public-key cryptosystem based on a polynomial ring in 1997 as an alternative to RSA and DH whose difficulties are based on number-theoretic hard problems such as integer factorization and discrete log problem, respectively. They founded the company so-called as NTRU ${ }^{1}$ Cryptosystem with Lieman and initiated an open-source lattice-based cryptography consisting of two algorithms: NTRUENCRYPT used for encryption/decryption and NTRUSIGN used for digital signatures. Their security relies on the presumed difficulty of factoring certain polynomials in a truncated polynomial ring into a quotient of two polynomials having very small coefficients.

NTRUSIGN was designed based on the GGH signature scheme [7] which was proposed in 1995 based on solving the closest vector problem (CVP) in a lattice and asymptotically is more efficient than RSA in the computation time for encryption, decryption, signing, and verifying are all quadratic in the natural security parameter. The signer demonstrates knowledge of a good basis for the lattice by using it to solve CVP on a point representing the message; the verifier uses a bad basis for the same lattice to verify that the signature under consideration is actually a lattice point and is sufficiently close to the message point.

On the other hand, Min et al.[12] suggested weak property of malleability of NTRUSIGN using the annihilating polynomial from a given message and signature pair to generate a valid signature. Nguyen and Regev [13] had cryptanalyzed the original GGH signature scheme including NTRUSIGN in 2006 successfully extracting secret information from many known signatures characterized by multivariate optimization problems. Their experiments showed that 90,000 signatures are sufficient to recover the NTRUSign-251 secret key.

In a nutshell, FALCON follows a framework introduced in 2008 by Gentry, Peikert, and Vaikuntanathan [6] which we call the GPV framework for short over the NTRU lattices and uses a typically hash-and-sign paradigm. Their high-level idea is the following:

1. The public key is a long basis of a $q$-ary lattice.
2. The private key is (essentially) a short basis of the same lattice.
3. In the signing procedure, the signer:
(a) generates a random value, salt;
(b) computes a target $\mathbf{c}=H(M \|$ salt $)$, where $H$ is a hash function sending input to a randomlooking point (on the grid);
(c) uses his knowledge of a short basis to compute a lattice point $\mathbf{v}$ close to the target $\mathbf{c}$;
(d) outputs (salt, $\mathbf{s}$ ), where $\mathbf{s}=\mathbf{c}-\mathbf{v}$.
4. The verifier accepts the signature ( $s a l t, \mathbf{s}$ ) if and only if:
(a) the vector s is short;
(b) $H(M \|$ salt $)-\mathbf{s}$ is a point on the lattice generated by his public key.

Only the signer should be able to efficiently compute $v$ close enough to an arbitrary target. This is a decoding problem that can be solved when a basis of short vectors is known. On the other hand, anyone wanting to check the validity of a signature should be able to verify lattice membership. The KeyGen, Sign and Verif procedures for FALCON will be introduced briefly in the later Section by restating the original specification as in [3]. For details, the readers can refer to [3].

### 3.1 Key Generation of FALCON

For the class of NTRU lattices, a trapdoor pairs is $\left(h, \mathbf{B}_{f, g}\right)$ where $h=f^{-1} g, \mathbf{B}_{f, g}$ is trapdoor basis over $\mathcal{L}_{\text {NTRU }}$ and Pornin \& Prest [15] showed that a completion $(F, G)$ can be computed in $O(d \log d)$ time from short polynomials $f, g \in \mathcal{Z}$. In practice, their implementation is as efficient as can be for this technical procedure: it is called NtruSolve in FALCON. Their algorithm only depends on the underlying ring and has now a stable version for $\mathbb{Z}[x] /\left(x^{d}+1\right)$, where $d=2^{n}$.

Figure 1 illustrates the flowchart of the key generation procedure for FALCON.

[^23]

Fig. 1: Flowchart of KeyGen for FALCON

Algorithm 1 describes the pseudo-code for key generation of FALCON.

```
Algorithm 1: KeyGen of FALCON
    Input: A monic polynomial \(\phi \in \mathbb{Z}[x]\), a modulus \(q\)
    Output: A secret key sk, a public key pk
        \(f, g, F, G \leftarrow\) NtruGen ; /* Solving the NTRU equation */
        \(\mathbf{B} \leftarrow\left[\begin{array}{l}g-f \\ G-F\end{array}\right] ;\)
        \(\hat{\mathbf{B}} \leftarrow \mathrm{FFT}(\mathbf{B}) ; \quad / *\) Compute FFT for each \(\{g,-f, G,-F\} * /\)
        \(\mathbf{G} \leftarrow \hat{\mathbf{B}} \times \hat{\mathbf{B}}^{*} ;\)
        \(\mathrm{T} \leftarrow \mathrm{flDL}^{*}(\mathbf{G}) ; \quad\) /* Compute the LDL* tree */
        for each leaf of T do
            leaf.value \(\leftarrow \sigma / \sqrt{\text { leaf.value } ; \quad / * \text { Normalization step */ }}\)
        sk \(\leftarrow(\hat{\mathbf{B}}, \mathrm{T})\);
        \(h \leftarrow g f^{-1} \bmod q ;\)
        \(\mathrm{pk} \leftarrow h\);
        return sk , pk ;
```


### 3.2 Signing of FALCON

At a high level, the signing procedure in FALCON is at first to compute a hashed value $\mathbf{c} \in \mathbb{Z}_{q}[x] /(\phi)$ from the message, M and a salt $r$, then using the secret key, $f, g, F, G$ to generate two short values $\left(\mathbf{s}_{1}, \mathbf{s}_{2}\right)$ such that $\mathbf{s}_{1}+\mathbf{s}_{2} h=\mathbf{c} \bmod q$. An interesting feature is that only the first half of the signature ( $\mathbf{s}_{1}, \mathbf{s}_{2}$ ) needs to be sent along the message, as long as $h$ is available to the verifier. This comes from the identity $h \mathbf{s}_{1}=\mathbf{s}_{2} \bmod q$ defining these lattices, as we will see in the Verif algorithm description.

The core of FALCON signing is to use ffSampling (Algorithm 11 in [3]) which applies a randomizing rounding according to Gaussian distribution on the coefficient of $\mathbf{t}=\left(\mathbf{t}_{0}, \mathbf{t}_{1}\right) \in$ $(\mathbb{Q}[x] /(\phi))^{2}$ stored in the FALCON Tree, $\mathbf{T}$ at the KeyGen procedure of FALCON.

This fast Fourier sampling algorithm can be seen as a recursive version of Klein's well-known trapdoor sampler, but cannot be computed in parallel also known as the GPV sampler. Klein's sampler uses a matrix $\mathbf{L}$ and the norm of Gram-Schmidt vectors as a trapdoor while FALCON are using a tree of non-trivial elements in such matrices. Note that Fouque et. al.[4] suggested Gram-Schmidt norm leakage in FALCON by timing side channels in the implementation of the one-dimensional Gaussian samplers.

FALCON cannot output two different signatures for a message. This well-known concern of the GPV framework can be addressed in several ways, for example, making a stateful scheme or by hash randomization. FALCON chose the latter solution for efficiency purposes. In practice, Sign adds a random "salt" $r \in\{0,1\}^{k}$, where $k$ is large enough that an unfortunate collision of messages is unlikely to happen, that is, it hashes $(r \| M)$ instead of $M$. A signature is then sig $=\left(r\right.$, Compress $\left.\left(\mathbf{s}_{1}\right)\right)$.

Figure 2 and Algorithm 2 sketches the signing procedure for FALCON and shows its pseudocode for FALCON, respectively.


Fig. 2: Flowchart of Sign for FALCON.

```
Algorithm 2: Sign of FALCON
    Input: A message \(M \in\{0,1\}^{*}\), secret key sk, a bound \(\gamma\).
    Output: A pair ( \(r\), Compress \(\left.\left(\mathbf{s}_{1}\right)\right)\) with \(r \in\{0,1\}^{320}\) and \(\left\|\left(\mathbf{s}_{1}, \mathbf{s}_{2}\right)\right\| \leq \gamma\).
        \(r \leftarrow \mathcal{U}\left(\{0,1\}^{320}\right) ;\)
        \(\mathbf{c} \leftarrow \operatorname{HashToPoint}(r \| M, q, n)\);
        \(\mathbf{t} \leftarrow\left(-\frac{1}{q} \operatorname{FFT}(c) \odot \operatorname{FFT}(F), \frac{1}{q} \operatorname{FFT}(c) \odot \operatorname{FFT}(f)\right) ; \quad / * \mathbf{t}=(\operatorname{FFT}(c), \operatorname{FFT}(0)) \cdot \hat{\mathbf{B}}^{-1} * /\)
    do
        do
            \(\mathbf{z} \leftarrow\) ffSampling \(_{n}(\mathbf{t}, \mathrm{~T}) ;\)
            \(\mathbf{s}=(\mathbf{t}-\mathbf{z}) \hat{\mathbf{B}} ; \quad / *\) At this point, \(\mathbf{s}\) follows Gaussian distribution. */
            while \(\|s\|^{2}>\gamma\)
            \(\left(s_{1}, s_{2}\right) \leftarrow \mathrm{FFT}^{-1}(\mathbf{s}) ;\)
            \(s \leftarrow\) Compress \(\left(s_{2}, 8 \cdot\right.\) sbytelen -328\()\); \(\quad / *\) Remove 1 byte for the header, and 40
    bytes for r */
    while \((s=\perp)\)
    return \((r, s)\);
```


### 3.3 Verification of FALCON:

The last step of the scheme is thankfully simpler to describe. Upon receiving a signature ( $r, \mathbf{s}$ ) and message $M$, the verifier decompresses $\mathbf{s}$ to a polynomial $\mathbf{s}_{1}$ and $\mathbf{c}=(0, \mathrm{H}(r \| M))$, then wants to recover the full signature vector $\mathbf{v}=\left(\mathbf{s}_{1}, \mathbf{s}_{2}\right)$. If $\mathbf{v}$ is a valid signature, the verification identity is $(h,-1) \cdot(\mathbf{c}-\mathbf{v})=-\mathrm{H}(r \| M)-h \mathbf{s}_{1}+\mathbf{s}_{2} \bmod q=0$, or equivalently the verifier can compute

$$
\mathbf{s}_{2}=\mathrm{H}(r \| M)+h \mathbf{s}_{1} \bmod q .
$$

This is computed in the ring $R_{q}$, and can be done very efficiently for a good choice of modulus $q$ using the Number Theoretic Transform (NTT). FALCON currently follow the standard choice of $q=12,289$, as the multiplication in NTT format amounts to $d$ integer multiplications in $\mathbb{Z} / q \mathbb{Z}$. The last step is to check that $\left\|\left(\mathbf{s}_{1}, \mathbf{s}_{2}\right)\right\|^{2} \leq \gamma^{2}$ : the signature is only accepted in this case. The rejection bound $\gamma$ comes from the expected length of vectors outputted by Sample described as Algorithm 4 in [9].

Since they are morally Gaussian, they concentrate around their standard deviation; a "slack" parameter $\tau=1.042$ is tuned to ensure that $90 \%$ of the vectors generated by Sample will get through the loop:

$$
\gamma=\tau \cdot \sigma_{\mathrm{sig}} \cdot \sqrt{2 d}
$$

Algorithm 3 shows the pseudo-code of verification procedure of FALCON.

```
Algorithm 3: Verif of FALCON
    Input: A signature \((r, \mathbf{s})\) on \(M\), a public key \(\mathrm{pk}=h\), a bound \(\gamma\).
    Output: Accept or Reject.
        \(\mathbf{s}_{1} \leftarrow\) Decompress(s);
        \(\mathbf{c} \leftarrow \mathrm{H}(r \| M) ;\)
        \(\mathbf{s}_{2} \leftarrow \mathbf{c}+h \mathbf{s}_{1} \bmod q ;\)
        if \(\left\|\left(\mathbf{s}_{1}, \mathbf{s}_{2}\right)\right\|^{2}>\gamma^{2}\) then
        return Reject.
        end
        return Accept.
```


## 4 How SOLMAE works

SOLMAE is inspired from FALCON's design. Some of the new theoretical foundations were laid out in the presentation of Mitaka [1]. At a high level, it removes the inherent technicality of the sampling procedure, and most of its induced complexity from an implementation standpoint, for free, that is with no loss of efficiency. This simplicity translates into faster operations while preserving signatures and verification keys sizes, on top of allowing for additional features absent from FALCON, such as enjoying cheaper masking, and being parallelizable. By using the novel compression techniques and tools of [2], SOLMAE can also obtain smaller signatures and verification keys than those already achieved by FALCON. To sum up, SOLMAE aims to achieve better performances for the same security and advantages as FALCON.

While its predecessor FALCON could be summed-up as an efficient instantiation of the GPV framework, SOLMAE takes it one step further. The main ingredients in SOLMAE are:

- Hybrid sampler is a faster, simpler, parallelizable, and maskable Gaussian sampler to generate signatures;
- Optimally tuned key generation algorithm, enhancing the security of the used hybrid sampler to that of FALCON's level ${ }^{2}$;
- Dedicated compression techniques to reduce bandwidth consumption even further, at no cost on the security according to our analyses.

The KeyGen, Sign and Verif procedures for SOLMAE will be introduced briefly in the later Section by restating the original specification in [9]. For details, the readers can refer to [9].

### 4.1 Key Generation of SOLMAE

An important concern here is that not all pair $(f, g),(F, G)$ gives good trapdoor pairs for Sample described as Algorithm 4 in [9]. Schemes such as FALCON and Mitaka solve this technicality essentially by sieving among all possible bases to find the ones that reach an acceptable quality for the Sample procedure. This technique is costly, and many tricks were used to achieve an acceptable KeyGen. This sieving routine was bypassed by redesigning completely how good quality bases can be found. This improves the running time of KeyGen and also increases the security offered by Sample. In any case, note that NtruSolve's running time largely dominates the overall time for KeyGen: this is not avoidable as the basis completion algorithm requires working with quite large integers and relatively high-precision floating-point arithmetic.

At the end of the procedure, the secret key contains not only the secret basis but also the necessary data for Sign and Sample. This additional information can be represented by elements in $K_{\mathbb{R}}$ and is computed during or at the end of NtruSolve. All-in-all, KeyGen outputs:

$$
\begin{aligned}
\mathrm{sk} & =\left(\mathbf{b}_{1}=(f, g), \mathbf{b}_{2}=(F, G), \widetilde{\mathbf{b}}_{2}=(\widetilde{F}, \widetilde{G}), \Sigma_{1}, \Sigma_{2}, \beta_{1}, \beta_{2}\right), \\
\mathrm{pk} & =\left(h, q, \sigma_{\mathrm{sig}}, \eta\right),
\end{aligned}
$$

[^24]where we recall that $h=g / f \bmod q$. These parameters and a table of their practical values are described more thoroughly in [9].

Informally, they correspond to the following:

- $(f, g),(F, G)$ is a good basis of the lattice $\mathcal{L}_{\text {NTRU }}$ associated to $h$, with quality $\mathcal{Q}(f, g)=\alpha$, and $\widetilde{\mathbf{b}}_{2}$ is the Gram-Schmidt orthogonalization of $(F, G)$ with respect to $(f, g)$;
- $\sigma_{\text {sig }}, \eta$ are respectively the standard deviation for signature vectors, and a tight upper bound on the "smoothing parameter of $\mathbb{Z}^{d}$ ";
- $\Sigma_{1}, \Sigma_{2} \in K_{\mathbb{R}}$ represent covariance matrices for two intermediate Gaussian samplings in Sample;
- the vectors $\beta_{1}, \beta_{2} \in K_{\mathbb{R}}^{2}$ represent the orthogonal projections from $K_{\mathbb{R}}^{2}$ onto $K_{\mathbb{R}} \cdot \mathbf{b}_{1}$ and $K_{\mathbb{R}} \cdot \widetilde{\mathbf{b}}_{2}$ respectively. In other words, they act as "getCoordinates" for vectors in $K_{\mathbb{R}}^{2}$. They are used by Sample and are precomputed for efficiency.

Algorithm 4 computes the necessary data for signature sampling, then outputs the key pair. Note that NtruSolve could also compute the sampling data and the public key, but for clarity, the pseudo-code gives these tasks to KeyGen of SOLMAE. Figure 3 sketches the key generation procedure of SOLMAE


Fig. 3: Flowchart of KeyGen of SOLMAE.

```
Algorithm 4: KeyGen of SOLMAE
    Input: A modulus \(q\), a target quality parameter \(1<\alpha\), parameters \(\sigma_{\text {sig }}, \eta>0\)
    Output: A basis \(((f, g),(F, G)) \in R^{2}\) of an NTRU lattice \(\mathcal{L}_{\text {NTRU }}\) with \(\mathcal{Q}(f, g)=\alpha\);
        repeat
        \(\left.\mathbf{b}_{1}:=(f, g) \leftarrow \operatorname{PairGen}\left(q, \alpha, R_{-}, R_{+}\right)\right\}\)
        until \(f\) is invertible modulo \(q\);
        ; \(/ *\) Secret basis computation between \(R_{-}\)and \(R_{+}\)*/
        \(\mathbf{b}_{2}:=(F, G) \leftarrow \operatorname{NtruSolve}(q, f, g)\) :
        \(h \leftarrow g / f \bmod q ; \quad / *\) Public key data computation */
        \(\gamma \leftarrow 1.1 \cdot \sigma_{\text {sig }} \cdot \sqrt{2 d} ; \quad / *\) tolerance for signature length */
        \(\beta_{1} \leftarrow \frac{1}{\left\langle\mathbf{b}_{1}, \mathbf{b}_{1}\right\rangle_{K}} \cdot \mathbf{b}_{1} ; \quad / *\) Sampling data computation, in Fourier domain */
        \(\Sigma_{1} \leftarrow \sqrt{\frac{\sigma_{\text {sig }}^{2}}{\left\langle\mathbf{b}_{1}, \mathbf{b}_{1}\right\rangle_{K}}-\eta^{2}} ;\)
        \(\widetilde{\mathbf{b}}_{2}:=(\widetilde{F}, \widetilde{G}) \leftarrow \mathbf{b}_{2}-\left\langle\beta_{1}, \mathbf{b}_{2}\right\rangle \cdot \mathbf{b}_{1} ;\)
        \(\beta_{2} \leftarrow \frac{1}{\left\langle\mathbf{b}_{2}, \mathbf{b}_{2}\right\rangle_{K}} \cdot \widetilde{\mathbf{b}}_{2} ;\)
        \(\Sigma_{2} \leftarrow \sqrt{\frac{\sigma_{\text {sig }}^{2}}{\left\langle\mathbf{b}_{2}, \mathbf{b}_{2}\right\rangle_{K}}-\eta^{2}} ;\)
        \(\mathbf{s k} \leftarrow\left(\mathbf{b}_{1}, \mathbf{b}_{2}, \widetilde{\mathbf{b}}_{2}, \Sigma_{1}, \Sigma_{2}, \beta_{1}, \beta_{2}\right) ;\)
        \(\mathrm{pk} \leftarrow\left(q, h, \sigma_{\text {sig }}, \eta, \gamma\right) ;\)
        return sk, pk;
```

The function of two subroutines PairGen and NtruSolve are described below:

1. The PairGen algorithm generates $d$ complex numbers $\left(x_{j} e^{i \theta_{j}}\right)_{j \leq d / 2},\left(y_{j} e^{i \theta_{j}}\right)_{j \leq d / 2}$ to act as the FFT representations of two real polynomial $f^{\mathbb{R}}, g^{\mathbb{R}}$ in $K_{\mathbb{R}}$. The magnitude of these complex numbers is sampled in a planar annulus whose small and big radii are set to match a target $\mathcal{Q}(f, g)$ with UnifCrown ([9]). It then finds close elements $f, g \in \mathcal{Z}$ by round-off, unless maybe the rounding error was too large. When the procedure ends, it outputs a pair $(f, g)$ such that $\mathcal{Q}(f, g)=\alpha$, where $\alpha$ depends on the security level.
2. NtruSolve is exactly Pornin \& Prest's algorithm and implementation [15]. It takes as input $(f, g) \in \mathcal{Z}^{2}$ and a modulus $q$, and outputs $(F, G) \in \mathcal{Z}^{2}$ such that $(f, g),(F, G)$ is a basis of $\mathcal{L}_{\text {NTRU }}$ associated to $h=g / f \bmod q$. It does so by solving the Bézout-like equation $f G-g F=q$ in $\mathcal{Z}$ using recursively the tower of subfields for optimal efficiency.

### 4.2 Signing of SOLMAE

Recall that NTRU lattices live in $\mathbb{R}^{2 d}$. Their structure also helps to simplify the preimage computation. Indeed, the signer only needs to compute $\mathbf{m}=H(M) \in \mathbb{R}^{d}$, as then $\mathbf{c}=(0, \mathbf{m})$ is a valid preimage: the corresponding polynomials satisfy $(h, 1) \cdot \mathbf{c}=\mathbf{m}$.

As the same with Sign procedure of FALCON, an interesting feature is that only the first half of the signature ( $\mathbf{s}_{1}, \mathbf{s}_{2}$ ) $\in \mathcal{L}_{\text {NTRU }}$ needs to be sent along the message, as long as $h$ is available to the verifier. This comes from the identity $h \mathbf{s}_{1}=\mathbf{s}_{2} \bmod q$ defining these lattices, as we will see in the Verif algorithm description. ${ }^{3}$

Because of their nature as Gaussian integer vectors, signatures can be encoded to reduce the size of their bit-representation. The standard deviation of Sample is large enough so that the $\lfloor\log \sqrt{\bar{q}}\rfloor$ least significant bits of one coordinate are essentially random.

In practice, Sign adds a random "salt" $r \in\{0,1\}^{k}$, where $k$ is large enough that an unfortunate collision of messages is unlikely to happen, that is, it hashes $(r \| M)$ instead of $M$ - our analysis in this regard is identical to FALCON. A signature is then $\operatorname{sig}=\left(r, \operatorname{Compress}\left(s_{1}\right)\right)$. SOLMAE cannot output two different signatures for a message like FALCON which was mentioned in Section 3.2 .

Figure 4 sketches the signing procedure of SOLMAE and Algorithm 5 shows its pseudo-code.


Fig. 4: Flowchart of Sign of SOLMAE.

[^25]```
Algorithm 5: Sign of SOLMAE
    Input: A message \(M \in\{0,1\}^{*}\), a tuple sk \(=\left((f, g),(F, G),(\widetilde{F}, \widetilde{G}), \sigma_{\text {sig }}, \Sigma_{1}, \Sigma_{2}, \eta\right)\), a
                rejection parameter \(\gamma>0\).
    Output: A pair ( \(r\), Compress \(\left.\left(\mathbf{s}_{1}\right)\right)\) with \(r \in\{0,1\}^{320}\) and \(\left\|\left(\mathbf{s}_{1}, \mathbf{s}_{2}\right)\right\| \leq \gamma\).
        \(r \leftarrow \mathcal{U}\left(\{0,1\}^{320}\right)\);
        \(\mathbf{c} \leftarrow(0, \mathrm{H}(r \| M)) ;\)
        \(\hat{\mathbf{c}} \leftarrow \mathrm{FFT}(\mathbf{c})\);
        repeat
        \(\left(\hat{s}_{1}, \hat{s}_{2}\right) \leftarrow \hat{\mathbf{c}}-\operatorname{Sample}(\hat{\mathbf{c}}, \mathbf{s k}) ; \quad / *\left(\mathbf{s}_{1}, \mathbf{s}_{2}\right) \leftarrow D_{\mathcal{L}_{\text {NTR }}, \mathbf{c}, \sigma_{\text {sig }}} \quad * /\)
        until \(\left\|\left(\mathrm{FFT}^{-1}\left(\hat{s}_{1}\right), \mathrm{FFT}^{-1}\left(\hat{s}_{2}\right)\right)\right\|^{2} \leq \gamma^{2} ;\)
    \(s_{1} \leftarrow \mathrm{FFT}^{-1}\left(\hat{s}_{1}\right)\);
    \(s \leftarrow\) Compress \(\left(s_{1}\right)\);
        return \((r, s)\);
```


### 4.3 Verification of SOLMAE

This is the same as the Verification of FALCON stated in Section 3.3.

## 5 Asymptotic Complexity of FALCON and SOLMAE

To the best of our allowable knowledge as of writing this paper, we will suggest the asymptotic computational complexity of FALCON and SOLMAE algorithms with their pseudo-codes described their specifications based on the following assumptions to make our computation work to be simple:
(i) Multiplication of large integers can be done by integer-type Karatsuba algorithm or SchönhageStrassen algorithm. However, we assumed multiplication of large integers can be done in $\Theta(1)$.
(ii) The multiplication and division of polynomials in $\mathbb{Z}[x] /\left(x^{d}+1\right)$ or $\mathbb{Q}[x] /\left(x^{d}+1\right)$ are assumed to compute the polynomial-type Karatsuba algorithm or operate pointwise in Fourier domain. It is known that the time complexity of the Karatsuba algorithm and FFT (or $\mathrm{FFT}^{-1}$ ) are $\Theta\left(d^{3 / 2}\right)$ and $\Theta(d \log (d))$, respectively. We assume that all polynomial operations are done in the Fourier domain, so polynomial multiplication and division in $\mathbb{Z}[x] /\left(x^{d}+1\right)$ or $\mathbb{Q}[x] /\left(x^{d}+1\right)$ takes $\Theta(d \log (d))$ time. Since every inverse element of $\mathbb{Z}_{q}$ is stored in the list and the division of polynomials in $\mathbb{Z}_{q}[x] /\left(x^{d}+1\right)$ can be done in the NTT domain, the division of polynomials in $\mathbb{Z}_{q}[x] /\left(x^{d}+1\right)$ also takes $\Theta(d \log d)$.
(iii) Some number of rejection samplings may inevitably happen in FALCON and SOLMAE. If one-loop for rejection sampling takes $t$ times and its probability of the acceptance is $p$, the expectation value of the total time is $\sum_{k=1}^{\infty} p(1-p)^{k-1} \cdot k t=\frac{t}{p} \approx t$ since the value $1 / p$ does not influence our asymptotic analysis due to its fixed constant value. So, we may ignore the number of rejections occurred in the rejection sampling. In fact, our experiment reveals that more or less 5 times rejections have occurred.
(iv) Ignore some minor operations and trivial computations which do not affect the total asymptotic complexity so much.

### 5.1 Asymptotic Complexity of FALCON

Using the previous assumption stated in Section 5, Table 1 is the detailed analysis of the asymptotic complexity of KeyGen in FALCON from its algorithm whose total complexity to complete takes $\Theta(d \log d)$.

Table 1: Asymptotic complexity of KeyGen in FALCON

| No. | Computation | Complexity | Location | Comment $(d$ is degree) |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | NTRUGen $(\phi, q)$ | $\Theta(d \log d)$ | Step 1 of Alg. 1 | See below $^{\dagger}$ |  |  |  |  |
| 2 | FFT $(f)$ | $\Theta(d \log d)$ | Step 3 of Alg. 1 |  |  |  |  |  |
| 3 | $\mathbf{B} \times \hat{\mathbf{B}}^{*}$ | $\Theta(d \log d)$ | Step 4 of Alg. 1 | Polynomial multiplications |  |  |  |  |
| 4 | fLDL $^{*}(\mathbf{G})$ | $\Theta(d \log d)$ | Step 5 of Alg. 1 | See below |  |  |  |  |
| 5 | Normalization $^{\ddagger}$ | $\Theta(d)$ | Step 6-7 of Alg. 1 $d$ leaf nodes in FALCON tree |  |  |  |  |  |
| 6 | $g f^{-1} \bmod q$ | $\Theta(d \log d)$ | Step 9 of Alg. 1 | See the beginning of Section 5 |  |  |  |  |
| Total Complexity of KeyGen : $\Theta(d \log d)$ |  |  |  |  |  |  |  |  |

${ }^{\dagger}$ In Algorithm 6: NTRUGen(), Step 2 and Step 5(or 6) take $\Theta(d)$ and $\Theta(d \log d)$, respectively.
Since the recurrence relation of NtruSolve is $T(d)=T(d / 2)+\Theta(d \log d)$, thus Step 8 in Algorithm 6 takes $\Theta(d \log d)$.
${ }^{\ddagger}$ Algorithm 9: $\operatorname{ffLDL}^{*}(\mathbf{G})$ in [3] recursively calls ffLDL ${ }^{*}\left(\mathbf{G}_{0}\right)$ and $\operatorname{ffLDL}^{*}\left(\mathbf{G}_{1}\right)$, and other processes such as LDL* and Splitfft both take $\Theta(d)$, so the recursive formula is $T(d)=$ $2 T(d / 2)+\Theta(d)$. From this, we can get $T(d)=\Theta(d \log d)$.

```
Algorithm 6: NTRUGen \((\phi, q)\)
    Input: A monic polynomial \(\phi \in \mathbb{Z}[x]\) of degree \(n\), a modulus \(q\)
    Output: Polynomials \(f, g, F, G\)
        \(\sigma \leftarrow 1.17 \sqrt{q / 2 n} ;\)
        2: for \(i\) from 0 to \(n-1\) do
        \(f_{i} \leftarrow D_{\mathbb{Z}, \sigma_{\{f, g\}}, 0} ;\)
        \(g_{i} \leftarrow D_{\mathbb{Z}, \sigma_{\{f, g\}}, 0} ;\)
        end
        \(f \leftarrow \Sigma_{i} f_{i} x^{i} ;\)
        \(g \leftarrow \Sigma_{i} g_{i} x^{i} ;\)
    5: if \(N T T(f)\) contains 0 as a coefficient then
    restart
        end
    6: \(\gamma \leftarrow \max \left\{\|(g,-f)\|,\left\|\left(\frac{q f *}{f f *+g g *}, \frac{q g *}{f f *+g g *}\right)\right\|\right\}\);
    7: if \(\gamma>1.17 \sqrt{q}\) then
    | restart
        end
        ;
    8: \(F, G \leftarrow\) NtruSolve \(_{n, q}(f, g)\);
    9: if \((F, G)=\perp\) then
    I restart
        end
        return \(f, g, F, G\);
```

Tables 2 and 3 are the asymptotic complexity of Sign and Verif in FALCON, respectively whose total complexity to complete takes $\Theta(d \log d)$.

Table 2: Asymptotic complexity of Sign in FALCON

| No. | Computation | Complexity | Location | Comment( $d$ is degree) |
| :---: | :---: | :---: | :---: | :---: |
| 1 | HashToPoint ( $r \\| M, q$, | $\Theta(d)$ | Step 2 of Alg. 2 |  |
| 2 | FFT | $\Theta(d \log d)$ | Step 3 of Alg. 2 |  |
| 3 | $\operatorname{ampling}_{n}(\mathbf{t}, T)$ | $\Theta(d \log d)$ | Step 6 of Alg. 2 | See below $\dagger$ |
| 4 | $(\mathbf{t}-\mathbf{z}) \hat{\mathbf{B}}$ | $\Theta(d \log d)$ | Step 7 of Alg. 2 | Polynomial multiplicatio |
| 5 | $\\|s\\|^{2}$ | $\Theta(d)$ | Step 8 of Alg. 2 | Calculating norm |
| 6 | invFFT | $\Theta(d \log d)$ | Step 9 of Alg. 2 |  |
| 7 | Compress | $\Theta(d)$ | Step 10 of Alg. 2 | See below $\ddagger$ |
| Total Complexity of Sign: $\Theta(d \log d)$ |  |  |  |  |
| ${ }^{\dagger}$ ffSampling $_{d}$ recursively calls ffSampling ${ }_{d / 2}$ two times, and other processes such as splitfft and mergefft take $\Theta(d)$. So, the recursive formula is $T(d)=2 T(d / 2)+\Theta(d)$. If we solve this, we get $T(d)=\Theta(d \log d)$. <br> ${ }^{\ddagger}$ The compression function converts $d$ degree polynomial into string of length slen $(=666)$ slen $\approx d$, so it is irrelevant to say that the compression function takes $\Theta(d)$. |  |  |  |  |

Table 3: Asymptotic complexity of Verif in FALCON

| No. | Computation | Complexity | Location | Comment $(d$ is degree) |
| :---: | :---: | :---: | :---: | :---: |
| 1 | HashToPoint $(r \\| m, q, n)$ | $\Theta(d)$ | Step 1 of Alg. 3 |  |
| 2 | Decompress $(\mathbf{s}, 8 \cdot \operatorname{sbytelen}-328)$ | $\Theta(d)$ | Step 2 of Alg. 3 More or less on par with Compress in Table 2 |  |
| 3 | $c-s_{2} h \bmod q$ | $\Theta(d \log d)$ | Step 5 of Alg. 3 | Polynomial multiplication |
| 4 | $\left\\|\left(s_{1}, s_{2}\right)\right\\|^{2}$ | $\Theta(d)$ | Step 6 of Alg. 3 | Calculating norm |
| Total Complexity of Verif: |  |  |  |  |

### 5.2 Asymptotic Complexity of SOLMAE

Based on the previous assumption stated in Section 5 as the same manner as we analyze the asymptotic complexity of FALCON, Table 4 is the asymptotic complexity of KeyGen in SOLMAE whose total complexity to complete takes $\Theta(d \log d)$.

Table 4: Asymptotic complexity of KeyGen in SOLMAE

| No. | Computation | Complexity | Location | Comment $(d$ is degree $)$ |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | Pairgen | $\Theta(d \log d)$ | Step 1 of Alg. 4 | See below $\dagger$ |  |  |  |  |
| 2 | NtruSolve $(q, f, g)$ | $\Theta(d \log d)$ | Step 2 of Alg. 4 | Explained in Table 1 |  |  |  |  |
| 3 | $g / f \bmod q$ | $\Theta(d \log d)$ | Step 3 of Alg. 4 | Polynomial operations |  |  |  |  |
| 4 | Key computations | $\Theta(d \log d)$ | Step 4-9 of Alg. 4 Polynomial operations |  |  |  |  |  |
| Total Complexity of KeyGen: |  |  |  |  |  | $\Theta(d \log d)$ |  |  |

${ }^{\dagger}$ In Algorithm 7:PairGen, Steps 1,3,and 5 all take $\Theta(d)$ time. Steps 2 and 4 take $\Theta(d \log d)$ time.

```
Algorithm 7: PairGen
    Input: A modulus \(q\), a target quality parameter \(1<\alpha\), two radii parameters \(0<R_{-}<R_{+}\)
    Output: A pair \((f, g)\) with \(\mathcal{Q}(f, g)=\alpha\)
        1: for \(i=1\) to \(d / 2\) do
        \(x_{i}, y_{i} \leftarrow \operatorname{UnifCrown}\left(R_{-}, R_{+}\right) ; \quad / *\) see Algorithm 9 in [9] */
        \(\theta_{x}, \theta_{y} \leftarrow \mathcal{U}(0,1) ;\)
        \(\varphi_{f, i} \leftarrow\left|x_{i}\right| \cdot e^{2 i \pi \theta_{x}} ;\)
        \(\varphi_{g, i} \leftarrow\left|y_{i}\right| \cdot e^{2 i \pi \theta_{y}} ;\)
        end
    2: \(\left(f^{\mathbb{R}}, g^{\mathbb{R}}\right) \leftarrow\left(\mathrm{FFT}^{-1}\left(\left(\varphi_{f, i}\right)_{i \leq d / 2}\right), \mathrm{FFT}^{-1}\left(\left(\varphi_{g, i}\right)_{i \leq d / 2}\right)\right)\);
    3: \((\mathbf{f}, \mathbf{g}) \leftarrow\left(\left\lfloor f_{i}^{\mathbb{R}}\right\rceil\right)_{i \leq d / 2},\left(\left\lfloor g_{i}^{\mathbb{R}}\right\rceil\right)_{i \leq d / 2} ;\)
    4: \((\varphi(f), \varphi(g)) \leftarrow(\operatorname{FFT}(\mathbf{f}), \mathrm{FFT}(\mathbf{g})) ;\)
        for \(i=1\) to \(d / 2\) do
        if \(q / \alpha^{2}>\left|\varphi_{i}(f)\right|^{2}+\left|\varphi_{i}(g)\right|^{2}\) or \(\alpha^{2} q<\left|\varphi_{i}(f)\right|^{2}+\left|\varphi_{i}(g)\right|^{2}\) then
            restart;
        end
        end
        return (f, g);
```

Table 5 is the asymptotic complexity of Sign in SOLMAE whose total complexity to complete takes $\Theta(d \log d)$.

Table 5: Asymptotic complexity of Sign in SOLMAE

| No. | Computation | Complexity |  | Location |
| :---: | :---: | :---: | :---: | :---: |
| 1 | $\mathbf{H}(r \\| M)$ | $\Theta(d)$ | Step 2 of Alg. 5 This is same as HashToPoint() |  |
| 2 | $\mathrm{FFT}(\mathbf{c})$ | $\Theta(d \log d)$ | Step 3 of Alg. 5 |  |
| 3 | Sample $(\hat{c}, s k)$ | $\Theta(d \log d)$ | Step 4 of Alg. 5 | See below $\dagger$ |
| 4 | $\mathrm{FFT}^{-1}\left(\hat{s_{1}}\right)$ | $\Theta(d \log d)$ | Step 5 of Alg. 5 |  |
| 5 | Compress $\left(s_{1}\right)$ | $\Theta(d)$ | Step 6 of Alg. 5 | Explained in Table 2 |
| Total Complexity of Sign: |  |  |  |  |

${ }^{\dagger}$ In Sample (Algorithm 4 in [9],) there are some polynomial multiplications and additions which take $\Theta(d \log d)$ and calls PeikertSampler(Algorithm 5 in[9]) two times. In PeikerSampler, Step 1 takes $\Theta(d)$ (Generating normal vector with N -sampler takes $\Theta(d)$ and multiplying $\Sigma$ takes $\Theta(d)$ since $\Sigma$ is a diagonal matrix.). Steps 2,3 , and 5 take $\Theta(d \log d)$ since FFT computation is required. Step 4 takes $\Theta(d)$ simply since the loop iterates $d$ times.

The asymptotic complexity of verification in SOLMAE is omitted since the algorithm is identical to verification in FALCON. Our asymptotic analysis discussed here is the first step to estimate the execution time of FALCON and SOLMAE roughly. We can claim that KeyGen, Sign, Verif procedures take $\Theta(d \log d)$ together with FALCON and SOLMAE here. This analysis does imply that FALCON and SOLMAE show the same execution times regardless of its implemented platform.

## 6 Gaussian Sampler

Gaussian sampler plays a significant role in preventing quantum-secure signature schemes from secret key leakage attacks described in [13]. FALCON and SOLMAE use discrete Gaussian sampling with fixed and variable center values for efficient and secure sampling. We describe the theoretical significance of $\mathcal{N}$-Sampler (Algorithm 10 in [9]) and and the visual analysis UnifCrown sampler (Algorithm 9 in [9]) used in SOLMAE specification [9].

## 6.1 $\boldsymbol{\mathcal { N }}$-Sampler

A multivariate normal distribution is a natural distribution that is used in many fields. The process of generating a sample that follows normal distribution is called Gaussian sampling. In SOLMAE, we use an $\mathcal{N}$-Sampler (which is the same as the Gaussian sampler) to generate the FFT representation of $d$-dimensional standard normal distribution. To generate $\mathbb{R}^{d}$ vector that follows the multivariate normal distribution of mean $\mathbf{0}$ and variance matrix $\frac{d}{2} \cdot I_{d}$, we can generate independent $\frac{d}{2}$ many random vectors that follow a bivariate normal distribution with mean $\mathbf{0}$ and variance matrix $\frac{d}{2} \cdot I_{2}$, then concatenate them together. Also, sampling bivariate normal distribution with mean $\mathbf{0}$ and variance matrix $\frac{d}{2} \cdot I_{2}$ can be done by using Box-Muller transform [5]. We describe how it works: First, generate $u_{1}$ and $u_{2}$, two independent random numbers, that follow uniform distribution between 0 and 1. Then, compute $R$ and $\theta$ as shown below.

$$
\begin{equation*}
R=\sqrt{-d \cdot \ln \left(u_{1}\right)}, \quad \Theta=2 \pi \cdot u_{2} \tag{14}
\end{equation*}
$$

Finally, calculate $X$ and $Y$, to convert $(R, \Theta)$ into Cartesian coordinates.

$$
\begin{equation*}
X=R \cdot \cos (\Theta), \quad Y=R \cdot \sin (\Theta) \tag{15}
\end{equation*}
$$

Theorem 6.1. ( $X, Y$ ) in Eq.(15) follows bivariate normal distribution with mean $\mathbf{0}$ and variance matrix $\frac{d}{2} \cdot I_{2}$.
Proof. By using the random variable transform theorem stated in [10], we show that this theorem holds as follows:

$$
\begin{aligned}
p d f_{X, Y}(x, y) & =p d f_{U_{1}, U_{2}}\left(u_{1}, u_{2}\right) \cdot\left|\begin{array}{ll}
\frac{\partial u_{1}}{\partial x} & \frac{\partial u_{1}}{\partial y} \\
\frac{\partial u_{2}}{\partial x} & \frac{\partial u_{2}}{\partial y}
\end{array}\right| \\
& =\left|\begin{array}{cc}
-\frac{2 x}{d} \cdot e^{-\frac{x^{2}+y^{2}}{d}}-\frac{2 y}{d} \cdot e^{-\frac{x^{2}+y^{2}}{d}} \\
\frac{1}{2 \pi} \cdot \frac{-y}{x^{2}+y^{2}} & \frac{1}{2 \pi} \cdot \frac{x}{x^{2}+y^{2}}
\end{array}\right|\left(u_{1}=e^{-\frac{x^{2}+y^{2}}{d}}, u_{2}=\frac{1}{2 \pi} \tan ^{-1}\left(\frac{y}{x}\right)\right) \\
& =(2 \pi)^{-1} \cdot|\Sigma|^{-1 / 2} \exp \left(-\frac{1}{2} \mathbf{x}^{T} \Sigma^{-1} \mathbf{x}\right)\left(\mathbf{x}=(x, y)^{T}, \Sigma=\frac{d}{2} \cdot I_{2}\right)
\end{aligned}
$$

where pdf means the probability density function. Thus, we see that ( $X, Y$ ) in Eq.(15) follows the bivariate normal distribution.

Figure 5 shows 10 bivariate normal samplings using Box-Muller transform generated by Python script.

| -1.5600039712701361 | -1.1549240059661916 |
| ---: | ---: |
| -1.4247377976757196 | 1.002076190337793 |
| 0.9969000368756169 | -1.993812973058359 |
| 0.7107783282470497 | 0.0979834381524135 |
| -0.4516874832960174 | -0.9235298958094609 |
| 0.04449314089974015 | 1.1053117363335245 |
| -0.9864717691744923 | 0.020836466309925545 |
| 0.887687084897981 | -0.010185532828900362 |
| -1.4066801271173832 | -0.7906097922917507 |
| 0.9722996719071684 | -1.6390701046508105 |

Fig. 5: 10 bivariate normal samplings

To check whether the $\mathcal{N}$-Sampler used in SOLMAE reference implementation generates the multivariate normal distribution of mean zero and variance matrix $\frac{d}{2} \cdot I_{d}$ properly, we made a checking program in Python script that produces a sample of size 1,000 , then plots the random vectors' projections to $\mathbb{R}^{2}$ and Chi-square QQ-plot using the built-in library provided in Python. Fig. 6 illustrates the 2 -dimensional plot of this $\mathcal{N}$-Sampler. Figures 6(a) and 6(b) are its scatter plot and QQ-plot, respectively. From this experiment, we can see that this $\mathcal{N}$-Sampler works properly.


Fig. 6: Plot of $\mathcal{N}$-Sampler

### 6.2 UnifCrown Sampler

UnifCrown sampler used in SOLMAE is a method that generates a random vector that follows uniform distribution over $\Omega=\left\{(x, y) \in \mathbb{R}^{2} \mid R_{\text {min }}^{2}<x^{2}+y^{2}<R_{\text {max }}^{2}, x>0, y>0\right\}$ (i.e., the probability density function of random vector $(X, Y)$ is $f_{X, Y}(x, y)=\frac{4}{\pi \cdot\left(R_{\max }^{2}-R_{\min }^{2}\right)}$. $\left.I_{\left(R_{\text {min }}^{2}<x^{2}+y^{2}<R_{\text {max }}^{2}, x>0, y>0\right)}\right)$. With some calculations, we can easily see that if $U_{\rho}, U_{\theta}$ follows uniform distribution over $[0,1],(X, Y)=\sqrt{R_{\min }^{2}+U_{\rho}\left(R_{\max }^{2}-R_{\min }^{2}\right)}\left(\cos \left(\frac{\pi}{2} \cdot U_{\theta}\right), \sin \left(\frac{\pi}{2} \cdot U_{\theta}\right)\right)$ follows uniform distribution over $\Omega$.

To verify this implementation visually, the scatter plot of UnifCrown sampler with 10,000 samples was depicted in Fig. 7. From this, we can see that UnifCrown sampler works properly.


Fig. 7: Scatter plot of UnifCrown Sampler

## 7 Sample Execution and Performance of FALCON-512 and SOLMAE-512

Using Python implementation of SOLMAE in https://github.com/kjkim0410/SOLMAE_python_ 512 and FALCON [3], we will describe their practical execution and performance comparison here.

For your clear understanding how FALCON-512 and SOLMAE-512 operate step-by-step, we run the total execution of FALCON-512 and SOLMAE-512 once using the same 512 -byte
message and same 40-byte salt which was randomly generated by using urandom() in Python os module as below:

## Salt: b730f4e48087d8c5d6dcc085a5ad47437fd4da454c4598142b5284a794660a2cf5322d3425c631c2

 Length of Salt: 40Message: bf467a6d349c6409eba490a9ec34443ad9c009b49a0a0e71974893...d147eb98e818e600e8f6 Length of Message: 512

Each algorithm generates a set of secret keys $(f, g, F, G)$ and public key $(h)$ at first using its KeyGen procedure. Due to the space problem, we printed out the partial output of the intermediate values such as HashtoPoint, $\mathrm{s}_{0}$ signature, $\mathrm{s}_{1}$ signature, norm of signature with the allowable bound of signature. Also, the values of the message, key, and signal are too big, so they are partially expressed in this paper. The full information of executing FALCON-512 and SOLMAE-512 can refer to FALCON_512_EX2.txt and SOLMAE_512_EX2.txt, respectively at blog https://ircs.re.kr/?p=1769 for details.

### 7.1 Sample Execution of FALCON-512

The following is a partial printout of key generation, signification, and verification with FALCON512.

```
f: [-3, 1, 0, -5, 10, -5, 3, 4, 2, 4, 4, ..., 3, -3, 2, -7, 5, 2, 3, 4, -1, -2]
g: [-3, -3, 4, 5, 5, -6, 10, 1, -4, 3, ..., -9, -3, 1, -2, -7, -3, 7, -2, 0, 2]
F: [23, 10, 2, -16, 14, -26, -20, -1, ..., -7, -14, -24, -21, -23, 18, -1, 44]
G: [-2, 1, 4, 2, -25, 50, 14, 28, -19, ..., -32, -10, -2, 14, 1, -14, -12, 5]
h: [2923, 7873, 9970, 6579, 16, 10828, 337, 8243, ..., 6409, 6857, 2467, 5207]
HashToPoint: [8332, 4711, 5492, 4716, 9558, 8284, ..., 6556, 7525, 11628, 5028]
s0 Signature: [-34, 182, -7, 82, -86, 113, ..., 204, 212, 349, -65, -89, -3]
s1 Signature: [-120, -58, 30, 133, 126, 13, ..., -205, -50, 114, -502, 290, 136]
Norm of Signature : 27,222,436
Bound of Signature: 34,034,726
Signature: f8dd47a0abf43635d313e9d5a5dbb7ec2354a805d3...4200000000000000000000000
Length of Signature: 666
Verification result = True
```


### 7.2 Sample Execution of SOLMAE-512

The following is a partial printout of key generation, signification, and verification with SOLMAE512.

```
f: [0, 1, -3, -6, -3, 2, -4, 1, -1, 0, ..., 2, -3, -5, 0, -5, -5, -1, 1, -4, 3]
g: [5, 3, 5, 2, 0, 4, -10, -6, -3, 1, -2,..., 2, 4, -2, 4, -3, 3, 3, 2, 2, -3]
F: [-4, 18, 18, -36, 4, -24, 45, 42, 23, 31, ..., 3, 0, 0, 40, 42, -20, -35, 19]
```

```
G: [27, -11, 70, -13, 19, -10, -37, 20, ..., -45, 12, -3, -32, 48, 14, -16, -1]
h: [11703, 2428, 2427, 11947, 9582, 8908, 3567, ..., 6080, 7718, 3106, 11973]
HashToPoint: [8332, 4711, 5492, 4716, 9558, 8284, ..., 6556, 7525, 11628, 5028]
s0 Signature: [74, -6, 253, 187, -285, 210, ..., -225, -195, 364, 325, 138, -44]
s1 Signature: [-50, -47, 198, 13, 218, -201, ..., 114, -234, 5, -119, -41, 170]
Norm of Signature : 30,454,805
Bound of Signature: 33,870,790
Signature: 4ac35f53b674a92dd3ef7f28a9cee2bd2cc48cc8c471d1...9ac800000000000000000
Length of Signature: 666
Verification result = True
```


### 7.3 Performance Comparison of FALCON-512 and SOLMAE-512

Note that Python code is not so good tool to evaluate the exact performance of FALCON and SOLMAE. However, we can grab a rough idea of their relative performance which one can work fast. The specification of our test platform is Intel Core i7-9700 CPU at 3 GHz clock speed with 16 GRAM. We limited our experiment to the relative performance of KeyGen, Sign, and Verif procedures on FALCON-512 and SOLMAE-512 only. We executed 3 cases of each test which is executed 1,000 times iteration.

Tables 6 and 7 indicates the average time in second of KeyGen and Sign and Verif procedures of FALCON-512 and SOLMAE-512, respectively.

Table 6: Average time of KeyGen

|  | FALCON-512 | SOLMAE-512 |
| :--- | :---: | :---: |
| Test 1 | 3.6316 | 2.7346 |
| Test 2 | 3.6908 | 2.7633 |
| Test 3 | 3.7250 | 2.7306 |

Table 7: Average time of Sign and Verif

|  | Time of Sign |  | Time of Verif |  |
| :--- | :---: | :---: | :---: | :---: |
| Algo. | FALCON-512 | SOLMAE-512 | FALCON-512 | SOLMAE-512 |
| Test 1 | $6.4849 \times 10^{-2}$ | $5.9507 \times 10^{-2}$ | $5.4598 \times 10^{-3}$ | $5.4609 \times 10^{-3}$ |
| Test 2 | $6.4664 \times 10^{-2}$ | $5.9432 \times 10^{-2}$ | $5.4359 \times 10^{-3}$ | $5.3684 \times 10^{-3}$ |
| Test 3 | $6.4900 \times 10^{-2}$ | $5.8873 \times 10^{-2}$ | $5.4271 \times 10^{-3}$ | $5.3648 \times 10^{-3}$ |

Our experiments show almost indistinguishable performances between FALCON-512 and SOLMAE-512 by their Python implementation in Sign and Verif procedures while the KeyGen procedure of FALCON takes longer time than that of SOLMAE. We couldn't check the performance of FALCON-1024 and SOLMAE-1024 due to the deadlock issue of our experiment with the limited precision inherited from Python language. The SOLMAE specification [9] states that the Sign procedure of SOLMAE takes 2 times faster than that of FALCON by the reference implementation of SOLMAE in C language.

## 8 Concluding Remarks

FALCON is claimed to have the advantage of providing short public keys and signatures as well as high-security levels; plagued by a contrived signing algorithm, not very fast for signing and hard to parallelize; very little flexibility in terms of parameter settings. However, SOLMAE has a simple, fast, parallelizable signing algorithm, with flexible parameters with its novel key generation algorithm.

In this paper, after giving a brief description of the specification of FALCON and SOLMAE, we found that their asymptotic computational complexity of KeyGen, Sign and Verif procedures take $\Theta(n \log n)$ simultaneously. Also, our computer experiments using their Python implementation exhibit empirically that KeyGen of FALCON-512 only takes longer time than that of SOLMAE512 by about a second. But we can say that this is not an exact evaluation of their performance by Python implementation.

Further work such as elaborated analysis of computational complexity on FALCON and SOLMAE asymptotically is left to do next.

## Acknowledgement

This work was partially supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean Government (20ZR1300, Core Technology Research on Trust Data Connectome). Jaehyun Kim and Jueun Jung from Seoul National Unviersity, Korea have partially contributed this work while they were intern researchers at IRCS during their summer break in 2023.

## References

1. Espitau, T., Fouque, P.A., Gérard, F., Rossi, M., Takahashi, A., Tibouchi, M., Wallet, A., Yu, Y.: Mitaka: a simpler, parallelizable, maskable variant of falcon. Advances in Cryptology, Proc. of EUROCRYPTO 2022, Part III pp. 222-253 (2022) 1, 8
2. Espitau, T., Tibouchi, M., Wallet, A., Yu, Y.: Shorter hash-and-sign lattice-based signatures. Advances in Cryptology, Proc. of CRYPTO 2022, Part II pp. 245-275 (2022) 8
3. Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: Falcon: Fast-fourier lattice-based compact signatures over ntru, https://falcon-sign.info/ 5, 6, 12, 16
4. Fouque, P.A., Kirchner, P., Tibouchi, M., Wallet, A., Yu, Y.: Key recovery from gram-schmidt norm leakage in hash-and-sign signatures over ntru lattices. Cryptology ePrint Archive, Paper 2019/1180 (2019), https://eprint.iacr.org/2019/1180, https://eprint.iacr.org/2019/1180 6
5. G. E. P. Box, M.E.M.: A note on the generation of random normal deviatess. Ann. Math. Statist. pp. 610-611 (1958) 15
6. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 197-206. ACM Press (May 2008). https://doi.org/10.1145/1374376.1374407 1, 5
7. Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. Advances in Cryptology, Proc. of Crypto 1997 pp. 112-131 (1997) 5
8. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998. Lecture Notes in Computer Science, vol. 1423, pp. 267-288. Springer (1998) 5
9. Kim, K., Tibouchi, M., Espitau, T., Takashima, A., Wallet, A., Yu, Y., Guilley, S., Kim, S.: Solmae : Algorithm specification. Updated SOLMAE, IRCS Blog (2023), https://ircs.re.kr/?p=1714 7, 8, 9, 10, 14, 18
10. Kim, W.: Mathematical Statistics(in Korean). Minyoungsa, Seoul, Korea (2021) 15
11. KpqC: Korean post-quantum crytography (2020), https://kpqc.or.kr/ 1
12. Min, S., Yamamoto, G., Kim, K.: Weak property of malleability in ntrusign. Proc. of ACISP 2004, LNCS 3108 pp. 379-390 (2004) 5
13. Nguyen, P.Q., Regev, O.: Learning a parallelepiped: Cryptanalysis of ggh and ntru signatures. Journal of Cryptology 22(2), 139-160 (2009) 5, 14
14. NIST: Post-quantum crytography (2016), https://csrc.nist.gov/projects/ post-quantum-cryptography 1

Session 8-1 $\quad$ The $26^{\text {th }}$ Annual International Conference on Information Security and Cryptology
15. Pornin, T., Prest, T.: More efficient algorithms for the NTRU key generation using the field norm. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 504-533. Springer, Heidelberg (Apr 2019). https://doi.org/10.1007/978-3-030-17259-6_17 5, 10
16. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Review 41(2), 303-332 (1999) 1
17. Wikipedia: Harvest now, decrypt later (2023), https://en.wikipedia.org/wiki/Harvest_now_ decrypt_later 1

# Security Evaluation on KpqC Round 1 Lattice-based Algorithms Using Lattice Estimator 

Suhri Kim ${ }^{1}$, Eunmin Lee ${ }^{1}$, Joohee Lee ${ }^{1 \star}$, Minju Lee ${ }^{1}$, and Hyuna Noh ${ }^{1}$<br>Sungshin Women's University, Seoul, Republic of Korea<br>\{suhrikim, 20211089, jooheelee, 20211082, 20211056\}@sungshin.ac.kr


#### Abstract

Post-quantum cryptography is expected to become one of the fundamental technologies in the field of security that requires public-key cryptosystems, potentially replacing standards such as RSA and ECC, as it is designed to withstand attacks using quantum computers. In South Korea, there is an ongoing standardization effort called the KpqC (Korean Post-Quantum Cryptography) competition for developing postquantum cryptography as a national standard. The competition is in its first round, and it has introduced a total of 16 candidate algorithms for evaluation. In this paper, we analyze the security of five algorithms among the eight lattice-based schemes in the first round of the KpqC competition. We assess their security using M. Albrecht's Lattice Estimator, focusing on problems related to LWE (Learning with Errors) and LWR (Learning with Rounding). Additionally, we compare the security analysis results with the claims in the proposal documents for each algorithm. When an algorithm fails to achieve the level of security in its proposal, we suggest potential types of attacks that need to be considered for further analysis and improvement.


Keywords: Post-Quantum Cryptography, KpqC Competition, LWE, LWR

## 1 Introduction

In 1994, Peter Shor proposed polynomial-time quantum algorithms for solving discrete logarithm and factoring problems, posing a significant threat to the security of standard public-key cryptosystems such as RSA and ECC [32]. Against this backdrop, there have been active international standardization efforts for Post-Quantum Cryptography (PQC), which aims to provide new standards that are resistant to attacks using quantum computers. Since the end of 2016, the National Institute of Standards and Technology (NIST) in the United States has been conducting a standardization project for PQC in the areas of Key Encapsulation Mechanism (KEM) and digital signature. Over three rounds of evaluations,

[^26]NIST selected one KEM and three signature schemes as standards in 2022 [28]. Currently, there is an ongoing process for additional selections and evaluations in the fourth round and an on-ramp for the digital signature category [29]. Similarly, in South Korea, a national standardization competition for post-quantum cryptography, known as the KpqC competition, began in 2022 [10].

Lattice-based cryptography is a field of post-quantum cryptography that relies on the hard problems related to lattices, including NTRU [19], Learning with Errors (LWE) [31, 9], Learning with Rounding (LWR) [30], and Short Integer Solution (SIS) [1]. It has gained significant attention and recognition due to its fast computational speed and balanced performance in terms of communication overhead compared to other post-quantum cryptosystems: in the U.S. NIST standardization competition, the one selected KEM standard and two digital signature schemes out of a total of selected three post-quantum signatures are lattice-based schemes.

In the context of the KpqC competition, the lattice-based submissions in the first round include three and five schemes in the Key Encapsulation Mechanism (KEM) and digital signature categories, respectively. Each of these schemes is built upon specific underlying problems, which are summarized in Table 1.

Table 1: KpqC Competition - Round 1 Lattice-based Submissions

| Category | Algorithm | Base Problem |
| :--- | :--- | :--- |
| KEM | NTRU+ | NTRU, RLWE |
|  | SMAUG | MLWE, MLWR |
|  | TiGER | RLWR, RLWE |
| Signature | GCKSign | GCK |
|  | HAETAE | MLWE, MSIS |
|  | NCC-Sign | RLWE, RSIS |
|  | Peregrine | NTRU, RSIS |
|  | SOLMAE | NTRU, RSIS |

In this paper, we analyze the security of Learning with Errors (LWE) and Learning with Rounding (LWR) based algorithms, a total of 5 schemes (NTRU+, SMAUG, TiGER, HAETAE, NCC-Sign), among the lattice-based algorithms in the 1st round of the KpqC competition. We analyze the security of the LWE/LWR problem instances used in each algorithm. For security analysis of the LWE/LWR problems, we utilize M. Albrecht's Lattice Estimator [3]. The Lattice Estimator is an open-source tool written in Sage that quantifies specific attack complexities for various types of LWE attacks, including those described in [3]. It takes LWE (LWR) parameters as inputs and computes the attack com-
plexities along with additional parameters required for the respective attack methods.

Using the Lattice Estimator for security analysis, we derive classical security estimation results for the 5 algorithms, as shown in Table 2. For the time complexity calculation of the BKZ algorithm, we employ the Core-SVP model [4], which is consistent with the methods used in the proposal documents for 4 of the 5 algorithms, excluding NCC-Sign. In Table 2, the column 'Claimed' is the claimed security shown in the proposal documents for each algorithm, and 'Estimated' is the security that we estimated by using the Lattice Estimator. For NTRU+, we observed that the description in the specification document is different from the reference implementation which is reflected in our security estimations with respective cases. More precisely, the LWE secret, which is sampled in the encapsulation phase and denoted as $r$ in their scheme description, is sampled from $\{0,1\}^{n}$ according to the specification document (See Algorithm 6 and 9 of the NTRU+ document in [10]), while it is sampled from the centered binomial distribution in their implementation. We estimate both cases and denote the security estimation for the NTRU+ version of the specification document in parentheses. Also, for NCC-Sign, we additionally estimated the security without the Core-SVP model shown in the parentheses, since the proposal document of NCC-Sign presented the security result without using the Core-SVP model. The results are summarized as follows.

- We have observed a discrepancy between the claimed attack complexities for NTRU+ and the estimated attack complexities derived using the Lattice Estimator. For the NTRU+576, NTRU+768, and NTRU+864 parameters, we achieve 115.9 bits, 164.7 bits, and 189.2 bits, respectively, for the version of the reference implementation. These values exhibit a difference of 0.1 to 3.7 bits compared to the classical security levels claimed in the specification document. Also, larger gaps were observed between the claimed security and the estimation for the version of the specification document that utilizes the LWE with uniform binary secrets.
- For the SMAUG1280 parameters (Security level V) and TiGER256 (Security level V), classical security levels of 260.3 bits and 263 bits were claimed, but when measured using the Lattice Estimator, the attack complexities were found to be 259.2 bits and 262.0 bits, respectively.
- For HAETAE, the claimed parameters from the proposal document and the security analysis results of the Lattice Estimator are found to be similar, with an error range of less than 1 bit.
- For NCC-Sign, the proposal document presents security analysis results without using the Core-SVP model, and it is confirmed that the measured results using the Lattice Estimator were consistent. However, when measured using the Core-SVP model, it is determined that for parameters I, III, and V, the classical security levels were 123.2 bits, 190.1 bits, and 273.3 bits, respectively. This shows a difference of 18 to 24.5 bits compared to the results without the Core-SVP model in the NCC-Sign proposal document.

Table 2: Claimed vs. Estimated Security for the Round 1 Lattice-based Submissions. For NTRU+, the estimated results for the specification document version are reported in parentheses. For NCC-Sign, the estimated results without the Core-SVP model are reported in parentheses.

|  | Security Level | Claimed | Estimated |
| :--- | :--- | :--- | :--- |
|  | I $(n=576)$ | 116 | $115.9(\mathbf{1 0 8 . 9})$ |
| NTRU+ | I $(n=768)$ | 161 | $\mathbf{1 6 4 . 7}(\mathbf{1 5 6 . 5})$ |
|  | III | 188 | $\mathbf{1 8 9 . 2 ( \mathbf { 1 7 5 . 4 } )}$ |
|  | V | 264 | $263.4(\mathbf{2 4 3 . 5})$ |
| SMAUG | I | 120.0 | 120.0 |
|  | III | 180.2 | 180.2 |
|  | V | 260.3 | $\mathbf{2 5 9 . 2}$ |
| TiGER | I | 130 | 130.5 |
|  | III | 200 | $\mathbf{2 0 6 . 1}$ |
|  | V | 263 | $\mathbf{2 6 2 . 0}$ |
|  | I | 125 | 125.5 |
|  | III | 236 | 236.1 |
|  | V | 288 | 287.1 |
|  | I | 147.7 | $123.2(147.7)$ |
|  | NCC-Sign | III | 211.5 |

Based on the results, the additional attacks that each scheme needs to further consider through the Lattice Estimator are as follows:

- For SMAUG, it is confirmed that the SMAUG512 and SMAUG768 parameters achieve the claimed security levels. However, in the case of SMAUG1280, the claimed values and the estimated values differ for all attacks (usvp, bdd, bdd_hybrid, dual, dual_hybrid), which are displayed in Table 3. We remark that the displayed measurements are for the LWR instances.
- For TiGER, it is confirmed that the TiGER128 and TiGER192 parameters achieved the claimed security levels. However, for the TiGER256 parameters, the security level against dual_hybrid attacks differs between the claimed and the estimated values, which are displayed in Table 4. The displayed measurements are for the LWR instances when the dual_hybrid attack is applied.

Paper Organization This paper is structured as follows: In Chapter II, we introduce lattice-based hard problems, LWE and LWR, and provide definitions for

Table 3: Claimed vs. Estimated Classical Security for the SMAUG1280 parameter set

|  | claimed | estimated |
| :--- | :--- | :--- |
| usvp | 317.1 | 316.2 |
| bdd | 319.5 | 318.4 |
| bdd_hybrid | 290.0 | 288.5 |
| dual | 329.1 | 328.2 |
| dual_hybrid | 260.3 | 259.2 |

Table 4: Claimed vs. Estimated Classical Security for the TiGER256 parameter set

|  | claimed | estimated |
| :--- | :--- | :--- |
| dual_hybrid | $\geq 263$ | 262.0 |

KEM and digital signatures. In Chapter III, we briefly describe the key features of the KpqC 1st round candidates, including three KEM schemes and two digital signature schemes. In Chapter IV, we present the time complexity computation methods for the BKZ algorithm used in the security analysis of LWE/LWRbased algorithms and discuss the attacks covered in the Lattice Estimator. In Chapter V, we present the security analysis results obtained from the Lattice Estimator for each scheme's proposed parameters and compare them with the claimed security. Finally, in Chapter VI, we summarize the main results and conclude the paper.

## 2 Preliminaries

### 2.1 The LWE and LWR Problem

In this section, we introduce lattice-based hard problems, Learning with Errors (LWE)[31] and Learning with Rounding (LWR)[30].

### 2.1.1 LWE

Let $m, n, q$ be positive integers, $s \in \mathbb{Z}_{q}^{n}$ be a secret vector and $\chi$ be an error distribution on $\mathbb{Z}$. The LWE distribution $A_{m, n, q, \chi}^{L W E}(s)$ consisting of $m$ samples is obtained as follows: For each $i \in\{1,2, \ldots, m\}$, compute

$$
b_{i}=\left\langle\vec{a}_{i}, \vec{s}\right\rangle+e_{i} \quad \bmod q
$$

by choosing a vector $\vec{a}_{i} \in \mathbb{Z}_{q}^{n}$ uniformly and a small error $e_{i} \in \mathbb{Z}$ from the distribution $\chi$, and then output $\left\{\left(\vec{a}_{i}, b_{i}\right)\right\}_{i=1}^{m}$ as the result.

The decision LWE problem is to distinguish either given samples $\left\{\left(\vec{a}_{i}, b_{i}\right)\right\}_{i=1}^{m}$ is from the distribution $A_{m, n, q, \chi}^{L W E}$ or from the uniform distribution. The search LWE problem is to find $s \in \mathbb{Z}_{q}^{n}$, given independent samples $\left\{\left(\vec{a}_{i}, b_{i}\right)\right\}_{i=1}^{m}$ from $A_{m, n, q, \chi}^{L W E}(s)$.

Variants of LWE. Let $n$ and $q$ be positive integers and $f(x) \in \mathbb{Z}[x]$ an irreducible polynomial of degree $n$. We define a polynomial ring $\mathcal{R}=\mathbb{Z}[x] / f(x)$ and its quotient ring $\mathcal{R}_{q}=\mathbb{Z}_{q}[x] /(f(x))$ modulo $q$. The Module LWE (MLWE) problem [8] is a variant of the LWE problem defined over a module $\mathcal{R}_{q}^{k}$ for positive integers $k$. The distribution $A_{m, n, q, k, \chi}^{M L W E}(\vec{s})$ for the secret value $\vec{s} \in \mathcal{R}_{q}^{k}$ is defined as follows: For $i \in\{1,2, \ldots, m\}$, sample uniform random $\vec{a}_{i} \in \mathcal{R}_{q}^{k}$ and $e_{i} \in \mathcal{R} \leftarrow \chi^{n}$, calculate $b_{i}=\left\langle\vec{a}_{i}, \vec{s}\right\rangle+e_{i} \quad \bmod q \in \mathcal{R}_{q}$ and return the set of pairs $\left\{\left(\vec{a}_{i}, b_{i}\right)\right\}_{i=1}^{m}$ as results. It is also classified into the decision MLWE and search MLWE problems as in the LWE problem. For the specific case of MLWE when the dimension of module $k$ is 1 , we call it as Ring-LWE (RLWE) problem [25].

### 2.1.2 LWR

The LWR problem introduced by Banerjee et al. [6] obfuscates the secret by applying a deterministic rounding procedure $(\lfloor\cdot\rceil)$ to linear equations instead of adding errors sampled from discrete Gaussian distributions. Given positive integers $m, n, q, p$, let $\vec{s} \in \mathbb{Z}_{q}^{n}$ be an $n$-dimensional secret vector. The LWR distribution $A_{m, n, q, p}^{L W R}(\vec{s})$ over $\mathbb{Z}_{q}^{m \times n} \times \mathbb{Z}_{p}^{m}$ consisting of $m$ samples is obtained as follows : For $i \in\{1,2, \ldots, m\}$, compute $b_{i}=\left\lfloor(p / q) \cdot\left(\left\langle\vec{a}_{i}, \vec{s}\right\rangle \bmod q\right)\right\rceil$ where $\vec{a}_{i} \in \mathbb{Z}_{q}^{n}$ is uniformly sampled, and return the set of pairs $\left\{\left(\vec{a}_{i}, b_{i}\right)\right\}_{i=1}^{m}$. The decision LWR problem is to distinguish either given samples $\left\{\left(\vec{a}_{i}, b_{i}\right)\right\}_{i=1}^{m}$ is from the distribution $A_{m, n, q, \chi}^{L W R}$ or from the uniform distribution. The search LWR problem is to find $\vec{s} \in \mathbb{Z}_{q}^{n}$, given independent samples $\left\{\left(\vec{a}_{i}, b_{i}\right)\right\}_{i=1}^{m}$ from $A_{m, n, q, p}^{L W R}(\vec{s})$. This definition can be extended to Ring-LWR (RLWR) and Module-LWR (MLWR) by using vectors of polynomials as in the LWE problem.

### 2.2 The Round 1 LWE/LWR-based Candidates

### 2.2.1 KEM

A Key Encapsulation Mechanism (KEM) is a triple of algorithms, $\Pi=$ (KeyGen, Encaps, Decaps), where
$-(p k, s k) \leftarrow \operatorname{KeyGen}\left(1^{\lambda}\right):$ The key generation algorithm takes security parameter $\lambda>0$ as an input and then outputs the pair of public key and private key $(p k, s k)$.
$-(c, K) \leftarrow \operatorname{Encaps}(p k):$ The encapsulation algorithm takes the public key $p k$ as an input and then outputs a pair of secret key $K$ and ciphertext $c$.

- $(K$ or $\perp) \leftarrow \operatorname{Decaps}(s k, c)$ : The decapsulation algorithm takes the private key $s k$ and the ciphertext $c$ as input, and then outputs the shared key $K$ or $\perp$.

For correctness, it is required that, for all $(p k, s k) \leftarrow \operatorname{KeyGen}\left(1^{\lambda}\right)$ and for all $(c, K) \leftarrow \operatorname{Encaps}(p k)$, Decaps $(s k, c)=K$ holds. In this section, we review the distinguished features of the KpqC Round 1 lattice-based KEMs NTRU+, SMAUG, and TiGER.

NTRU+. NTRU+ is an algorithm that improves the efficiency of the existing NTRU scheme [19]. It follows the strategy to construct NTT (Number Theory Transform)-friendly settings for NTRU which has been introduced in NTTRU [26] and NTRU-B [17]. The security of NTRU+ is based on the NTRU and RLWE problems. The main features are as follows:

- NTRU+ utilizes the NTT-friendly polynomial rings $\mathcal{R}_{q}=\mathbb{Z}_{q}[x] /(f(x))$, where $f(x)=x^{n}-x^{n / 2}+1$ is a cyclotomic trinomial of degree $n=2^{i} 3^{j}$, and adapt NTT in all computations.
- In the encapsulation and decapsulation, new methods for secret key encoding (SOTP) and decoding (Inv) were proposed. The SOTP and Inv operations for $m \in\{0,1\}^{n}, u=\left(u_{1}, u_{2}\right) \in\{0,1\}^{2 n}$, and $y \in\{-1,0,1\}^{n}$ are designed as follows.

$$
\begin{align*}
\operatorname{SOTP}(m, u) & =\left(m \oplus u_{1}\right)-u_{2} \in\{-1,0,1\}^{n}  \tag{1}\\
\operatorname{Inv}(y, u) & =\left(y+u_{2}\right) \oplus u_{1} \tag{2}
\end{align*}
$$

One can easily check $\operatorname{lnv}(\operatorname{SOTP}(m, u), u)=m$.

- To satisfy IND-CCA (Indistinguishability against adaptive Chosen-Ciphertext Attacks) security, NTRU + applies a modified transform of the conventional Fujisaki-Okamoto (FO) transform [18]. The difference is that the decapsulation procedures require re-encryption when applying the FO transform, while NTRU+ removes the re-encryption in the decapsulation by recovering the random polynomial (denoted by $r$ in their scheme) used in the encapsulation twice and then comparing between them.

SMAUG. SMAUG is designed based on the hardness of MLWE and MLWR problems, both of which utilize the sparse ternary secrets following the approaches in Lizard [14] and RLizard [21]. The main features are as follows.

- SMAUG KEM is obtained by first constructing an IND-CPA (Indistinguishability against Chosen-Plaintext Attacks) secure public-key encryption (PKE) scheme and then applying the FO transform [18] on it to achieve the INDCCA security.
- The secret keys for MLWE and MLWR are sampled as sparse ternary vectors with fixed Hamming weights, respectively.
- The moduli $q$ and $p$ are set to powers of 2 in order to replace the rounding operations in the encapsulation with bit-wise shift operations.

TiGER. TiGER is designed based on the RLWE and RLWR problems with sparse secrets. The main features are as follows.

- TiGER consists of an IND-CPA PKE scheme, and an IND-CCA KEM obtained by applying the FO transform to it.
- All integer modulus in the scheme are set to be power of 2 for the same reason as in SMAUG, in order to replace the rounding operations with bitwise shifts.
- TiGER pre-defines the Hamming weight of the secrets of RLWE and RLWR and generates sparse vectors. Additionally, the errors for RLWE are also sampled as sparse vectors.
- The sizes of ciphertexts and public keys are relatively small because of using a small modulus of 1 byte $(q=256)$ for all suggested parameters.
- When encoding the secret key in TiGER KEM, they employ an Error Correcting Code (ECC) to reduce decryption failure rates. Therefore, it is possible to adjust the decryption failure rate to be negligible in the security parameter, despite using the small modulus $q$. They utilize XEf [5], D2 [4] for the ECC methods.


### 2.2.2 Digital Signatures

Digital signatures is a triple of algorithms $\Pi=$ (KeyGen, Sign, Verify). The key generation (KeyGen) algorithm generates a pair of a public key and a private key. The signing (Sign) algorithm takes the private key and a message as inputs to generate a signature. The verification (Verify) algorithm takes the public key, message, and signature value as inputs to verify the validity of the signatures. These can be summarized as follows:
$-(p k, s k) \leftarrow \operatorname{Key} \operatorname{Gen}\left(1^{\lambda}\right):$ The key generation algorithm takes security parameter $\lambda$ as an input and then outputs a pair of public key and private key ( $p k, s k$ ).
$-\sigma \leftarrow \operatorname{Sign}(s k, m)$ : The signature algorithm takes the private key $s k$ and a message $m$ as inputs and then outputs a signature $\sigma$.

- 1 or $0 \leftarrow \operatorname{Verify}(p k, m, \sigma)$ : The verification algorithm takes the public key $p k$, a message $m$, and a signature $\sigma$ as inputs. It outputs 1 if the signature is valid, and 0 otherwise.

In this section, we summarize the distinguished features of the KpqC Round 1 lattice-based signature schemes HAETAE and NCC-Sign.

HAETAE. HAETAE utilizes the Fiat-Shamir with Aborts paradigm [23, 24] as in the CRYSTALS-Dilithium [16], one of the standards selected in the NIST PQC standardization project. HAETAE uses a bimodal distribution proposed in the rejection sampling of BLISS signatures [15]. The main features are as follows:

- In lattice-based digital signature algorithms, the distribution used for rejection sampling has a significant impact on the signature size. HAETAE uses
a hyperball uniform distribution to reduce the signature size, albeit at the cost of speed compared to Dilithium.
- HAETAE leverages a module structure and uses a predefined polynomial ring $\mathcal{R}_{q}=\mathbb{Z}_{q}[x] /\left(x^{256}+1\right)$ for all parameter sets, making it easy to adjust parameters according to the required security level.

NCC-Sign. NCC-Sign is a digital signature algorithm that combines the design rationale of CRYSTALS-Dilithium and NTRU prime [7], which were also round 3, 4 candidates for NIST PQC standardization project KEM algorithms. NCC-Sign also adopts Fiat-Shamir with Aborts paradigm as in HAETAE and Dilithium, but instead of using a cyclotomic polynomial ring of $\mathcal{R}_{q}=\mathbb{Z}_{q}[x] /\left(x^{n}+\right.$ 1), it uses the non-cyclotomic polynomial ring of the form $\mathcal{R}_{q}=\mathbb{Z}_{q}[x] /\left(x^{p}+x+1\right)$, where $p$ is a prime. The main features are as follows:

- Due to the use of a non-cyclotomic ring, NTT cannot be applied to polynomial multiplications. In NCC-Sign, polynomial multiplication is computed using the Toom-Cook method, one of the divide-and-conquer techniques. For a prime $p$ such that $p \leq 4 n, n \in \mathbb{Z}$, the algorithm computes polynomial multiplication of degree $4 n$ and exploits Toom-Cook-4-way and Karatsuba multiplication.


## 3 Security Analysis Methods

### 3.1 Time complexity Estimation of the BKZ algorithm

The BKZ algorithm [12] is a state-of-the-art lattice basis reduction algorithm used to find short bases within a given lattice, and it exhibits exponential time complexity. To analyze the security of the LWE/LWR-based algorithms, the instances of LWE/LWR used in the algorithms are induced to the problems to find short vectors in lattices which are given by the choices of attack strategies such as Dual and Primal attacks. Hence, it can be solved by using the BKZ algorithm.

The core idea behind the BKZ algorithm is to iteratively apply a Shortest Vector Problem (SVP) solver to sub-lattices of dimension smaller than the original lattice. When the dimension of the sub-lattice to which the SVP solver is applied is $\beta>0$, it is referred to as $\beta$-BKZ, and this sub-lattice is called a 'block'.

The Core-SVP model [4] is a measurement model used to estimate the time complexity of the BKZ algorithm from a conservative perspective. When calculating the time complexity of the BKZ algorithm using the Core-SVP model, the time complexity of $\beta$-BKZ is estimated to be $2^{c \cdot \beta}$, which is a lower bound of the time complexity of a single application of the SVP solver $\left(2^{c \cdot \beta+o(\beta)}\right)$, where $c \in[0,1]$ is constant. This conservative model is designed to ensure that the security predictions of the BKZ algorithm remain unaffected by improvements in the efficiency of either the number of iterations of applying the SVP solver or
the efficiency of the SVP solver itself, thus preserving the algorithm's security guarantees.

In the Core-SVP model, the constant $c \in[0,1]$ used for calculating the BKZ time complexity is determined based on the efficiency of the SVP solver. In [4], it was employed as shown in Table 5. For quantum SVP solvers, continuous improvements in efficiency have led to the existence of algorithms with $c_{Q}=$ 0.257 [11]. In this paper, we calculate the BKZ time complexity using $c=0.292$ for classical security. When using the Core-SVP model, quantum security (in bits) can be simply estimated by multiplying classical security (in bits) with $c_{Q} / 0.292$.

Table 5: The BKZ time complexity $(T)$ for classical security and quantum security in the Core-SVP model

|  | classical | quantum |
| :--- | :--- | :--- |
| $c$ | 0.292 | 0.265 |
| $T$ | $2^{0.292 \beta}$ | $2^{0.265 \beta}$ |

### 3.2 Dual Attack

The dual attack identifies a short vector $v$ that is orthogonal to matrix $A$. Given $(A, \vec{b}) \in \mathbb{Z}_{q}^{k \times l} \times \mathbb{Z}_{q}^{k}$ either from the LWE distribution or the uniform distribution, a lattice $\Lambda_{m}^{\text {dual }}$ can be defined as follow. Let $A_{[m]}$ be the upmost $m \times l$ sub-matrix of $A$ for $m \leq k$.

$$
\Lambda_{m}^{\text {dual }}:=\left\{(\vec{u}, \vec{v}) \in \mathbb{Z}^{m} \times \mathbb{Z}^{l}: A_{[m]}^{\top} \vec{u}+\vec{v}=0 \bmod q\right\}
$$

If it is the case $\vec{b}=A \vec{s}+\vec{e}$, with a short non-zero element $(\vec{u}, \vec{v})$, an attacker can compute $\left\langle\vec{u}, \vec{b}_{[m]}\right\rangle=-\langle\vec{v}, \vec{s}\rangle+\left\langle\vec{u}, \vec{e}_{[m]}\right\rangle$, where $\vec{b}_{[m]}$ and $\vec{e}_{[m]}$ are the upmost $m$-dimensional sub-vector of $\vec{b}$. Hence, the attacker can determine it is an LWE instance if $\left\langle\vec{u}, \vec{b}_{[m]}\right\rangle$ is short enough. Therefore, finding a sufficiently short nonzero vector in the lattice $\Lambda_{m}^{\text {dual }}$ implies solving the decision-LWE problem. To find a short lattice element of $\Lambda_{m}^{\text {dual }}$, the attack employs the $\beta$-BKZ lattice basis reduction algorithm.

### 3.3 Primal Attack

The primal attack on LWE addresses the bounded distance decoding (BDD) problem directly. In other words, when provided with LWE samples $(A, b)$, it
seeks a vector $w=A s$ such that $\|b-w\|$ is unusually small. There are two main strategies to solve BDD: the first strategy is to utilize Babai's nearest algorithm with lattice basis reduction [22], and the second is to reduce BDD problem into unique-SVP (uSVP) problem and solve it using the lattice basis reduction algorithms [2, 4]. Here, we will elaborate on the second method, which is more widely considered.

Given an LWE instance $(A, b=A s+e) \in \mathbb{Z}_{q}^{m \times n} \times \mathbb{Z}_{q}^{m}$, a lattice $\Lambda_{m}$ can be defined as follow. $B=\left(A_{[m]}\left|I_{m}\right| b_{[m]}\right) \in \mathbb{Z}_{q}^{m \times(n+m+1)}$.

$$
\Lambda_{m}=\left\{v \in \mathbb{Z}_{q}^{n+m+1}: B v \bmod q\right\}
$$

Therefore, a short non-zero vector in the lattice $\Lambda_{m}$ can be transformed into the non-trivial solutions for the LWE equation. This attack utilizes the $\beta$-BKZ algorithm to find the sufficiently short vector in the lattice $\Lambda_{m}$.

### 3.4 Hybrid Attack

An attack that combines techniques, such as meet-in-the-middle, with either Primal or Dual attacks is known as a hybrid attack. Hybrid attacks are generally not as efficient as Primal or Dual attacks, but they can be effective in cases where the secret key in LWE follows a specialized distribution. In [20], by incorporating lattice reduction techniques and implementing a meet-in-the-middle (MITM) strategy, it is possible to diminish the complexity of the attack on the NTRUEncrypt private key from $2^{84.2}$ to $2^{60.3}$ for the parameter set for 80 -bit security. Also, Jung Hee Cheon et al. [13] introduced a hybrid attack strategy that integrates dual lattice attacks with the MITM approach. This approach involves increasing the error size while simultaneously reducing the dimension and Hamming weights of the secret vector. As the MITM attack cost is strongly correlated with the dimension of the secret vector but less affected by error size, this trade-off significantly reduces the overall cost of the MITM attack when applying it to the LWE with sparse secrets.

## 4 KpqC Round 1 LWE/LWR-based algorithms Security analysis

### 4.1 Parameters

In this section, we summarize the proposed parameters used in the underlying LWE/LWR instances in the respective schemes. For simplicity, we use the same notations as in the original specification documents.

NTRU + Parameters. NTRU+ is based on NTRU and RLWE, and the proposed parameters used to analyze attack complexities of RLWE are as shown in Table 6. They use the quotient ring $\mathcal{R}_{q}=\mathbb{Z}[X] /\left(X^{n}-X^{n / 2}+1\right)$ for dimension $n=2^{i} 3^{j}$ and fixed modulus $q=3457$ for all parameters. For the RLWE
secret distribution and error distribution, they utilize the uniform distribution on $(0,1)$ and the centered binomial distribution in their specification document and reference implementation, respectively.

Table 6: NTRU + Proposed parameter sets

|  | 576 | 768 | 864 | 1152 |
| :--- | :--- | :--- | :--- | :--- |
| $n$ | 576 | 768 | 864 | 1152 |
| $q$ | 3457 | 3457 | 3457 | 3457 |
| security level | I | I | III | V |

SMAUG Parameters. SMAUG is based on MLWE/MLWR, and the parameters used for the attack on MLWE/MLWR are as shown in Table 7. They use the quotient ring $\mathcal{R}_{q}=\mathbb{Z}_{q}[X] /\left(X^{n}+1\right)$ for power of 2 integer $n$ and positive integer $q$. The secret keys for each LWE and LWR instance, denoted as $s$ and $r$ are sampled as sparse vectors with fixed Hamming weights, where the Hamming weights are denoted as $h_{s}, h_{r}$, respectively. $\sigma$ is the standard deviation of the discrete Gaussian distribution to sample the errors in LWE.

Table 7: SMAUG Proposed parameter sets

|  | SMAUG128 | SMAUG192 | SMAUG256 |
| :--- | :--- | :--- | :--- |
| $n$ | 512 | 768 | 1280 |
| $m$ | 512 | 768 | 1280 |
| $q$ | 1024 | 1024 | 1024 |
| $p$ | 256 | 256 | 256 |
| $h_{r}$ | 132 | 147 | 140 |
| $h_{s}$ | 140 | 150 | 145 |
| $\sigma$ | 1.0625 | 1.0625 | 1.0625 |
| security level | I | III | V |

TiGER Parameters. TiGER is based on RLWR/RLWE and the parameters used for the attack are as shown in Table 8. They use the quotient ring $\mathcal{R}_{q}=$ $\mathbb{Z}_{q}[X] /\left(X^{n}+1\right)$ for a power of 2 integer $n$ and a positive integer $q$. $k_{1}$ and $k_{2}$ are power of 2 's and represents the modulus used for ciphertext compression.
$h_{s}$ and $h_{r}$ are the Hamming weights of the secret key and the ephemeral secret used for encapsulation. $h_{e}$ is the Hamming weight of the LWE error.

Table 8: TiGER Proposed parameter sets

|  | TiGER128 | TiGER192 | TiGER256 |
| :--- | :--- | :--- | :--- |
| $n$ | 512 | 1024 | 1024 |
| $m$ | 512 | 1024 | 1024 |
| $q$ | 256 | 256 | 256 |
| $p$ | 128 | 64 | 128 |
| $h_{r}$ | 128 | 84 | 198 |
| $h_{s}$ | 160 | 84 | 198 |
| $h_{e}$ | 32 | 84 | 32 |
| $k_{1}$ | 64 | 64 | 128 |
| $k_{2}$ | 64 | 4 | 4 |
| security level | I | III | V |

HAETAE Parameters. HAETAE is based on MLWE/MSIS and the parameters used for the attack are as shown in Table 9. They use the quotient ring $\mathcal{R}_{q}=\mathbb{Z}_{q}[X] /\left(X^{n}+1\right)$ for positive integers $n$ and $q$ which are set to 256 and 64513 , respectively, for all parameter sets. $(k, \ell)$ denotes the matrix size of the module structure. They select the private key from the uniform distribution over $[-\eta, \eta]$, and $\tau$ refers to the Hamming weight of the binary challenge.

Table 9: HAETAE Proposed parameter sets

|  | HAETAE120 | HAETAE180 | HAETAE260 |
| :--- | :--- | :--- | :--- |
| $n$ | 256 | 256 | 256 |
| $q$ | 64513 | 64513 | 64513 |
| $(k, \ell)$ | $(2,4)$ | $(3,6)$ | $(4,7)$ |
| $\eta$ | 1 | 1 | 1 |
| $\tau$ | 39 | 49 | 60 |
| security level | I | III | V |

NCC-Sign Parameters. NCC-Sign is based on RLWE/RSIS and the parameters used for the attack are shown in Table 10. They use the ring $\mathcal{R}_{q}=$ $\mathbb{Z}_{q}[X] /\left(X^{p}-X-1\right)$ for prime numbers $p$ and $q$. Also, they select the private key from the distribution over $[-\eta, \eta]$, and $\tau$ refers to the number of nonzero coefficients in $\{-1,0,1\}$.

Table 10: NCC-Sign Proposed parameter sets

|  | I | III | V |
| :--- | :--- | :--- | :--- |
| $p$ | 1021 | 1429 | 1913 |
| $q$ | 8339581 | 8376649 | 8343469 |
| $\eta$ | 2 | 2 | 2 |
| $\tau$ | 25 | 29 | 32 |
| security level | I | III | V |

### 4.2 Analysis using the Lattice Estimator

In this section, we report our estimated results for the lattice attacks in [3] outlined in Section 3. The results of security analysis using the Lattice Estimator for NTRU+, SMAUG, TiGER, HAETAE, and NCC-Sign schemes are shown in Table 11, Table 12, Table 13, Table 14, Table 15, Table 16, and Table 17.

The column names in each table, "sec" and " $\beta$ " represent classical security in bits and BKZ block size respectively. For the BKZ time complexity estimation, we use the Core-SVP model except Table 17. Among the row names in each table, "usvp" refers to the attack complexity for the Primal attack described in Section 3.3, and bdd, bdd_hybrid, bdd_mitm_hybrid attacks are variations of the Primal attack. Also, "dual" means the attack complexity for the Dual attack explained in Section 3.2, and dual_hybrid, dual_mitm_hybrid are variations of the Dual attack. For more details about the attacks, we recommend to see [3]. We remark that when analyzing the security of SMAUG and TiGER, we measured attack complexities for both LWE and LWR instances, and reported the minimum value. In the case of NTRU+, since it does not use a sparse secret key in the LWE instance, during the security analysis, we did not measure the attack complexities for bdd_mitm_hybrid and dual_mitm_hybrid, which are expected to be less efficient compared to other attacks.

In the case of NTRU+, Table 11 shows dual_hybrid has the smallest attack complexity. In Table 12, overall attack complexities have increased, and usvp has the smallest complexity. In the case of SMAUG, according to Table 13, the most effective attack differs for each parameter set: the most effective attack for SMAUG128 is usvp, dual hybrid for SMAUG192, and dual_hybrid for SMAUG256. In the case of TiGER, as listed in Table 14, TiGER128 exhibits
the smallest complexity for Primal attack usvp. For TiGER192 and TiGER256, dual_hybrid is the most effective method.

In the case of HAETAE, in Table 15, for the claimed security of 120 bits and 260 bits, the most effective attack method is dual_hybrid followed by usvp. For the security of 180 bits, usvp has the smallest attack complexity. In the case of NCC-Sign, Table 16 and Table 17 show similar results. In Table 16, usvp is confirmed to have the smallest attack complexity, while in Table 17, bdd exhibits the smallest attack complexity.

Table 11: NTRU+ Security Estimation

|  | 576 |  | 768 |  |  | 864 |  | 1152 |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :---: |
|  | sec | $\beta$ | sec | $\beta$ | sec | $\beta$ | sec | $\beta$ |  |
| usvp | 109.8 | 376 | $\mathbf{1 5 6 . 5}$ | 536 | 180.2 | 617 | 252.9 | 866 |  |
| bdd | 110.8 | 375 | 157.4 | 535 | 181.0 | 617 | 253.7 | 865 |  |
| bdd_hybrid | 111.0 | 375 | 157.4 | 535 | 181.2 | 617 | 316.1 | 864 |  |
| dual | 114.8 | 393 | 162.4 | 556 | 186.9 | 640 | 261.3 | 895 |  |
| dual_hybrid | 108.9 | 372 | 153.0 | 523 | $\mathbf{1 7 5 . 4}$ | 599 | $\mathbf{2 4 3 . 5}$ | 833 |  |

Table 12: NTRU+ Security Estimation _ rev

|  | 576 |  | 768 |  | 864 |  | 1152 |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
|  | sec | $\beta$ | sec | $\beta$ | sec | $\beta$ | sec | $\beta$ |
| usvp | $\mathbf{1 1 5 . 9}$ | 397 | 164.7 | 564 | 189.8 | 650 | 266.0 | 911 |
| bdd | 116.9 | 397 | 165.7 | 563 | 190.7 | 649 | 266.9 | 911 |
| bdd_hybrid | 193.2 | 397 | 264.1 | 563 | 300.0 | 649 | 408.9 | 911 |
| dual | 120.9 | 414 | 171.1 | 586 | 196.5 | 673 | 274.8 | 941 |
| dual_hybrid | 117.2 | 400 | 164.9 | 564 | $\mathbf{1 8 9 . 2}$ | 647 | $\mathbf{2 6 3 . 4}$ | 901 |

Table 13: SMAUG Security Estimation

|  | 128 |  | 192 |  | 256 |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
|  | sec | $\beta$ | sec | $\beta$ | sec | $\beta$ |
|  | $\mathbf{1 2 0 . 0}$ | 411 | 187.2 | 641 | 316.2 | 1083 |
|  | 120.9 | 411 | 188.5 | 642 | 318.4 | 1090 |
| bdd_hybrid | 121.3 | 411 | 189.0 | 642 | 288.5 | 674 |
| bdd_mitm_hybrid | 166.5 | 410 | 221.0 | 496 | 277.8 | 680 |
| dual | 125.9 | 431 | 195.3 | 669 | 328.2 | 1124 |
| dual_hybrid | 122.7 | 399 | $\mathbf{1 8 0 . 2}$ | 575 | $\mathbf{2 5 9 . 2}$ | 749 |

Table 14: TiGER Security Estimation

|  | 128 |  | 192 |  | 256 |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
|  | sec | $\beta$ | sec | $\beta$ | sec | $\beta$ |
|  | $\mathbf{1 3 0 . 5}$ | 447 | 277.4 | 950 | 279.7 | 958 |
|  | 131.4 | 445 | 281.5 | 964 | 280.7 | 958 |
| bdd_hybrid | 131.4 | 445 | 220.2 | 472 | 280.7 | 958 |
| bdd_mitm_hybrid | 173.8 | 419 | 212.7 | 503 | 316.5 | 730 |
| dual | 137.5 | 471 | 290.5 | 995 | 291.7 | 999 |
| dual_hybrid | 131.9 | 428 | 206.1 | 535 | $\mathbf{2 6 2 . 0}$ | 835 |

Table 15: HAETAE Security Estimation

|  | 120 |  | 180 |  | 260 |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
|  | sec | $\beta$ | sec | $\beta$ | sec | $\beta$ |
|  | $\mathbf{1 2 5 . 6}$ | 430 | 238.0 | 815 | 290.2 | 994 |
| bdd | 126.6 | 429 | 238.8 | 815 | 291.1 | 993 |
| bdd_hybrid | 126.6 | 429 | 238.8 | 815 | 291.1 | 993 |
| bdd_mitm_hybrid | 219.3 | 429 | 390.9 | 815 | 472.7 | 993 |
| dual | 130.5 | 447 | 245.6 | 841 | 298.4 | 1022 |
| dual_hybrid | 126.4 | 432 | $\mathbf{2 3 6 . 1}$ | 808 | $\mathbf{2 8 7 . 1}$ | 982 |

Table 16: NCC-Sign Security Estimation with the Core-SVP model

|  | 1 |  | 3 |  | 5 |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
|  | sec | $\beta$ | sec | $\beta$ | sec | $\beta$ |
| usvp | $\mathbf{1 2 3 . 2}$ | 422 | 190.1 | 651 | $\mathbf{2 7 3 . 3}$ | 936 |
| bdd | 124.6 | 421 | 191.0 | 651 | 274.3 | 935 |
| bdd_hybrid | 124.6 | 421 | 191.0 | 651 | 274.3 | 935 |
| bdd_mitm_hybrid | 270.0 | 421 | 406.1 | 651 | 588.6 | 935 |
| dual | 126.4 | 433 | 194.2 | 665 | 278.6 | 954 |
| dual_hybrid | 124.8 | 427 | 191.1 | 654 | 273.6 | 937 |

Table 17: NCC-Sign Security Estimation without the Core-SVP model as they evaluated in the Round 1 Proposal (less conservative)

|  | 1 |  | 3 |  | 5 |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
|  | sec | $\beta$ | $\sec$ | $\beta$ | sec | $\beta$ |
| usvp | 149.7 | 422 | 213.9 | 651 | 294,0 | 936 |
| bdd | $\mathbf{1 4 7 . 7}$ | 413 | 211.5 | 641 | 291.3 | 924 |
| bdd_hybrid | $\mathbf{1 4 7 . 7}$ | 413 | 211.5 | 641 | 291.3 | 924 |
| bdd_mitm_hybrid | 261.8 | 421 | 394.9 | 651 | 574.2 | 935 |
| dual | 153.8 | 433 | 219.6 | 668 | 302.4 | 962 |
| dual_hybrid | 150.5 | 421 | 214.9 | 651 | 295.5 | 937 |

### 4.3 Comparisons with the Claimed Security

We present the comparison of the claimed vs. estimated (classical) security in bits for each scheme in Fig. 1a, Fig. 1b, Fig. 1c, Fig. 1d, and Fig. 1e.

For NTRU+, we measured the security based on both the specification document and the implementation. The result from the implementation was similar to the claimed security in the proposal document. However, the result based on the specification document indicated lower security than the implementation result. The reason for these different results occurred from the process of sampling the secret ' $r$ ' value in the LWE instances using the $H$ function in the Encaps algorithm in NTRU+ (See Algorithm 6 and 9 in the NTRU + specification document). The specification samples the secret ' $r$ ' with uniform binary values, however, the implementation samples it with ternary values following the centered binomial distribution.

(a) Comparison of the claimed security and estimated results in which estimated results are measured for the versions of specification and implementation for NTRU+, respectively

(c) Comparison of the claimed security and estimated results for TiGER parameters

(b) Comparson of the claimed security and estimated results for SMAUG parameters

(d) Comparison of the claimed security and estimated results for HAETAE parameters

(e) Comparison of the claimed security and estimated results with Core-SVP and without Core-SVP for NCC-Sign parameters

The differences between the analysis by using the Lattice Estimator and the analysis presented in the proposal document can be summarized as follows.

- For the SMAUG1280 parameters, the claimed security in the proposal document is of 260.3 bits, but the estimated result using Lattice Estimator resulted in an attack amount of 259.2 bits.
- In the case of TiGER256(Security level V), the classical security of 263 bits was claimed, but the estimated result was 262.0 bits.
- The estimated results of NTRU+ were found different from the claimed attack complexities for all parameters. For the NTRU+576, NTRU+768, NTRU +864 , and NTRU +1152 parameters, they each satisfy classical security levels of 115.9 bits, 164.7 bits, 189.2 bits, and 263.4 bits, respectively, for the implementation version. These values differ by 0.1 to 3.7 bits from the classical security levels claimed in the proposal document, which were 116 bits, 161 bits, 188 bits, and 264 bits. For the document version of NTRU+ using LWE with uniform binary secrets, the gaps between the claimed and estimated security get larger.
- For HAETAE, the result claimed in the proposal document and security analysis results were similar about all parameters, with an error range of less than 1 bit.
- In the case of NCC-Sign, the proposal document provided results of security analysis without using the Core-SVP model and the estimations using the Lattice Estimator were found to match these results. When we measured using the Core-SVP model, it was observed that parameters I, III, and V achieve classical security levels of 123.2 bits, 190.1 bits, and 273.3 bits, respectively. This represents differences of 18 to 24.5 bits compared to the results presented in the NCC-Sign proposal document.


## 5 Conclusion

In this paper, we discussed the results of a security analysis using the Lattice Estimator for five Round 1 lattice-based candidates proposed in the KpqC Competition. It was found that NTRU+ had differences of approximately 0.1 to 3.7 bit compared to the claimed results of security analysis for all parameters when using the centered binomial distribution as a secret distribution in LWE. For SMAUG and TiGER, the classical security of parameters in the security level V was observed to differ by approximately 1 bit from the estimated results. In the case of HAETAE and NCC-Sign, we confirmed that the claimed parameters are closely similar to the security analysis results. We also remark that the Lattice Estimator does not exhaustively cover all recent attacks for LWE including [27]. We will analyze the KpqC Round 1 lattice-based schemes further by applying various recent LWE attacks for future works.

Acknowledgement. This work is the result of commissioned research project supported by the affiliated institute of ETRI [2023-080].

## References

[1] Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing. p. 99-108. STOC '96, Association for Computing Machinery, New York, NY, USA (1996). https://doi.org/10.1145/237814.237838, https://doi.org/10. 1145/237814. 237838
[2] Albrecht, M.R., Göpfert, F., Virdia, F., Wunderer, T.: Revisiting the expected cost of solving usvp and applications to lwe. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017. pp. 297-322. Springer International Publishing, Cham (2017)
[3] Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Cryptology ePrint Archive, Paper 2015/046 (2015), https://eprint.iacr. org/2015/046, https://eprint.iacr.org/2015/046
[4] Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key Exchange-A new hope. In: 25th USENIX Security Symposium (USENIX Security 16). pp. 327-343. USENIX Association, Austin, TX (Aug 2016), https://www.usenix.org/conference/usenixsecurity16/ technical-sessions/presentation/alkim
[5] Baan, H., Bhattacharya, S., Fluhrer, S., Garcia-Morchon, O., Laarhoven, T., Rietman, R., Saarinen, M.J., Tolhuizen, L., Zhang, Z.: Round5: Compact and Fast Post-quantum Public-Key Encryption, pp. 83-102 (07 2019). https://doi.org/10.1007/978-3-030-25510-7_5
[6] Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 719-737. Springer (2012)
[7] Bernstein, D.J., Chuengsatiansup, C., Lange, T., van Vredendaal, C.: Ntru prime: reducing attack surface at low cost. Cryptology ePrint Archive, Paper 2016/461 (2016), https://eprint.iacr.org/2016/461, https://eprint.iacr.org/2016/ 461
[8] Boudgoust, K., Jeudy, C., Roux-Langlois, A., Wen, W.: On the hardness of module-lwe with binary secret. In: Cryptographers' Track at the RSA Conference. pp. 503-526. Springer (2021)
[9] Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors (2013)
[10] Center, K.R.: Kpqc competition round 1, available from: https://www.kpqc.or. $\mathrm{kr} /$ competition.html [last accessed June 2023]
[11] Chailloux, A., Loyer, J.: Lattice sieving via quantum random walks (2021)
[12] Chen, Y., Nguyen, P.Q.: Bkz 2.0: Better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology - ASIACRYPT 2011. pp. 1-20. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
[13] Cheon, J.H., Hhan, M., Hong, S., Son, Y.: A hybrid of dual and meet-in-themiddle attack on sparse and ternary secret lwe. IEEE Access 7, 89497-89506 (2019). https://doi.org/10.1109/ACCESS.2019.2925425
[14] Cheon, J.H., Kim, D., Lee, J., Song, Y.: Lizard: Cut off the tail! a practical postquantum public-key encryption from lwe and lwr. In: International Conference on Security and Cryptography for Networks. pp. 160-177. Springer (2018)
[15] Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal gaussians. In: Annual Cryptology Conference. pp. 40-56. Springer (2013)
[16] Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: A lattice-based digital signature scheme. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 238-268 (2018)
[17] Duman, J., Hövelmanns, K., Kiltz, E., Lyubashevsky, V., Seiler, G., Unruh, D.: A thorough treatment of highly-efficient ntru instantiations. In: IACR International Conference on Public-Key Cryptography. pp. 65-94. Springer (2023)
[18] Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) Advances in Cryptology - CRYPTO' 99. pp. 537-554. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
[19] Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: International algorithmic number theory symposium. pp. 267-288. Springer (1998)
[20] Howgrave-Graham, N.: A hybrid lattice-reduction and meet-in-the-middle attack against ntru. In: Menezes, A. (ed.) Advances in Cryptology - CRYPTO 2007. pp. 150-169. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
[21] Lee, J., Kim, D., Lee, H., Lee, Y., Cheon, J.H.: Rlizard: Post-quantum key encapsulation mechanism for iot devices. IEEE Access 7, 2080-2091 (2018)
[22] Lindner, R., Peikert, C.: Better key sizes (and attacks) for lwe-based encryption. Cryptology ePrint Archive, Paper 2010/613 (2010), https://eprint.iacr.org/ 2010/613, https://eprint.iacr.org/2010/613
[23] Lyubashevsky, V.: Fiat-shamir with aborts: Applications to lattice and factoringbased signatures. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 598-616. Springer (2009)
[24] Lyubashevsky, V.: Lattice signatures without trapdoors. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 738-755. Springer (2012)
[25] Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. Journal of the ACM (JACM) 60(6), 1-35 (2013)
[26] Lyubashevsky, V., Seiler, G.: Nttru: truly fast ntru using ntt. Cryptology ePrint Archive (2019)
[27] May, A.: How to meet ternary lwe keys. Cryptology ePrint Archive, Paper 2021/216 (2021), https://eprint.iacr.org/2021/216, https://eprint.iacr. org/2021/216
[28] NIST: Post-quantum cryptography, available from: https://csrc.nist.gov/ projects/post-quantum-cryptography [last accessed June 2023]
[29] NIST: Standardization of additional digital signature schemes, available from: https://csrc.nist.gov/projects/pqc-dig-sig/standardization [last accessed August 2023]
[30] Pointcheval, D., Johansson, T.: Advances in cryptology - eurocrypt 2012 : 31st annual international conference on the theory and applications of cryptographic techniques, cambridge, uk, april 15-19, 2012. proceedings. Lecture notes in computer science, 7237, EUROCRYPT (31st : 2012 : Cambridge, England), Springer, Berlin (2012)
[31] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM 56(6) (sep 2009). https://doi.org/10.1145/1568318.1568324, https://doi.org/10.1145/1568318.1568324
[32] Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing 26(5), 14841509 (oct 1997). https://doi.org/10.1137/s0097539795293172, https://doi.org/ 10.1137\%2Fs0097539795293172

# On the security of REDOG 

Tanja Lange, Alex Pellegrini, and Alberto Ravagnani<br>Eindhoven University of Technology, the Netherlands<br>tanja@hyperelliptic.org, alex.pellegrini@live.com, a.ravagnani@tue.nl


#### Abstract

We analyze REDOG, a public-key encryption system submitted to the Korean competition on post-quantum cryptography. REDOG is based on rank-metric codes. We prove its incorrectness and attack its implementation, providing an efficient message recovery attack. Furthermore, we show that the security of REDOG is much lower than claimed. We then proceed to mitigate these issues and provide two approaches to fix the decryption issue, one of which also leads to better security.


Keywords: post-quantum crypto, code-based- crypto, rank-metric codes

## 1 Introduction

This paper analyzes the security of the REinforced modified Dual-Ouroboros based on Gabidulin codes, REDOG [KHL ${ }^{+} 22 a$ ], a public-key encryption system submitted to KpqC, the Korean competition on post-quantum cryptography. REDOG is a code-based cryptosystem using rank-metric codes, aiming at providing a rank-metric alternative to Hamming-metric code-based cryptosystems.

Rank-metric codes were introduced by Delsarte [Del78] and independently rediscovered by Gabidulin [Gab85] in 1985, who focused on those that are linear over a field extension. Gabidulin, Paramonov, and Tretjakov [GPT91] proposed their use for cryptography in 1991. The GPT system was attacked by Overbeck [Ove05,Ove08] who showed structural attacks, permitting recovery of the private key from the public key.

During the mid 2010s new cryptosystems using rank-metric codes were developed such as Ouroboros [DGZ17] and the first round of the NIST competition on post-quantum cryptography saw 5 systems based on rank-metric codes: LAKE [ABD $\left.{ }^{+} 17 \mathrm{a}\right]$, LOCKER $\left[\mathrm{ABD}^{+} 17 \mathrm{~b}\right], \mathrm{McNie}\left[\mathrm{GKK}^{+} 17\right]$, OuroborosR $\left[\mathrm{AAB}^{+} 17 \mathrm{a}\right]$. RQC $\left[\mathrm{AAB}^{+} 17 \mathrm{~b}\right]$. For further information about all these systems

[^27]see NIST's Round-1 Submissions page. Gaborit announced an attack weakening McNie and the McNie authors adjusted their parameters. A further attack was published in [LT18] and NIST did not advance McNie into the second round of the competition.

ROLLO, a merger of LAKE, LOCKER and Ouroboros-R, and RQC made it into the the second round but got broken near the end of it by significant advances in the cryptanalysis of rank-metric codes and the MinRank problem in general, see $\left[\mathrm{BBB}^{+} 20\right]$ and $\left[\mathrm{BBC}^{+} 20\right]$. In their report at the end of round $2\left[\mathrm{AASA}^{+} 20\right]$, NIST wrote an encouraging note on rank-metric codes: "Despite the development of algebraic attacks, NIST believes rank-based cryptography should continue to be researched. The rank metric cryptosystems offer a nice alternative to traditional hamming metric codes with comparable bandwidth." (capitalization as in the original).

Kim, Kim, Galvez, and Kim [KKGK21] proposed a new rank-metric system in 2021 which was then analyzed by Lau, Tan, and Prabowo in [LTP21] who also proposed some modifications to the issues they found. REDOG closely resembles the system in [LTP21] and uses the same parameters.

Our contribution In this paper we expose weaknesses of REDOG and show that the system, as described in the documentation, is incorrect. To start with, we prove that REDOG does not decrypt correctly. The documentation and [LTP21] contain an incorrect estimate of the rank of an element which causes the input to the decoding step to have too large rank. The system uses Gabidulin codes [Gab85] which are MRD (Maximum Rank Distance) codes, meaning that vectors with errors of rank larger than half the minimum distance will decode to a different codeword, thus causing incorrect decryption in the REDOG system.

As a second contribution we attack ciphertexts produced by REDOG's reference implementation. We show that we can use techniques from the Hamming metric to obtain a message-recovery attack. This stems from a choice in the implementation which avoids the above-mentioned decryption problem. However, the errors introduced in the ciphertext have a specific shape which allows us to apply basic techniques of Information Set Decoding (ISD) over the Hamming metric to recover the message in seconds.

As a third contribution, we show that, independently of the special choice of error vectors in the implementation, the security of the cryptosystem is lower than the claimed security level. The main effect comes from a group of attacks published in $\left[\mathrm{BBC}^{+} 20\right]$ which the REDOG designers had not taken into account. An smaller effect comes from a systematic scan through all attack parameters.

Finally, we provide two ways to make REDOG's decryption correct. The first is a minimal change to fix the system by changing the space from which some matrix $P^{-1}$ is chosen in a way that differs from the choice in REDOG and avoids the issue mentioned above. However, this still requires choosing much larger parameters to deal with our third contribution. The second way makes a different change to REDOG which improves the resistance to attacks while also fixing the decryption issue. We show that, using this strategy, not only are

REDOG's parameters sufficient to reach any claimed security level, but they provide security abundantly beyond each level, allowing room for an eventual optimization. Note, however, that these estimates are obtained from big- $\mathcal{O}$ complexity estimates, putting all constants to 1 and lower-order terms to 0 , and thus underestimate the security.

## 2 Preliminaries and background notions

This section gives the necessary background on rank-metric codes for the rest of the paper. Let $\left\{\alpha_{1}, \ldots, \alpha_{m}\right\}$ be a basis of $\mathbb{F}_{q^{m}}$ over $\mathbb{F}_{q}$. Write $x \in \mathbb{F}_{q^{m}}$ uniquely as $x=\sum_{i=1}^{m} X_{i} \alpha_{i}, X_{i} \in \mathbb{F}_{q}$ for all $i$. So $x$ can be represented as $\left(X_{1}, \ldots, X_{m}\right) \in$ $\mathbb{F}_{q}^{m}$. We will call this the vector representation of $x$. Extend this process to $\mathbf{v}=\left(v_{1}, \ldots, v_{n}\right) \in \mathbb{F}_{q^{m}}^{n}$ defining a map Mat $: \mathbb{F}_{q^{m}}^{n} \rightarrow \mathbb{F}_{q}^{m \times n}$ by:

$$
\mathbf{v} \mapsto\left[\begin{array}{cccc}
V_{11} & V_{21} & \ldots & V_{n 1} \\
V_{12} & V_{22} & \ldots & V_{n 2} \\
\vdots & \vdots & \ddots & \vdots \\
V_{1 m} & V_{2 m} & \ldots & V_{n m}
\end{array}\right]
$$

Definition 2.1. The rank weight of $\mathbf{v} \in \mathbb{F}_{q^{m}}^{n}$ is defined as $\mathrm{wt}_{R}(\mathbf{v}):=\operatorname{rk}_{q}(\operatorname{Mat}(\mathbf{v}))$ and the rank distance between $\mathbf{v}, \mathbf{w} \in \mathbb{F}_{q^{m}}^{n}$ is $d_{R}(\mathbf{v}, \mathbf{w}):=\mathbf{w t}_{R}(\mathbf{v}-\mathbf{w})$.

Remark 2.2. It can be shown that the rank distance does not depend on the choice of the basis of $\mathbb{F}_{q^{m}}$ over $\mathbb{F}_{q}$. In particular, the choice of the basis is irrelevant for the results in this document.

When talking about the space spanned by $\mathbf{v} \in \mathbb{F}_{q^{m}}^{n}$, denoted as $\langle\mathbf{v}\rangle$, we mean the $\mathbb{F}_{q}$-subspace of $\mathbb{F}_{q}^{m}$ spanned by the columns of $\operatorname{Mat}(\mathbf{v})$.

For completeness, we introduce the Hamming weight and the Hamming distance. These notions will be used in our message recovery attack against REDOG's implementation.

The Hamming weight of a vector $\mathbf{v} \in \mathbb{F}_{q^{m}}^{n}$ is defined as $\mathrm{wt}_{H}(\mathbf{v}):=\#\{i \in$ $\left.\{1, \ldots, n\} \mid v_{i} \neq 0\right\}$ and the Hamming distance between vectors $\mathbf{v}, \mathbf{w} \in \mathbb{F}_{q^{m}}^{n}$ is defined as $d_{H}(\mathbf{v}, \mathbf{w}):=\mathbf{w t}_{H}(\mathbf{v}-\mathbf{w})$.

Let $D=d_{R}$ or $D=d_{H}$. Then an $[n, k, d]$-code $C$ with respect to $D$ over $\mathbb{F}_{q^{m}}$ is a $k$-dimensional $\mathbb{F}_{q^{m}}$-linear subspace of $\mathbb{F}_{q^{m}}^{n}$ with minimum distance

$$
d:=\min _{\mathbf{a}, \mathbf{b} \in C, \mathbf{a} \neq \mathbf{b}} D(\mathbf{a}, \mathbf{b})
$$

and correction capability $\lfloor(d-1) / 2\rfloor$. If $D=d_{R}$ (resp. $D=d_{H}$ ) then the code $C$ is also called a rank-metric (resp. Hamming-metric) code. All codes in this document are linear over the field extension $\mathbb{F}_{q^{m}}$.

We say that $G$ is a generator matrix of $C$ if its rows span $C$. We say that $H$ is a parity check matrix of $C$ if $C$ is the right-kernel of $H$.

A very well-known family of rank metric codes are Gabidulin codes [Gab85], which have $d=n-k+1$.

In this paper we can mostly use these codes as a black box, knowing that there is an efficient decoding algorithm using the parity-check matrix of the code and decoding vectors with errors of rank up to $\lfloor(d-1) / 2\rfloor$.

A final definition necessary to understand REDOG is that of isometries.
Definition 2.3. Consider vectors in $\mathbb{F}_{q^{m}}^{n}$. An isometry with respect to the rank metric is a matrix $P \in \mathrm{GL}_{n}\left(\mathbb{F}_{q^{m}}\right)$ satisfying that $\mathrm{wt}_{R}(\mathbf{v} P)=\mathrm{wt}_{R}(\mathbf{v})$ for any $\mathbf{v} \in \mathbb{F}_{q^{m}}^{n}$.

Obviously matrices $P \in \mathrm{GL}_{n}\left(\mathbb{F}_{q}\right)$ are isometries as $\mathbb{F}_{q}$-linear combinations of the coordinates of $\mathbf{v}$ do not increase the rank and the rank does not decrease as $P$ is invertible. The rank does also not change under scalar multiplication by some $\alpha \in \mathbb{F}_{q^{m}}^{*}: \mathrm{wt}_{R}(\alpha \mathbf{v})=\mathrm{wt}_{R}(\mathbf{v})$. Note that the latter corresponds to multiplication by $P=\alpha I_{n}$.

Berger [Ber03] showed that any isometry is obtained by composing these two options.
Theorem 2.4. [Ber03, Theorem 1] The isometry group of $\mathbb{F}_{q^{m}}^{n}$ for the rank metric is generated by scalar multiplications by elements in $\mathbb{F}_{q^{m}}^{*}$ and elements of $\mathrm{GL}_{n}\left(\mathbb{F}_{q}\right)$. This group is isomorphic to the product group $\left(\mathbb{F}_{q^{m}}^{*} / \mathbb{F}_{q}^{*}\right) \times \mathrm{GL}_{n}\left(\mathbb{F}_{q}\right)$.

## 3 System specification

This section introduces the specification of REDOG. We follow the notation of [LTP21], with minor changes.

The system parameters are positive integers $(n, k, \ell, q, m, r, \lambda, t)$, with $\ell<n$ and $\lambda t \leq r \leq\lfloor(n-k) / 2\rfloor$, as well as a hash function hash : $\mathbb{F}_{q^{m}}^{2 n-k} \rightarrow \mathbb{F}_{q^{m}}^{\ell}$.

KeyGen:

1. Select $H=\left(H_{1} \mid H_{2}\right), H_{2} \in \mathrm{GL}_{n-k}\left(\mathbb{F}_{q^{m}}\right)$, a parity check matrix of a [ $2 n-k, n$ ] Gabidulin code, with syndrome decoder $\Phi$ correcting $r$ errors.
2. Select a full rank matrix $M \in \mathbb{F}_{q^{m}}^{\ell \times n}$ and isometry $P \in \mathbb{F}_{q^{m}}^{n \times n}$ (w.r.t. the rank metric).
3. Select a $\lambda$-dimensional subspace $\Lambda \subset \mathbb{F}_{q^{m}}$, seen as $\mathbb{F}_{q}$-linear space, containing 1 and select $S^{-1} \in \mathrm{GL}_{n-k}(\Lambda)$; see Section 4 for the definition.
4. Compute $F=M P^{-1} H_{1}^{T}\left(H_{2}^{T}\right)^{-1} S$ and publish the public key $\mathrm{pk}=(M, F)$. Store the secret key sk $=(P, H, S, \Phi)$.
Encrypt $\left(\mathbf{m} \in \mathbb{F}_{q^{m}}^{\ell}, \mathrm{pk}\right)$
5. Generate uniformly random $\mathbf{e}=\left(\mathbf{e}_{1}, \mathbf{e}_{2}\right) \in \mathbb{F}_{q^{m}}^{2 n-k}$ with $\mathrm{wt}_{R}(\mathbf{e})=t$, $\mathbf{e}_{1} \in \mathbb{F}_{q^{m}}^{n}$ and $\mathbf{e}_{2} \in \mathbb{F}_{q^{m}}^{n-k}$.
6. Compute $\mathbf{m}^{\prime}=\mathbf{m}+$ hash(e).
7. Compute $\mathbf{c}_{1}=\mathbf{m}^{\prime} M+\mathbf{e}_{1}$ and $\mathbf{c}_{2}=\mathbf{m}^{\prime} F+\mathbf{e}_{2}$ and send $\left(\mathbf{c}_{1}, \mathbf{c}_{2}\right)$.

Decrypt (( $\left.\mathbf{c}_{1}, \mathbf{c}_{2}\right)$, sk)

1. Compute $\mathbf{c}^{\prime}=\mathbf{c}_{1} P^{-1} H_{1}^{T}-\mathbf{c}_{2} S^{-1} H_{2}^{T}=\mathbf{e}^{\prime} H^{T}$ where the vector $\mathbf{e}^{\prime}:=$ $\left(\mathbf{e}_{1} P^{-1},-\mathbf{e}_{2} S^{-1}\right)$.
2. Decode $\mathbf{c}^{\prime}$ using $\Phi$ to obtain $\mathbf{e}^{\prime}$, recover $\mathbf{e}=\left(\mathbf{e}_{1}, \mathbf{e}_{2}\right)$ using $P$ and $S$.
3. Solve $\mathbf{m}^{\prime} M=\mathbf{c}_{1}-\mathbf{e}_{1}$. Output $\mathbf{m}=\mathbf{m}^{\prime}-\operatorname{hash}(\mathbf{e})$.

Suggested parameters We list the suggested parameters of REDOG for 128,192 and 256 bits of security, following [KHL+22a] submitted to KpqC.

| Security parameter | $(n, k, \ell, q, m, r, \lambda, t)$ |
| :---: | :---: |
| 128 | $(44,8,37,2,83,18,3,6)$ |
| 192 | $(58,10,49,2,109,24,3,8)$ |
| 256 | $(72,12,61,2,135,30,3,10)$ |

Table 1. Suggested parameters; see [KHL $\left.{ }^{+} 22 \mathrm{a}\right]$.

## 4 Incorrectness of decryption

This section shows that decryption typically fails for the version of REDOG specified in $\left[\mathrm{KHL}^{+} 22 \mathrm{a}\right.$, LTP21]. The novelty of this specification, compared to that introduced in [KKGK21], lies in the selection of the invertible matrix $S^{-1}$ in Step 3, which is selected with the property that $S^{-1} \in \mathrm{GL}_{n-k}(\Lambda)$, where $\Lambda$ is a $\lambda$-dimensional $\mathbb{F}_{q}$-subspace of $\mathbb{F}_{q^{m}}$. This method has been first proposed by Loidreau in [Loi17], but it appears to be incorrectly applied in REDOG. Before providing more details about this claim and proving the incorrectness of REDOG's decryption process, we will shed some light on the object $\mathrm{GL}_{n-k}(\Lambda)$. Unlike the notation suggests, this is not a group, but a potentially unstructured subset of $\mathrm{GL}_{n-k}\left(\mathbb{F}_{q^{m}}\right)$ defined as follows:

Let $\left\{1, \alpha_{2}, \ldots, \alpha_{\lambda}\right\} \subset \mathbb{F}_{q^{m}}$ be a set of elements that are $\mathbb{F}_{q}$-linearly independent. Let $\Lambda \subset \mathbb{F}_{q^{m}}$ be the set of $\mathbb{F}_{q}$-linear combinations of these $\alpha_{i}$ 's. This set forms an $\mathbb{F}_{q}$-linear vectorspace. Now, $S^{-1} \in \mathrm{GL}_{n-k}(\Lambda)$ is defined to mean that $S$ is an invertible $(n-k) \times(n-k)$ matrix with the property that the entries of $S^{-1}$ are elements of $\Lambda$. Note that such an $S$ exists because $\lambda \geq 1$ by assumption. The REDOG documentation $\left[\mathrm{KHL}^{+} 22 \mathrm{a}\right]$ points out that this does not imply that $S \in \mathrm{GL}_{n-k}(\Lambda)$, hence, despite what the notation may suggest, $\mathrm{GL}_{n-k}(\Lambda)$ is not a group in general.

We continue by giving a proof, and an easy generalization for any $q$, of [Loi17, Proposition 1].

Proposition 4.1. Let $\lambda, t, n$ be positive integers such that $\lambda t \leq n, A \in \mathrm{GL}_{n}(\Lambda)$ where $\Lambda \subset \mathbb{F}_{q^{m}}$ is a $\lambda$-dimensional subspace of $\mathbb{F}_{q^{m}}$, and $\mathbf{x} \in \mathbb{F}_{q^{m}}^{n}$ with $\mathrm{wt}_{R}(\mathbf{x})=$ $t$. Then

$$
\mathrm{wt}_{R}(\mathrm{x} A) \leq \lambda t .
$$

Proof. Let $\Gamma$ be the subspace of $\mathbb{F}_{q^{m}}$ generated by the entries of $\mathbf{x}=\left(x_{1}, \ldots, x_{n}\right)$. Since $\Gamma$ has dimension $t$, we can write $\Gamma=\left\langle y_{1}, \ldots, y_{t}\right\rangle$ with $y_{i} \in \mathbb{F}_{q^{m}}$. Similarly for $\Lambda$, we can write $\Lambda=\left\langle\alpha_{1}, \ldots, \alpha_{\lambda}\right\rangle$ with $\alpha_{i} \in \mathbb{F}_{q^{m}}$. Express $\mathbf{x} A$ as

$$
\mathbf{x} A=\left(\sum_{i=1}^{n} x_{i} A_{i, 1}, \ldots, \sum_{i=1}^{n} x_{i} A_{i, n}\right) .
$$

Fix $j \in\{1, \ldots, n\}$. Then

$$
(\mathrm{x} A)_{j}=\sum_{i=1}^{n} x_{i} A_{i, j}=\sum_{i=1}^{n}\left(\left(\sum_{h=1}^{t} x_{i, h} y_{h}\right)\left(\sum_{k=1}^{\lambda} A_{i, j, k} \alpha_{k}\right)\right),
$$

with $x_{i, h}, A_{i, j, k} \in \mathbb{F}_{q}$. By rearranging the terms we obtain

$$
\begin{equation*}
(\mathbf{x} A)_{j}=\sum_{h=1}^{t} \sum_{k=1}^{\lambda}\left(\sum_{i=1}^{n} x_{i, h} A_{i, j, k}\right) y_{h} \alpha_{k} . \tag{1}
\end{equation*}
$$

Therefore each entry of $\mathbf{x} A$ can be expressed as an $\mathbb{F}_{q^{-}}$-linear combination of the $\lambda t$ elements of the form $y_{h} \alpha_{k}$.

We will now show that REDOG typically does not decrypt correctly. In order to do so, we need some preliminary results and tools. The proof of the next lemma uses some tools from combinatorics. It computes the probability that a randomly selected $t$-tuple of elements of a $t$-dimensional vector space spans the entire space.

Lemma 4.2. Let $V$ be a $t$-dimensional subspace $V \subseteq \mathbb{F}_{q}^{m}$ and let $S \in V^{s}$ be a uniformly random s-tuple of elements of $V$. The probability $p(q, s, t)$ that $\left\langle S_{i}\right|$ $i \in\{1, \ldots, s\}\rangle=V$ is 0 if $0 \leq s<t$ and

$$
p(q, s, t)=\sum_{i=0}^{t}\left[\begin{array}{l}
t  \tag{2}\\
i
\end{array}\right]_{q}(-1)^{t-i} q^{s(i-t)+\binom{t-i}{2}}
$$

otherwise, where $\left[\begin{array}{l}t \\ i\end{array}\right]_{q}$ is the $q$-binomial coefficient, counting the number of subspaces of dimension i of $\mathbb{F}_{q}^{t}$, and $\binom{a}{b}=0$ for $a<b$. In particular, this probability does not depend on $m$ or on the choice of $V$, but only on its dimension.

Proof. Let $(\mathcal{P}, \subseteq)$ be the poset (partially ordered set) of subspaces of $\mathbb{F}_{q}^{m}$ ordered by inclusion. Recall that the Möbius function of $\mathcal{P}$, and of any finite poset, is defined, for $A, B \in \mathcal{P}$, as

$$
\mu(B, A)= \begin{cases}1 & \text { if } B=A \\ -\sum_{C \mid B \subseteq C \subset A} \mu(B, C) & \text { if } B \subset A \\ 0 & \text { otherwise }\end{cases}
$$

For the poset of subspaces, the Möbius function is computed e.g. in [Sta11, Example 3.10.2] as

$$
\mu(B, A)= \begin{cases}(-1)^{k} q^{\binom{k}{2}} & \text { if } B \subseteq A \text { and } \operatorname{dim}(A)-\operatorname{dim}(B)=k  \tag{3}\\ 0 & \text { otherwise } .\end{cases}
$$

We want to compute the function $f: \mathcal{P} \rightarrow \mathbb{N}$ defined as

$$
f(A)=\#\left\{S \in\left(\mathbb{F}_{q}^{m}\right)^{s} \mid\langle S\rangle=A\right\}
$$

Clearly, if $s<\operatorname{dim} A$, there does not exist any $s$-tuple $S$ spanning $A$, hence $f(A)=0$, which gives the first case of (2). We can therefore restrict ourselves to the case $s \geq \operatorname{dim} A$. Define the auxiliary function $g: \mathcal{P} \rightarrow \mathbb{N}$ as

$$
\begin{aligned}
g(A) & =\sum_{B \subseteq A} f(B) \\
& =\#\left\{S \in\left(\mathbb{F}_{q}^{m}\right)^{s} \mid\langle S\rangle \subseteq A\right\} \\
& =|A|^{s}=q^{s \operatorname{dim} A}
\end{aligned}
$$

Then by Möbius inversion we can compute:

$$
\begin{equation*}
f(A)=\sum_{B \subseteq A} g(B) \mu(B, A) \tag{4}
\end{equation*}
$$

Splitting the sum over the dimensions, and substituting the values in Equation 3, we can obtain

$$
\begin{aligned}
f(V) & =\sum_{i=0}^{t} \sum_{U \subseteq V, \operatorname{dim} U=i} g(U) \mu(U, V) \\
& =\sum_{i=0}^{t} q^{s i}(-1)^{t-i} q^{\binom{t-i}{2}} \sum_{U \subseteq V, \operatorname{dim} U=i} 1 \\
& =\sum_{i=0}^{t}\left[\begin{array}{l}
t \\
i
\end{array}\right]_{q}(-1)^{t-i} q^{s i+\binom{t-i}{2}}
\end{aligned}
$$

The probability can be computed by dividing $f(V)$ by the number of $s$-tuples of elements of $V$, that is, $q^{s t}$.

Remark 4.3. The probability given in Lemma 4.2 can be interpreted as the ratio of the number of surjective linear maps from $\mathbb{F}_{q}^{s}$ onto $\mathbb{F}_{q}^{t}$ over the total number of linear maps.

We next compute the probability that by truncating a rank $t$ vector, the rank stays the same.

Theorem 4.4. Let $\mathbf{e}=\left(\mathbf{e}_{1}, \mathbf{e}_{2}\right) \in \mathbb{F}_{q^{m}}^{2 n-k}$, with $\mathbf{e}_{1} \in \mathbb{F}_{q^{m}}^{n}$ and $\mathbf{e}_{2} \in \mathbb{F}_{q^{m}}^{n-k}$, be a uniformly random error with $\mathrm{wt}_{R}(\mathbf{e})=t$. Then $\mathrm{wt}_{R}\left(\mathbf{e}_{1}\right)=t$ and $\mathrm{wt}_{R}\left(\mathbf{e}_{2}\right)=t$ with probability $p(q, n, t) / p(q, 2 n-k, t)$ and $p(q, n-k, t) / p(q, 2 n-k, t)$ respectively.

Proof. By definition, the probability that $\mathrm{wt}_{R}\left(\mathbf{e}_{1}\right)=t$ is the ratio

$$
\begin{equation*}
\pi=\frac{\#\left\{\mathbf{e} \in \mathbb{F}_{q^{m}}^{2 n-k} \mid \mathrm{wt}_{R}(\mathbf{e})=t \text { and } \mathrm{wt}_{R}\left(\mathbf{e}_{1}\right)=t\right\}}{\#\left\{\mathbf{e} \in \mathbb{F}_{q^{m}}^{2 n-k} \mid \mathrm{wt}_{R}(\mathbf{e})=t\right\}} \tag{5}
\end{equation*}
$$

We can split the cardinalities above over all the subspaces of $\mathbb{F}_{q}^{m}$ of dimension $t$ as follows:

$$
\begin{equation*}
\pi=\frac{\sum_{V \subset \mathbb{F}_{q}^{m}, \operatorname{dim} V=t} \#\left\{\mathbf{e} \in \mathbb{F}_{q^{m}}^{2 n-k} \mid\langle\mathbf{e}\rangle=\left\langle\mathbf{e}_{1}\right\rangle=V\right\}}{\sum_{V \subset \mathbb{F}_{q}^{m}, \operatorname{dim} V=t} \#\left\{\mathbf{e} \in \mathbb{F}_{q^{m}}^{2 n-k} \mid\langle\mathbf{e}\rangle=V\right\}} \tag{6}
\end{equation*}
$$

It is not hard to prove that the summands in (4) are independent of the space $V$. Therefore

$$
\pi=\frac{\#\left\{\mathbf{e} \in \mathbb{F}_{q^{m}}^{2 n-k} \mid\langle\mathbf{e}\rangle=\left\langle\mathbf{e}_{1}\right\rangle=V\right\}}{\#\left\{\mathbf{e} \in \mathbb{F}_{q^{m}}^{2 n-k} \mid\langle\mathbf{e}\rangle=V\right\}}=\frac{\#\left\{\mathbf{e}_{1} \in \mathbb{F}_{q^{m}}^{n} \mid\left\langle\mathbf{e}_{1}\right\rangle=V\right\} q^{t(n-k)}}{\#\left\{\mathbf{e} \in \mathbb{F}_{q^{m}}^{2 n-k} \mid\langle\mathbf{e}\rangle=V\right\}}
$$

where $V$ is any subspace of $\mathbb{F}_{q}^{m}$ of dimension $t$. By applying Lemma 4.2 we then get

$$
\pi=\frac{p(q, n, t) q^{n t} q^{t(n-k)}}{p(q, 2 n-k, t) q^{(2 n-k) t}}=\frac{p(q, n, t)}{p(q, 2 n-k, t)}
$$

as claimed. The probability for $\mathbf{e}_{2}$ can be computed with the same arguments as for $\mathbf{e}_{1}$.

Remark 4.5. In the context of a REDOG instance, the data $q, n$ and $t$ is fixed, hence, for the sake of reading simplicity, we denote the probability given in Theorem 4.4 by

$$
\bar{p}(r, t)=\frac{p(q, r, t)}{p(q, 2 n-k, t)} .
$$

Example 4.6. Consider the suggested parameters of REDOG for 128 bits of security from Table 1. Using SageMath $\left[\mathrm{S}^{+} 21\right]$ we computed the probability that $\mathrm{wt}_{R}\left(\mathbf{e}_{1}\right)=t$, that is $\bar{p}(44,6)=0.999999999996419$, and the probability that $\mathrm{wt}_{R}\left(\mathbf{e}_{2}\right)=t$, that is $\bar{p}(36,6)=0.999999999083229$.

We are ready to state the following theorem, which directly implies that REDOG's decryption process fails with extremely high probability.
Theorem 4.7. Let ( $n, k, q, m, \lambda, t$ ) be integers with $k<n<m$ and $\lambda t \leq m$. Let $\Lambda \subset \mathbb{F}_{q^{m}}$ be a $\lambda$-dimensional subspace of $\mathbb{F}_{q^{m}}$ and $\mathbf{e}=\left(\mathbf{e}_{1}, \mathbf{e}_{2}\right)$ as in Theorem 4.4. Let $P \in \mathbb{F}_{q^{m}}^{n \times n}$ be a random isometry matrix (w.r.t. the rank metric) and $S^{-1} \in$ $\mathrm{GL}_{n-k}(\Lambda)$. Then $\mathbf{e}^{\prime}:=\left(\mathbf{e}_{1} P^{-1},-\mathbf{e}_{2} S^{-1}\right)$ has rank weight $\mathrm{wt}_{R}\left(\mathbf{e}^{\prime}\right) \geq \lambda t+1$ with probability bounded from below by

$$
p_{\text {fail }}(n, k, q, m, \lambda, t):=\bar{p}(n, t) \bar{p}(n-k, \lambda t) \bar{p}(n-k, t)\left(1-\left[\begin{array}{c}
\lambda t \\
t
\end{array}\right]_{q} /\left[\begin{array}{c}
m \\
t
\end{array}\right]_{q}\right) .
$$

Proof. By Theorem 2.4, the isometry $P$ is of the form $\alpha \bar{P}$ for $\alpha \in \mathbb{F}_{q^{m}}^{*}$ and $\bar{P} \in$ $\mathrm{GL}_{n}\left(\mathbb{F}_{q}\right)$, where $q^{m} \gg q$ and thus typically $\alpha \notin \mathbb{F}_{q}$. Because of the multiplication by $\alpha^{-1}$, we can assume that the linear transformation induced by $P^{-1}$ takes a $t$ dimensional subvectorspace of $\mathbb{F}_{q}^{m}$ to a random $t$-dimensional subspace. Similarly we assume that $S^{-1}$ sends a $t$-dimensional subspace of $\mathbb{F}_{q}^{m}$ to a random subspace of dimension at most $\lambda t$, by Proposition 4.1. We get the lower bound on the failure probability by showing the following:

1. $\operatorname{wt}_{R}\left(\mathbf{e}_{1} P^{-1}\right)=t$ with probability $\bar{p}(n, t)$;
2. $\operatorname{wt}_{R}\left(-\mathbf{e}_{2} S^{-1}\right)=\lambda t$ with probability $\bar{p}(n-k, t) \bar{p}(n-k, \lambda t)$;
3. under the conditions in (1) and (2), $\left\langle\mathbf{e}_{1} P^{-1}\right\rangle \not \subset\left\langle-\mathbf{e}_{2} S^{-1}\right\rangle$ with probability $1-\left[\begin{array}{c}\lambda t \\ t\end{array}\right]_{q} /\left[\begin{array}{c}m \\ t\end{array}\right]_{q}$.
Note that (1) follows directly from Theorem 4.4 and the fact that $P$ is an isometry of the space w.r.t the rank metric.

Likewise, $\mathrm{wt}_{R}\left(-\mathbf{e}_{2}\right)=t$ with probability $\bar{p}(n-k, t)$. The proof of Proposition 4.1 shows that for $\mathbf{e}_{2}$ with $\mathrm{wt}_{R}\left(-\mathbf{e}_{2}\right)=t$ we have that $-\mathbf{e}_{2} S^{-1}$ is contained in a $\lambda t$-dimensional subspace of $\mathbb{F}_{q}^{m}$. Again by Theorem 4.4 we obtain that $\left\langle-\mathbf{e}_{2} S^{-1}\right\rangle$ spans the entire space with probability $\bar{p}(n-k, \lambda t)$, proving (2).

To prove (3) we will compute the opposite, i.e. the probability that $\left\langle\mathbf{e}_{1} P^{-1}\right\rangle$ is a subspace of $\left\langle-\mathbf{e}_{2} S^{-1}\right\rangle$. As mentioned at the beginning of the proof, we treat $\left\langle\mathbf{e}_{1} P^{-1}\right\rangle$ as a random $t$-dimensional subspace of $\mathbb{F}_{q^{m}}$. Thus we can compute this probability as the ratio between the number of $t$-dimensional subspaces of $\left\langle-\mathbf{e}_{2} S^{-1}\right\rangle$ and of $\mathbb{F}_{q}^{m}$, that is, $\left[\begin{array}{c}\lambda t \\ t\end{array}\right]_{q} /\left[\begin{array}{c}m \\ t\end{array}\right]_{q}$.

Combining the probabilities and observing that $(1-3)$ imply $\mathrm{wt}_{R}\left(\mathbf{e}^{\prime}\right) \geq \lambda t+1$ gives the result.

Remark 4.8. There are more ways to $\operatorname{get}^{\mathrm{wt}_{R}\left(\mathbf{e}^{\prime}\right) \geq \lambda t+1 \text { by relaxing the first }}$ two requirements in the proof of Theorem 4.7 and studying the dimension of the union in the third, but $p_{\text {fail }}$ is large enough for the parameters in REDOG to prove the point.
Remark 4.9. The proof of property (3) relies on $\mathbf{e}_{1} P^{-1}$ being a random subspace of dimension $t$. We note that for $\alpha \in \mathbb{F}_{q}$ we have $\left\langle\mathbf{e}_{1}\right\rangle=\left\langle\mathbf{e}_{1} P^{-1}\right\rangle \subset\left\langle\mathbf{e}_{2} S^{-1}\right\rangle$ for $S^{-1} \in \mathrm{GL}_{n-k}(\Lambda)$ and $1 \in \Lambda$. The latter constraint is stated in [ $\left.\mathrm{KHL}^{+} 22 \mathrm{a}\right]$ and [LTP21] and it is possible that the authors were not aware of the full generality of isometries. See also the full version [LPR23] for further observations on [LTP21] which are consistent with this misconception.

Recall that the decoder $\Phi$ can only correct errors up to rank weight $r=\lambda t$. By Theorem 4.7 we have that $\mathbf{e}^{\prime}$ has rank weight $\geq \lambda t+1$, hence the following corollary.

Corollary 4.10. Let $(n, k, \ell, q, m, r, \lambda, t)$ be the parameters of a instance of $R E$ $D O G$ with $r=\lambda t$. Then REDOG will produce decryption failures with probability at least $p_{\text {fail }}(n, k, q, m, \lambda, t)$.

Note that a $[2 n-k, n]$ Gabidulin code has minimum distance $d_{R}=2 n-k-$ $n+1=n-k+1$ and can thus correct at most $\lfloor(n-k) / 2\rfloor$ errors and that all instances of REDOG in Table 1 satisfy $\lfloor(n-k) / 2\rfloor=r=\lambda t$.

Example 4.11. As in Example 4.6, consider the suggested parameters for 128 bits of security. Then Theorem 4.7 states that $\mathrm{wt}_{R}\left(\mathbf{e}^{\prime}\right) \geq 19$ with probability at least $p_{\text {fail }}(44,8,2,83,3,6)=\bar{p}(44,8) \bar{p}(36,6) \bar{p}(36,18)\left(1-\left[\begin{array}{c}18 \\ 6\end{array}\right]_{2} /\left[\begin{array}{c}83 \\ 6\end{array}\right]_{2}\right)=$ 0.999996184401789 .

Table 2 reports the value of $p_{\text {fail }}$ for each set of security parameters given in Table 1. This shows that REDOG's decryption process fails almost always.

| Security parameter | $p_{\text {fail }}$ |
| :---: | :---: |
| 128 | 0.999996184401789 |
| 192 | 0.999999940394453 |
| 256 | 0.999999999068677 |

Table 2. Value of decryption failure probability $p_{\text {fail }}$ per suggested parameters.

## 5 Message recovery attack on REDOG's implementation

Theorem 4.7 and the numerical examples show that, with probability almost 1 , REDOG will fail decrypting. However, the probability is not exactly 1 and there exist some choices of $\mathbf{e}$ for which decryption still succeeds. One extreme way to avoid decryption failures, chosen in the refenrence implementation of REDOG, is to build errors as follows:

Algorithm 5.1 (REDOG's error generator)

1. Pick $\beta_{1}, \ldots, \beta_{t} \in \mathbb{F}_{q^{m}}$ being $\mathbb{F}_{q}$-linearly independent.
2. Pick random permutation $\pi$ on $2 n-k$ symbols.
3. Set $\mathbf{e}_{\text {init }}=\left(\beta_{1}, \ldots, \beta_{t}, 0, \ldots, 0\right) \in \mathbb{F}_{q^{m}}^{2 n-k}$. Output $\mathbf{e}=\pi\left(\mathbf{e}_{\text {init }}\right)$.

Error vectors in REDOG's reference implementation ${ }^{1}$, whose performance is analyzed in $\left[\mathrm{KHL}^{+} 22 \mathrm{~b}\right]$, are generated in an equivalent way to Algorithm 5.1. Indeed, $\mathbf{e}^{\prime}$ has rank weight $\mathrm{wt}_{R}\left(\mathbf{e}^{\prime}\right)=\left(\mathbf{e}_{1} P^{-1},-\mathbf{e}_{2} S^{-1}\right) \leq \lambda t$ and can therefore be decoded using $\Phi$.

Remark 5.2. Algorithm 5.1 produces an error vector $\mathbf{e} \operatorname{such}$ that $\mathrm{wt}_{H}(\mathbf{e})=$ $\mathrm{wt}_{R}(\mathbf{e})=t$ as only $t$ coordinates of $\mathbf{e}$ are nonzero.

We are ready to give the description of an efficient message recovery algorithm.

Algorithm 5.3 (Message recovery attack)
Input: REDOG's public key pk and a REDOG's ciphertext $\mathbf{c}=\left(\mathbf{c}_{1}, \mathbf{c}_{2}\right)=$ Encrypt( $\mathbf{m}, \mathrm{pk})$ generated by the reference implementation.
Output: m

1. Let $C^{\prime}$ be the linear $[2 n-k, \ell]$-code in the Hamming metric generated by $G=\left(\mathrm{pk}_{1} \mid \mathrm{pk}_{2}\right)$. Put $f=0$.
2. While $f=0$ :
(a) Randomly select $\ell$ columns of $G$ to form the matrix $A$. Let $\mathbf{c}_{A}$ be the matching positions in $\mathbf{c}$.

[^28](b) If $A$ is invertible
i. Compute $B=A^{-1}$ and $\overline{\mathbf{m}}=\mathbf{c}_{A} B$.
ii. Compute $\overline{\mathbf{c}}_{1}=\overline{\mathbf{m}} \mathrm{pk}_{1}$.
iii. If $\mathrm{wt}_{H}\left(\mathbf{c}_{1}-\overline{\mathbf{c}}_{1}\right)=t_{1} \leq t$
A. Compute $\overline{\mathbf{c}}_{2}=\overline{\mathbf{m}} \mathrm{pk}_{2}$.
B. If $\mathrm{wt}_{H}\left(\mathbf{c}_{2}-\overline{\mathbf{c}}_{2}\right)=t-t_{1}$

Put $\mathbf{m}^{\prime}=\overline{\mathbf{m}}, \mathbf{e}=\left(\mathbf{c}_{1}, \mathbf{c}_{2}\right)-\left(\overline{\mathbf{c}}_{1}, \overline{\mathbf{c}}_{2}\right)$ and $f=1$.
3. Compute $\mathbf{m}=\mathbf{m}^{\prime}-\operatorname{hash}(\mathbf{e})$.

The inner loop is Prange's information-set decoding algorithm [Pra62] in the generator-matrix form with early aborts. If the chosen $\ell$ positions are not all error free then $\overline{\mathbf{m}}$ equals $\mathbf{m}$ with one or more rows of $B$ added to it. Then $\overline{\mathbf{m}} \mathrm{pk}_{1}$ will be random vector and thus differ from $\mathbf{c}_{1}$ in more than $t$ positions. If the initial check succeeds there is a high chance of the second condition succceeding as well leading to $\mathbf{e}$ with $\mathrm{wt}_{H}(\mathbf{e})=t$.

We now analyze the success probability of each iteration of the inner loop of Algorithm 5.3. The field $\mathbb{F}_{q^{m}}$ is large, hence $A$ very likely to be invertible. The algorithm succeeds if the $\ell$ positions forming $A$ are chosen outside the positions where $\mathbf{e}$ has non-zero entries. This happens with probability $\binom{2 n-k-t}{\ell}\binom{2 n-k}{\ell}$.

Each trial costs the inversion of an $\ell \times \ell$ matrix and up to three matrix-vector products, where the vector has length $\ell$ and the matrices have $\ell, n$, and $n-k$ columns respectively, in addition to minor costs of two vector differences and two weight computations.

We implemented the attack in Algorithm 5.3 in Sagemath 9.5; see online for the code. We perform faster early aborts, testing $\overline{\mathbf{m}}$ on only $t+3$ columns of $\mathrm{pk}_{1}$. The probability that a coordinate matches between $\mathbf{c}_{1}$ and $\overline{\mathbf{c}}_{1}$ for $\overline{\mathbf{m}} \neq \mathbf{m}$ is $q^{-m}$ and thus negligible for large $m$. Hence, most candidate vectors $\overline{\mathbf{m}}$ are discared after $(t+3) \ell^{2}$ multiplications in $\mathbb{F}_{q^{m}}$. Running the attack on a Linux Mint virtual machine we broke the KAT ciphertexts included in the submssion package for all the proposed parameters. We also generated a bunch of ciphertexts corresponding to randomly chosen public keys and messages and measured the average running time of our algorithm.

As can be seen from Table 3, the attack on the reference implementation succeeds in few steps and is very fast to execute for all parameter sets.

| Security parameter | $\log _{2}($ Prob $)$ | Time $_{K A T}($ sec. $)$ | Time $_{100}($ sec. $)$ |
| :--- | :---: | ---: | ---: |
| 128 | -5.62325179726894 | $\sim 8.01$ | $\sim 9.17$ |
| 192 | -7.51182199577027 | $\sim 108.13$ | $\sim 112$ |
| 256 | -9.40052710879827 | $\sim 167.91$ | $\sim 133.43$ |

Table 3. Prob is the probability of success of one iteration of the inner loop of Algorithm 5.3. Time ${ }_{K A T}$ is the average timing of message recovery attack over entries in the KAT file ( 30 for 128 bits, 15 for 192 bits, 13 for 256 bits). Time ${ }_{100}$ is the average timing of message recovery attack over 100 ciphertext generated by REDOG's encryption.

## 6 Recomputing attacks costs

In this section we deal with the computation of complexities of general attacks against cryptosystems relying on the rank decoding problem. We noticed that the official REDOG submission [ $\mathrm{KHL}^{+} 22 \mathrm{a}$ ], as well as [LTP21] do not consider attack algorithms proposed in $\left[\mathrm{BBC}^{+} 20\right]$ and $\left[\mathrm{BBB}^{+} 23\right]$

Our computations are reported in Table 4 which shows that parameters suggested for REDOG provide significantly less security than expected. The tables also confirm that the parameters do provide the claimed security under attacks prior to $\left[\mathrm{BBC}^{+} 20\right]$ when using a realistic exponent for matrix multiplication. Note that the computations in these tables ignore all constants and lower-order terms in the big- $\mathcal{O}$ complexities. This is in line with how the authors of the attack algorithms use their results to determine the security of other systems, but typically constants are positive and large. We apply the same to $\left[\mathrm{BBB}^{+} 23\right]$ although their magma code makes different choices.

Overview of rank decoding attacks Recall that the public code is generated by the $\ell \times 2 n-k$ matrix $(M \mid F)$ over $\mathbb{F}_{q^{m}}$. The error vector added to the ciphertext is chosen to have rank $t$. In the description of the attacks we will give formulas for the costs using the notation of this paper, i.e., the dimension is $\ell$ and the error has rank $t$; we denote the length by $N$ for reasons that will become clear later. The complexity of algorithms also depends on the matrix multiplication exponent $\omega$.

The GRS [GRS16] algorithm is a combinatorial attack on the rank decoding problem. The idea behind this algorithm is to guess a vectorspace containing the space spanned by the error vector. In this way the received vector can be expressed in terms of the basis of the guessed space. The last step is to solve the linear system associated to the syndrome equations. This has complexity

$$
\begin{equation*}
\mathcal{O}\left((N-\ell)^{\omega} m^{\omega} q^{\min \{t\lfloor\ell m / N\rfloor,(t-1)\lfloor(\ell+1) m / N\rfloor\}}\right) . \tag{7}
\end{equation*}
$$

Note that we use $\omega$ here while the result originally was stated with exponent 3 . These matrices are not expected to be particularly sparse but should be large enough for fast matrix multiplication algorithms to apply. The same applies to the next formulas.

The second attack, introduced in [GRS16], which we denote GRS-alg, is an algebraic attack. Under the condition that $\ell>\lceil((t+1)(\ell+1)-N-1) / t\rceil$ the decoding problem can be solved in

$$
\begin{equation*}
\mathcal{O}\left(t^{\omega} \ell^{\omega} q^{t(\Gamma((t+1)(\ell+1)-N-1) / t\rceil)}\right) . \tag{8}
\end{equation*}
$$

The attack AGHT [AGHT18] is an improvement over the GRS combinatorial attack. The underlying idea is to guess the space containing the error in a specific way that provides higher chance of guessing a suitable space. It has complexity

$$
\begin{equation*}
\mathcal{O}\left((N-\ell)^{\omega} m^{\omega} q^{t(\ell+1) m / N-m}\right) . \tag{9}
\end{equation*}
$$

The BBB+ attack $\left[\mathrm{BBB}^{+} 20\right]$ translates the rank metric decoding problem into a system of multivariate equations and then uses Gröbner-basis methods to find solutions. Much of the analysis is spent on determining the degree of regularity, depending on the length, dimension, and rank of the code and error. If $m\binom{N-\ell-1}{t}+1 \geq\binom{ N}{t}$ then the problem can be solved in

$$
\begin{equation*}
\mathcal{O}\left(\left(\frac{((m+N) t)^{t}}{t!}\right)^{\omega}\right) \tag{10}
\end{equation*}
$$

If the condition is not satisfied then the complexity of solving the decoding problem becomes

$$
\begin{equation*}
\mathcal{O}\left(\left(\frac{((m+N) t)^{t+1}}{(t+1)!}\right)^{\omega}\right) \tag{11}
\end{equation*}
$$

or the same for $t+2$ in place of $t+1$. The authors of $\left[\mathrm{BBB}^{+} 20\right]$ use (11) in their calculations and thus we include that as well.

The BBC+-Overdetermined, BBC+-Hybrid and BBC+-SupportMinors improvements that will follow are all introduced in $\left[\mathrm{BBC}^{+} 20\right]$. They make explicit the use of extended linearization as a technique to compute Gröbner bases. For solving the rank-decoding problem it is not necessary to determine the full Gröbner basis but to find a solution to this system of equations. Extended linearization introduces new variables to turn a multivariate quadratic system into a linear system. The algorithms and complexity estimates differ in how large the resulting systems are and whether they are overdetermined or not, dependent on the system parameters.

BBC+-Overdetermined applies to the overdetermined case, which matches $m\binom{N-\ell-1}{t}+1 \geq\binom{ N}{t}$, and permits to solve the system in

$$
\begin{equation*}
\mathcal{O}\left(m\binom{N-\ell-1}{t}\binom{N}{t}^{\omega-1}\right) \tag{12}
\end{equation*}
$$

In case of an undetermined system, BBC+-Hybrid fixes some of the unknowns in a brute-force manner to produce to an overdetermined system in the remaining variables. The costs are testing all possible values for $j$ positions, where $j$ is the smallest non-negative integer such that $m\binom{N-\ell-1}{t}+1 \geq\binom{ N-j}{t}$, and for each performing the same matrix computations as in BBC on $j$ columns less. This leads to a total complexity of

$$
\begin{equation*}
\mathcal{O}\left(q^{j t} m\binom{N-\ell-1}{t}\binom{N-j}{t}^{\omega-1}\right) \tag{13}
\end{equation*}
$$

The brute-force part in BBC+-Hybrid quickly becomes the dominating factor. The BBC+-SupportMinors algorithm introduces terms of larger degrees first and then linearizes the system. This consists in multiplying the equations by some homogeneous monomials of degree $b$ so as to obtain a system of homogeneous equations. However, for the special case of $q=2$ the equations in the
system might not be homogeneous. In this case, homogeneous equations coming from smaller values of $b$ are considered. Let $A_{b}=\sum_{j=1}^{b}\binom{N}{t}\binom{m \ell+1}{j}$. The degree of the equations formed in BBC+-SupportMinors depends on $b$, where $0<b<2+t$ is minimal such that $a_{b}-1 \leq \sum_{j=1}^{b} \sum_{s=1}^{j}\left((-1)^{s+1}\binom{N}{t+s}\binom{m+s-1}{s}\binom{m \ell+1}{j-s}\right)$ if such a $b$ exists. In this case the problem can be solved with complexity

$$
\begin{equation*}
\mathcal{O}\left((m \ell+1)(t+1) A_{b}^{2}\right) . \tag{14}
\end{equation*}
$$

We do not report the last two attacks presented in $\left[\mathrm{BBC}^{+} 20\right]$ as the underlying approach has been pointed out to be incorrect in $\left[\mathrm{BBB}^{+} 23\right]$. More precisely, $\left[\mathrm{BBB}^{+} 23\right]$ show that the independence assumptions made in $\left[\mathrm{BBC}^{+} 20\right]$ are incorrect. The SupportMinors and MaxMinors modelings in $\left[\mathrm{BBC}^{+} 20\right]$ are not as independent as claimed, and $\left[\mathrm{BBB}^{+} 23\right]$ introduces a new approach that combines them while keeping independence, at least conjecturally and matched by experiments. They again multiply by monomials of degree up to $b-1$ but a relevant difference is that the equations from the SupportMinors system are kept over $\mathbb{F}_{q^{m}}$. They introduce the following notation:

$$
\begin{aligned}
\mathcal{N}_{b}^{\mathbb{F}_{q} m} & =\sum_{s=1}^{\ell}\binom{N-s}{t}\binom{\ell+b-1-s}{b-1}-\binom{N-\ell-1}{t}\binom{\ell-b-1}{b}, \\
\mathcal{N}_{b, s y z}^{\mathbb{F}_{q}} & =(m-1) \sum_{s=1}^{b}(-1)^{(s+1)}\binom{\ell+b-s-1}{b-s}\binom{N-\ell-1}{t+s}, \text { and } \\
\mathcal{M}_{b}^{\mathbb{F}_{q}} & =\binom{\ell+b-1}{b}\left(\binom{N}{t}-m\binom{N-\ell-1}{t}\right)
\end{aligned}
$$

and put $\mathcal{N}_{b}^{\mathbb{F}_{q}}=\mathcal{N}_{b}^{\mathbb{F}_{q} q^{m}}-\mathcal{N}_{b, s y z}^{\mathbb{F}_{q}}$.
The problem can then be solved by linearization whenever $\mathcal{N}_{b}^{\mathbb{F}_{q}} \geq \mathcal{M}_{b}^{\mathbb{F}_{q}}-1$. The complexity of solving the system is $T(m, N, \ell, t)=\mathcal{O}\left(\mathcal{N}_{b}^{\mathbb{F}_{q}}\left(\mathcal{M}_{b}^{\mathbb{F}_{q}}\right)^{\omega-1}\right)$.

Moreover, $\left[\mathrm{BBB}^{+} 23\right]$ introduce a hybrid strategy. Compared to $\mathrm{BBC}+-\mathrm{Hybrid}$ it randomly picks matrices from $\mathrm{GL}_{N}\left(\mathbb{F}_{q}\right)$ to randomly compute $\mathbb{F}_{q}$-linear combinations of the entries of the error vector and applies the same transformation to the generator matrix, hoping to achieve that the last $a$ positions of the error vector are all 0 and then shortening the code while also reducing the dimension.

This technique has complexity

$$
\begin{equation*}
\min _{a \geq 0}\left(q^{t a} \cdot T(m, N-a, \ell-a, t)\right) \tag{15}
\end{equation*}
$$

### 6.1 Lowering the attack costs beyond the formulas stated

The combinatorial attacks GRS and AGHT perform best for longer codes, however, algebraic attacks that turn each column into a new variable perform best with fewer variables. For each attack strategy we search for the best number of
columns that we should consider in order to obtain the cheapest cost of a successful break of REDOG. This is why we presented the above formulas using $N$ rather than the full code length $2 n-k$. The conditions given above determine the minimum length required relative to dimension and rank of the error.

We then evaluate the costs for each algorithm for each choice of length $N=$ $\ell+t+i$, for every value of $i=0,1, \ldots, 2 n-k-\ell-t$ satisfying the conditions of the attacks. Figure 1 shows the different behaviour of the algorithms for fixed $\ell$ and $t$ and increasing $i$. The jump in the $\mathrm{BBB}+$ plot is at the transition between the two formulas.


Fig. 1. Plots showing the $\log _{2}$ of the costs for AGHT and BBB+ for the parameters at the 128-bit security level for different choices of code length.

We point out that $\left[\mathrm{BBC}^{+} 20\right]$ also considered decreasing the length of the code for the case of overdetermined systems, see [ $\mathrm{BBC}^{+} 20$, Section 4.2] on puncturing the code in the case of "super"-overdetermined systems. We perform a systematic scan for all algorithms as an attacker will use the best possible attack.

The recomputed values We computed complexity costs for all the attacks introduced in the previous subsection, taking into consideration two values of matrix multiplication exponent, namely $\omega=2.807$ and $\omega=2.37$. For each possible length $N+i$ for $N=\ell+t$ and $i=0,1, \ldots, 2 n-k-\ell-t$ we computed the costs for each attack strategy, keeping the lowest value per strategy. For the two cases of BBB+ and the three strategies described for the BBC+-* algorithms, we selected the best complexity among them. For the sake of completeness, we report the value of $i$ in Table 4 as well and the value of $a$ for $\left[\mathrm{BBB}^{+} 23\right]$. All the values are stated as the $\log _{2}$ of the costs resulting from the complexity formulas. The lowest costs of the best algorithm are stated in blue. Note the above-mentioned caveats regarding evaluating big- $\mathcal{O}$ estimates for concrete parameters.

As shown in the tables, suggested parameters of REDOG for 128 and 192 levels of security do not resist BBC+ attack and Mixed-attack for any choice of $\omega$, and $\mathrm{BBB}+$ for $\omega=2.37$. Suggested parameters for level 256 resist all

| Algorithm | Formula | 128 level |  |  | 192 level |  |  | 256 level |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\omega=2.807$ | $\omega=2.37$ | $i$ | $\omega=2.807$ | $\omega=2.37$ | $i$ | $\omega=2.807$ | $\omega=2.37$ | $i$ |
| GRS [GRS16] | 7 | 228.03 | - | 36 | 392.30 | - | 48 | 604.07 | - | 60 |
| GRS-alg [GRS16] | 8 | 207.88 | - | 36 | 368.18 | - | 48 | 595.97 | - | 60 |
| AGHT [AGHT18] | 9 | 186.68 | - | 37 | 337.69 | - | 49 | 536.22 | - | 61 |
| $\mathrm{BBB}+\left[\mathrm{BBB}^{+} 20\right]$ | 10 \& 11 | 140.06 | 118.25 | 33 | 210.26 | 150 | 0 | 269.03 | 227.15 | 0 |
| $\mathrm{BBC}+\left[\mathrm{BBC}^{+} 20\right]$ | 12-14 | 77.83 | 65.73 | 33 | 175.72 | 159.57 | 48 | 337.92 | 318.01 | 61 |
| Mixed [ $\left.\mathrm{BBB}^{+} 23\right]$ | 15 | 80.94 | 68.61 | 32 | 166.67 | 149.49 | 49 | 347.38 | 311.77 | 61 |

Table 4. Values of the $\log _{2}$ of attack costs for REDOG's suggested parameters for all security level (see Table 1).
attacks except BBB+ for $\omega=2.37$. In Section 8 we propose a solution to the decryption failures that also boosts the security of REDOG.

## $7 \quad$ Solving decryption failures

The core of REDOG's decryption failures is given by point (3) of the proof of Theorem 4.7. Indeed, the crucial step for showing decoding failure of the decoder $\Phi$, is that $\left\langle\mathbf{e}_{1} P^{-1}\right\rangle \not \subset\left\langle-\mathbf{e}_{2} S^{-1}\right\rangle$.

In order to solve the issue of decryption failures in REDOG, we propose an alternative that keeps the random choice of an error vector $\mathbf{e}$ with $\mathrm{wt}_{R}(\mathbf{e})=t$ and changes the public key. The idea is to retain the method introduced in [Loi17], but also to make sure that $\mathrm{wt}_{R}\left(\mathrm{e}^{\prime}\right) \leq \lambda t$. We suggest to pick $P^{-1} \in \mathrm{GL}_{n}(\Lambda)$ randomly instead of it being an isometry of the space $\mathbb{F}_{q^{m}}^{n}$.

The proof of the next result is an adaptation of the proof of Proposition 4.1.

Proposition 7.1. Let $\Lambda \subset \mathbb{F}_{q^{m}}$ be a $\lambda$-dimensional subspace of $\mathbb{F}_{q^{m}}$ and $\mathbf{e}=$ $\left(\mathbf{e}_{1}, \mathbf{e}_{2}\right)$ a random vector with $\mathrm{wt}_{R}(\mathbf{e})=t$ with $\mathbf{e}_{1} \in \mathbb{F}_{q^{m}}^{n}$ and $\mathbf{e}_{2} \in \mathbb{F}_{q^{m}}^{n-k}$. Let $S^{-1} \in \mathrm{GL}_{n-k}(\Lambda)$ and $P^{-1} \in \mathrm{GL}_{n}(\Lambda)$. Then $\left\langle\mathbf{e}_{1} P^{-1},-\mathbf{e}_{2} S^{-1}\right\rangle \subseteq V$ for some $\lambda t$-dimensional $\mathbb{F}_{q}$-linear vectorspace $V$.

Proof. Let $\Gamma=\langle\mathbf{e}\rangle$ be the $\mathbb{F}_{q}$-linear subspace of $\mathbb{F}_{q^{m}}$ generated by e. As before we can write $\Gamma=\left\langle y_{1}, \ldots, y_{t}\right\rangle$. Write also $\Lambda=\left\langle\alpha_{1}, \ldots, \alpha_{\lambda}\right\rangle$. As in the proof of Proposition 4.1 we can express the $j$-th coordinate of $\mathbf{e}_{1} P^{-1}$ as a linear combination of the $\lambda t$ elements $y_{h} \alpha_{k}$ for $h=1, \ldots, t$ and $k=1, \ldots, \lambda$ as $\left(\mathbf{e}_{1} P^{-1}\right)_{j}=\sum_{h=1}^{t} \sum_{k=1}^{\lambda} c_{h, k} y_{h} \alpha_{k}$. The same can be done for each coordinate of $-\mathbf{e}_{2} S^{-1}$. Hence both subspaces are contained in the space $V=\left\langle y_{h} \alpha_{k}\right\rangle$ generated by these $\lambda t$ elements.
Corollary 7.2. Let $\mathbf{e}^{\prime}=\left(\mathbf{e}_{1} P^{-1},-\mathbf{e}_{2} S^{-1}\right)$ with $\mathbf{e}, P^{-1}$ and $S^{-1}$ as in Proposition 7.1. Then $\mathrm{wt}_{R}\left(\mathbf{e}^{\prime}\right) \leq \lambda t$.

The only change to the specification of REDOG is in the KeyGen algorithm in Step 3; encryption and decryption remain unchanged as in Section 3. Here is KeyGen for the updated version of REDOG with no decryption failures.

1. Select $H=\left(H_{1} \mid H_{2}\right), H_{2} \in \mathrm{GL}_{n-k}\left(\mathbb{F}_{q^{m}}\right)$, a parity check matrix of a $[2 n-$ $k, n]$ Gabidulin code, with syndrome decoder $\Phi$ correcting $r$ errors.
2. Select a full rank matrix $M \in \mathbb{F}_{q^{m}}^{\ell \times n}$.
3. Select a $\lambda$-dimensional subspace $\Lambda \subset \mathbb{F}_{q^{m}}$, seen as $\mathbb{F}_{q^{-}}$-linear space, and select $S^{-1} \in \mathrm{GL}_{n-k}(\Lambda)$ and $P^{-1} \in \mathrm{GL}_{n}(\Lambda)$.
4. Compute $F=M P^{-1} H_{1}^{T}\left(H_{2}^{T}\right)^{-1} S$ and publish the public key $\mathrm{pk}=(M, F)$. Store the secret key sk $=(P, H, S, \Phi)$.

Theorem 7.3. The updated version of $R E D O G$ is correct.
Proof. The correctness of the updated version of REDOG follows from the correctness of the original version, except for decryption correctness, which is proven by Corollary 7.2.

## 8 Solving decryption failures and boosting security

Our second idea of how to deal with REDOG not decrypting correctly is to change how e is sampled. While the approach in Section 7 works and preserves all considerations regarding parameter sizes, in Section 6 we have shown that these are too small to offer security against the best known attacks. The approach in this section provides a functioning system and increases the security offered by the parameters.

Recall that the public key is $(M \mid F)$, where $M$ has dimension $\ell \times n$ and $F$ has dimension $\ell \times(n-k)$ and both, $M$ and $F$, have full rank. The relative sizes in REDOG are such that $n-k=\ell-1$, so $F$ is just one column short of being square, and $n=\ell+t+1$. The parameters are chosen so that the decryption step can decode errors of rank up to $r$, while encryption in REDOG adds only an error vector of rank $t$ with $r \geq t \lambda$. All parameter sets have $\lambda=3$ and $r=\lambda t=(n-k) / 2$.

Encryption is computed as $\mathbf{c}=\mathbf{m}^{\prime}(M \mid F)+\mathbf{e}$, for $\mathbf{m}^{\prime} \in \mathbb{F}_{q^{m}}^{\ell}$. Decryption requires decoding in the Gabidulin code for error $\left(\mathbf{e}_{1} P^{-1},-\mathbf{e}_{2} S^{-1}\right)$, where $P$ is an isometry and $S^{-1} \in \mathrm{GL}_{n-k}(\Lambda)$. We have shown in Theorem 4.7 that this $\mathbf{e}^{\prime}$ typically has rank larger than $r$, which causes incorrect decoding, for REDOG's choice of $\mathbf{e}$ with $\mathrm{wt}_{R}(\mathbf{e})=t$. Where we proposed changing the definition of $P$ in the previous section to reach a system which has minimal changes compared to REDOG, we now suggest changing the way that $\mathbf{e}$ is chosen.

In particular, we redefine $\mathbf{e}$ to have different rank on the first $n$ positions and the last $n-k$ positions. Let $\mathbf{e}=\left(\mathbf{e}_{1}, \mathbf{e}_{2}\right)$ with $\mathrm{wt}_{R}\left(\mathbf{e}_{1}\right)=t_{1}$ and $\mathrm{wt}_{R}\left(\mathbf{e}_{2}\right)=$ $t_{2}$. This can be achieved by sampling $t_{1}$ random elements from $\mathbb{F}_{q^{m}}$, testing that this achieves rank $t_{1}$ and taking the $n$ positions in $\mathbf{e}_{1}$ as random $\mathbb{F}_{q}$-linear combinations of these $t_{1}$ elements. Because $m$ is significantly larger than $t_{1}$, this finds an $\mathbf{e}_{1}$ of rank $t_{1}$ on first try with high probability. Similarly, we pick $t_{2}$ random elements from $\mathbb{F}_{q^{m}}$ and use their $\mathbb{F}_{q}$-linear combinations for $\mathbf{e}_{2}$.

We keep $P$ being an isometry and $S^{-1} \in \mathrm{GL}_{n-k}(\Lambda)$ as in REDOG. Then the decoding step needs to find an error of rank $t_{1}+\lambda t_{2}$, namely $\mathbf{e}_{1} P^{-1}$ on the first
$n$ positions and $\mathbf{e}_{2} S^{-1}$ on the last $n-k$ positions. This will succeed if

$$
\begin{equation*}
r \geq t_{1}+\lambda t_{2} \tag{16}
\end{equation*}
$$

Hence, we can consider different splits of $r$ to maximize security.

Considerations for extreme choices of $\boldsymbol{t}_{\mathbf{1}}$ and $\boldsymbol{t}_{\mathbf{2}}$ As already explained in Section 6.1, the attacker can consider parts of $\mathbf{c}_{1}$ and $\mathbf{c}_{2}$, for example, the extreme choice of $t_{1}=0$ would mean that $\mathbf{c}_{1}$ is a codeword in the code generated by $M$ and thus $\mathbf{m}^{\prime}$ would be trivially recoverable from $\mathbf{c}_{1}=\mathbf{m}^{\prime} M$ by computing the inverse of an $\ell \times \ell$ submatrix of $M$. Because $\mathbb{F}_{q^{m}}$ is large, almost any choice of submatrix will be invertible.

The other extreme choice, $t_{2}=0$, does not cause such an obvious attack as for the REDOG parameters $F$ has one column fewer than it has rows, meaning that $\mathbf{c}_{2}=\mathbf{m}^{\prime} F$ cannot be solved for $\mathbf{m}^{\prime}$. Hence, at least one position of $\mathbf{c}_{1}$ needs to be included, but that means that we do not have a codeword in the code generated by that column of $M$ and $F$ but a codeword plus an error of rank 1. However, a brute-force attack on this system still succeeds with cost $q^{m}$ as follows:

Let $\bar{F}=\left(M_{i} \mid F\right)$ be the square matrix obtained from taking $M_{i}$, the $i$-th column of $M$, for a choice of $i$ that makes $\bar{F}$ invertible. Most choices of $i$ will succeed. Let $\overline{\mathbf{c}}=\left(c_{1 i}, \mathbf{c}_{2}\right)$, the $i$-th coordinate of $\mathbf{c}_{1}$ followed by $\mathbf{c}_{2}$.

For each $a \in \mathbb{F}_{q^{m}}$ compute $\overline{\mathbf{m}}=(\overline{\mathbf{c}}-(a, 0,0, \ldots, 0)) \bar{F}^{-1}$. Then compute $\overline{\mathbf{e}}=\mathbf{c}-\overline{\mathbf{m}}(M \mid F)$ and check if $\mathrm{wt}_{R}\left(\overline{\mathbf{e}}_{1}\right)=t_{1}$. If so put $\mathbf{m}^{\prime}=\overline{\mathbf{m}}$ and $\mathbf{e}=\overline{\mathbf{e}}$.

The matrix operations in this attack are cheap and can be made even cheaper by observing that $\overline{\mathbf{m}}=\overline{\mathbf{c}} \bar{F}^{-1}-a \mathbf{f}$, for $\mathbf{f}$ the first row of $\bar{F}^{-1}$, and $\overline{\mathbf{e}}=\mathbf{c}-$ $\left(\overline{\mathbf{c}} \bar{F}^{-1}\right)(M \mid F)+a \mathbf{f}(M \mid F)$, where everything including $\mathbf{f}(M \mid F) \in \mathbb{F}_{q^{m}}^{2 n-k}$ is fixed and can be computed once per target $\mathbf{c}$. Note also that only the $\mathbf{c}_{1}$ and $\mathbf{e}_{1}$ parts need to be computed as by construction $\mathbf{e}_{2}=0$. This leaves just $n$ multiplications and additions in $\mathbb{F}_{q^{m}}$ and the rank computation for each choice of $a$. The search over $a \in \mathbb{F}_{q^{m}}$ is thus the main cost for a complexity of $q^{m}$. For all parameters of REDOG this is less than the desired security.

Generalizations of the brute-force attack For $t_{1}=1$, a brute-force attack needs to search over all $a \in \mathbb{F}_{q^{m}}$, up to scaling by $\mathbb{F}_{q^{-}}$-elements, and over all choices of error patterns, where each position of the error is a random $\mathbb{F}_{q}$-multiple of $a$. We need $\ell$ positions from $\mathbf{c}_{1}=\mathbf{m}^{\prime} M+\mathbf{e}_{1}$ to compute a candidate $\overline{\mathbf{m}}^{\prime}$ as in the attack on $t_{1}=0$. Hence, for each $a \in \mathbb{F}_{q^{m}}$ we need to try at most the $q^{\ell}$ patterns for those $\ell$ positions of $\mathbf{e}_{1}$ for a cost of $\left(q^{m}-1\right) q^{\ell} /(q-1)$. For the REDOG parameters, $q=2$ and $m+\ell$ is significantly smaller than the security level. Hence, $t_{1}=1$ is also a bad choice.

Starting at $t_{1}=2$, when there are two elements $a, b \in \mathbb{F}_{q^{m}}$ and error patterns need to consider random $\mathbb{F}_{q}$-linear combinations of these two elements, the attack costs of $\left(q^{m}-1\right)\left(q^{m}-2\right) q^{2 \ell} /\left(2(q-1)^{2}\right)$ grow beyond the more advanced attacks considered in Section 6.1.

Lemma 8.1. In general, the brute-force attack on the left side takes

$$
\binom{q^{m}-1}{t_{1}} q^{t_{1} \ell} /(q-1)^{t_{1}}
$$

steps.
Proof. The error vector on the left, $\mathbf{e}_{1}$, has rank $t_{1}$, this means that there are $t_{1}$ elements $a_{1}, a_{2}, \ldots, a_{t_{1}} \in \mathbb{F}_{q^{m}}$ which are $\mathbb{F}_{q}$-linearly independent. There are $\binom{q^{m}-1}{t_{1}} /(q-1)^{t_{1}}$ such choices up to $\mathbb{F}_{q}$ factors.

Each of the $\ell$ positions takes a random $\mathbb{F}_{q}$-linear combination. For a fixed choice of the $a_{i}$ there are $q^{t_{1} \ell}$ choices for these linear combinations. Combining these quantities gives the result.

Similarly, for $t_{2}=1$ the brute-force attack is no longer competitive, yet less clearly so than for $t_{1}=2$ because $a$ and $b$ appear in separate parts. There are $q^{m}$ candidate choices for $e_{1 i}$ and $\left(q^{m}-1\right) q^{\ell-1} /(q-1)$ candidates for $\mathbf{e}_{2}$. For $q=2$ this amounts to roughly $2^{2 m+\ell-1}$ and $2 m+\ell-1$ is larger than the security level for all parameters in REDOG.

Lemma 8.2. In general, the brute-force attack on the right side takes

$$
q^{m}\binom{q^{m}-1}{t_{2}} q^{t_{2}(\ell-1)} /(q-1)^{t_{2}}
$$

steps.
Proof. There are $q^{m}$ choices for $e_{1 i}$. The result follows by the same arguments as for Lemma 8.1, and taking into account that $\mathbf{e}_{2}$ has length $\ell-1$.

We do not consider other combinations of columns from the left and right as those would lead to higher ranks than these two options. Depending on the sizes of $t_{1}$ and $t_{2}$, Lemma 8.1 or 8.2 gives the better result, but apart from extreme choices these costs are very high.

Finding good choices of $\boldsymbol{t}_{1}$ and $\boldsymbol{t}_{\mathbf{2}}$ We now turn to the more sophisticated attacks and try to find optimal splits of the decoding budget $r$ between $t_{1}$ and $t_{2}$ satisfying (16), to $r \geq t_{1}+\lambda t_{2}$. to make the best attacks as hard as possible. For any such choice, we consider attacks starting from the left with (parts of) $\mathbf{c}_{1}$ and $M$ or from the right with $\mathbf{c}_{2}, F$, and parts of $\mathbf{c}_{1}$ and $M$. The attacks and subattacks differ in how many columns they require, depending on the dimension and rank, and we scan the whole range of possible lengths from both sides.

Since $n=\ell+t+1$, for the $t$ parameter in REDOG, for small choices of $t_{1} \leq t$ the attack may take a punctured system on $\mathbf{c}_{1}$ and $M$ to recover $\mathbf{m}^{\prime}$, similar to the attacks considered in Section 6, or include part of $\mathbf{c}_{2}$ and $F$, while accepting an error of larger rank including part of $t_{2}$. Hence, the search from the left may start with puncturing of $\mathbf{c}_{1}$. Once parts of $\mathbf{c}_{2}$ are included, the rank typically

| parameter set | best attack | $\log _{2}($ cost | $N+i$ | $t_{1}$ | $t_{2}$ | $m$ | $n$ | $k$ | $\ell$ |
| :--- | ---: | ---: | ---: | ---: | ---: | ---: | ---: | ---: | ---: |
| 128-bit | brute-force | 320.00 | - | 12 | 2 | 83 | 44 | 8 | 37 |
| 192-bit | BBB+ | 458.25 | 61 | 15 | 3 | 109 | 58 | 10 | 49 |
| 256-bit | BBB+ | 628.20 | 75 | 21 | 3 | 135 | 72 | 12 | 61 |

Table 5. Best parameter choices and achieved security for $\omega=2.807$, using the original values for $\ell, k, m$, and $n$ and splitting the decoding capacity $r$ according to $r \geq t_{1}+\lambda t_{2}$.
increases by one for each extra position, again because $m$ is much larger than $t_{1}$ and $t_{2}$, until reaching $t_{1}+t_{2}$, after which the rank does not increase with increasing length.

If $t_{1}>t+1$ parts of $\mathbf{c}_{2}$ need to be considered in any case, with the corresponding increases in the rank of the error, in turn requiring more positions to deal with the increased rank, typically reaching $t_{1}+t_{2}$ before enough positions are available.

Starting from the right, the attacker will always need to include parts from $\mathbf{c}_{1}$ to even have an invertible system. Hence, the attack is hardest for $t_{1}$ maximal in (16) provided that the brute-force attack is excluded. This suggests choosing $t_{2}=1, t_{1}=r-\lambda$, as then the attacker is forced to decode an unstructured code with an error of rank $t_{1}+t_{2}=r-\lambda+1$.

A computer search, evaluating all attacks considered in Section 6 for all choices of $t_{2} \in\{1,2, \ldots, r / \lambda-1\}$ and considering both directions as starting points for the attacker confirms that $t_{2}=1$ is optimal. See online for the Sage code used for the search. The original parameters choices for REDOG then provide the attack costs in Table 5.

This means that this second idea solves decryption failures and takes the parameters of REDOG to a safe level of strength. Actually our optimized choice of $t_{1}$ and $t_{2}$ allows enough margin to shrink the other system parameters.

Note that, as pointed out before, these computations use big- $\mathcal{O}$ complexity estimates and put all constants to 1 and lower-order terms to 0 . This is in line with how estimates are presented in the papers introducing $\mathrm{BBB}+\left[\mathrm{BBB}^{+} 20\right]$ and $\mathrm{BBC}+\left[\mathrm{BBC}^{+} 20\right]$ but typically underestimates the security.

Remark 8.3. After we developed this idea but before posting it, the REDOG authors informed us that they fixed the decryption issue in a manner similar to the approach in this section, namely by having different ranks for $\mathbf{e}_{1}$ and $\mathbf{e}_{2}$. Their choice of $t_{1}=r / 2$ and $t_{2}=r /(2 \lambda)$ satisfies $r \geq t_{1}+\lambda t_{2}$. but provides less security against attacks. The Sage script gives the results in Table 6 as a byproduct of computing the costs for all values of $t_{2}$.

## 9 Conclusions and further considerations

In this paper we showed several issues with the REDOG proposal but also some ways to repair it. One other issue is that REDOG has rather large keys for
a rank-metric-based system. A strategy used by many systems in the NIST post-quantum competition, is to generate parts of the secret and public keys from seeds and storing or transmitting those seeds instead of the matrices they generated. Implementations written in C always need to define ways to take the output of a random-number generator and this strategy includes the use of a fixed such generator into the KeyGen, encryption, and decryption steps. For REDOG, this approach permits to reduce the size of the secret key sk and, at the same time, moderately shrink the size of the public key pk.

Let $f:\{0,1\}^{256} \rightarrow\{0,1\}^{*}$ be such a generator, where $\{0,1\}^{*}$ indicates that the output length is arbitrary, in a use of $f$ the output length $N$ must be specified. Most recent proposals use SHAKE-256 or SHAKE-512. The idea is to pick a random 256 -bit seed $s$ and initialize $f$ with this seed, the output bits of $f(s)$ are then used in place of the regular outputs of the random-number generator to construct elements of the public or secret key. This method is beneficial if $s$ is much smaller than the key element it replaces. The downside is that any use of that key element then incurs the costs of recomputing that element from $s$.

As one of the more interesting cases, we show how to build the isometry $P$ form $f(s)$ for some seed $s$. Let $(n, k, \ell, q, m, \lambda)$ denote the same quantities as in REDOG.

Example 9.1. Let $N=\left(n^{2}+m\right)\left\lceil\log _{2}(q)\right\rceil+256$ and let $\left\{\alpha_{1}, \ldots, \alpha_{m}\right\}$ be a basis of $\mathbb{F}_{q^{m}}$ over $\mathbb{F}_{q}$. Choose a random seed $s$ and produce the $N$-bit string $f(s)$. Use the first $n^{2}\left\lceil\log _{2}(q)\right\rceil$ bits of $f(s)$ to determine $n^{2}$ elements in $\mathbb{F}_{q}$ and build an $n \times n$ matrix $Q$ with these elements. The matrix $Q$ is invertible with probability roughly 0.29 . If this is not the case, use the last 256 bits of the output as a new seed $s^{\prime}$, discard $s$, and repeat the above with $f\left(s^{\prime}\right)$ (an average of 3 trials produces an invertible matrix).

Once an invertible $Q$ has been constructed, use the middle $m\left\lceil\log _{2} q\right\rceil$ bits of $f(s)$ to define $m$ coefficients in $\mathbb{F}_{q}$ and to determine an element $\gamma \in \mathbb{F}_{q^{m}}$ as the $\mathbb{F}_{q}$-linear combination of the $\alpha_{i}$. Then compute $P=\gamma Q$ which, by Theorem 2.4 is an isometry for the rank metric.

As a second example we show how to select $S$.
Example 9.2. We first observe that $\mathbb{F}_{q^{m}}$ is a large finite field, so any choice of $\lambda$ elements for $\lambda \ll m$ will be $\mathbb{F}_{q}$-linearly independent with overwhelming probability. Using $N=\left(m+(n-k)^{2}\right) \lambda\left\lceil\log _{2}(q)\right\rceil$ we can determine $\lambda$ random

| Intended security in bits | 128 | 192 | 256 |
| :--- | ---: | ---: | ---: |
| Achieved security in bits $(\omega=2.807)$ | 271.75 | 384.03 | 500.50 |
| Number of columns $(N+i)(\omega=2.807)$ | 46 | 61 | 76 |
| Achieved security in bits $(\omega=2.37)$ | 229.45 | 324.24 | 422.58 |
| Number of columns $(N+i)(\omega=2.37)$ | 46 | 61 | 76 |

Table 6. Results for the modified parameter for REDOG using $t_{1}=r / 2$ and $t_{2}=$ $r /(2 \lambda)$. The stated costs are achieved by BBB + at length $N+i$.
elements from $\mathbb{F}_{q^{m}}$ which define the subspace $\Lambda \subset \mathbb{F}_{q^{m}}$. We then define the $(n-k)^{2}$ entries of $S^{-1} \in \mathrm{GL}_{n-k}(\Lambda)$ as $\mathbb{F}_{q}$-linear combinations over those $\lambda$ elements, using the next $(n-k)^{2} \lambda\left\lceil\log _{2} q\right\rceil$ bits. The resulting matrix is almost certainly invertible and permits computing $S=\left(S^{-1}\right)^{-1}$.

Similar strategies can be applied to compute the matrices $M, H_{1}$ and $H_{2}$. Let $s_{P}, s_{S}, s_{M}, s_{H_{1}}, s_{H_{2}}$ be the seeds corresponding to the matrices $P, S, M, H_{1}$ and $H_{2}$, respectively. Then we can set $\mathrm{sk}=\left(s_{P}, s_{S}, s_{H_{1}}, s_{H_{2}}\right)$ and $\mathrm{pk}=\left(s_{M}, F\right)$ where $F=M P^{-1} H_{1}^{T}\left(H_{2}^{T}\right)^{-1} S$. This approach cannot be used to compress $F$ as it depends on the other matrices. In this way we reduced the private key size of REDOG to 1024 bits and public key of size of REDOG to $256+\ell(n-k) m\left\lceil\log _{2}(q)\right\rceil$. For the 128 -bit-security level, we obtain a secret key size of 0.13 KB compared to the original 1.45 KB and a public key size of $13,85 \mathrm{~KB}$, compared to the original $14,25 \mathrm{~KB}$ (which was obtained by choosing $M$ to be a circulant matrix) at the expense of having to recompute the matrices from their seeds when needed. Given that matrix inversion over $\mathbb{F}_{q^{m}}$ is not fast, implementations may prefer to include $S$ and $S^{-1}$ in sk and use seeds for the other matrices. To save even more space, it is possible to replace $s_{P}, s_{S}, s_{M}, s_{H_{1}}, s_{H_{2}}$ by a single seed $s$ and generating those five seeds as a call to $f(s)$. The public key then includes the derived value $s_{M}$ but the secret key consists only of $s$. Note that in that case each non-invertible $Q$ will be generated for each run expanding the secret seed, before finding the $Q$ and $P$ that were used in computing pk. In summary, this strategy provides a tradeoff between size and computing time.

## References

$\mathrm{AAB}^{+}$17a. Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillipe Gaborit, Adrien Hauteville, and Gilles Zémor. Ouroboros-R. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/ post-quantum-cryptography-standardization/round-1-submissions.
$\mathrm{AAB}^{+}$17b. Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillippe Gaborit, and Gilles Zémor. RQC. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/ post-quantum-cryptography-standardization/round-1-submissions.
$\mathrm{AASA}^{+}$20. Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. Status report on the second round of the NIST post-quantum cryptography standardization process. NIST IR 8309, 2020. https://doi.org/10.6028/NIST.IR. 8309.
$\mathrm{ABD}^{+} 17 \mathrm{a}$. Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, and Gilles Zémor. LAKE. Technical report, National Institute of Standards and Technology, 2017. available at
https://csrc.nist.gov/projects/post-quantum-cryptography/ post-quantum-cryptography-standardization/round-1-submissions.
$\mathrm{ABD}^{+} 17 \mathrm{~b}$. Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, and Gilles Zémor. LOCKER. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/ post-quantum-cryptography-standardization/round-1-submissions.
AGHT18. Nicolas Aragon, Philippe Gaborit, Adrien Hauteville, and Jean-Pierre Tillich. A new algorithm for solving the rank syndrome decoding problem. In ISIT, pages 2421-2425. IEEE, 2018.
$\mathrm{BBB}^{+}$20. Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, Vincent Neiger, Olivier Ruatta, and Jean-Pierre Tillich. An algebraic attack on rank metric code-based cryptosystems. In Anne Canteaut and Yuval Ishai, editors, EUROCRYPT 2020, Part III, volume 12107 of LNCS, pages 6493, Zagreb, Croatia, May 10-14, 2020. Springer, Heidelberg, Germany.
$\mathrm{BBB}^{+}$23. Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, and JeanPierre Tillich. Revisiting algebraic attacks on minrank and on the rank decoding problem. Designs, Codes and Cryptography, pages 1-37, 072023.
$\mathrm{BBC}^{+}$20. Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray A. Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier A. Verbel. Improvements of algebraic attacks for solving the rank decoding and MinRank problems. In Shiho Moriai and Huaxiong Wang, editors, ASIACRYPT 2020, Part I, volume 12491 of $L N C S$, pages 507-536, Daejeon, South Korea, December 7-11, 2020. Springer, Heidelberg, Germany.
Ber03. Thierry P. Berger. Isometries for rank distance and permutation group of Gabidulin codes. IEEE Trans. Inf. Theory, 49(11):3016-3019, 2003.
Del78. Philippe Delsarte. Bilinear forms over a finite field, with applications to coding theory. Journal of Combinatorial Theory, Series A, 25(3):226-241, 1978.

DGZ17. Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. Ouroboros: A simple, secure and efficient key exchange protocol based on coding theory. In Tanja Lange and Tsuyoshi Takagi, editors, Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, pages 18-34, Utrecht, The Netherlands, June 26-28, 2017. Springer, Heidelberg, Germany.
Gab85. Ernst M. Gabidulin. Theory of codes with maximum rank distance. Problems of Information Transmission, 21(1):1-12, 1985.
$\mathrm{GKK}^{+}$17. Lucky Galvez, Jon-Lark Kim, Myeong Jae Kim, Young-Sik Kim, and Nari Lee. McNie. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/ post-quantum-cryptography-standardization/round-1-submissions.
GPT91. Ernst M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov. Ideals over a non-commutative ring and thier applications in cryptology. In Donald W. Davies, editor, EUROCRYPT'91, volume 547 of $L N C S$, pages 482-489, Brighton, UK, April 8-11, 1991. Springer, Heidelberg, Germany.
GRS16. Philippe Gaborit, Olivier Ruatta, and Julien Schrek. On the complexity of the rank syndrome decoding problem. IEEE Transactions on Information Theory, 62(2):1006-1019, 2016.

KHL $^{+} 22$ a. Jon-Lark Kim, Jihoon Hong, Terry Shue Chien Lau, YounJae Lim, Chik How Tan, Theo Fanuela Prabowo, and Byung-Sun Won. REDOG. Submission to KpqC Round 1, 2022.
KHL ${ }^{+} 22$ b. Jon-Lark Kim, Jihoon Hong, Terry Shue Chien Lau, YounJae Lim, Chik How Tan, Theo Fanuela Prabowo, and Byung-Sun Won. REDOG and its performance analysis. Cryptology ePrint Archive, Report 2022/1663, 2022. https://eprint.iacr.org/2022/1663.

KKGK21. Jon-Lark Kim, Young-Sik Kim, Lucky Erap Galvez, and Myeong Jae Kim. A modified Dual-Ouroboros public-key encryption using Gabidulin codes. Appl. Algebra Eng. Commun. Comput., 32(2):147-156, 2021.
Loi17. Pierre Loidreau. A new rank metric codes based encryption scheme. In Tanja Lange and Tsuyoshi Takagi, editors, Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, pages 3-17, Utrecht, The Netherlands, June 26-28, 2017. Springer, Heidelberg, Germany.
LPR23. Tanja Lange, Alex Pellegrini, and Alberto Ravagnani. On the security of REDOG. Cryptology ePrint Archive, Paper 2023/1205, 2023. https: //eprint.iacr.org/2023/1205.
LT18. Terry Shue Chien Lau and Chik How Tan. Key recovery attack on McNie based on low rank parity check codes and its reparation. In Atsuo Inomata and Kan Yasuda, editors, IWSEC 18, volume 11049 of $L N C S$, pages 19-34, Sendai, Japan, September 3-5, 2018. Springer, Heidelberg, Germany.
LTP21. Terry Shue Chien Lau, Chik How Tan, and Theo Fanuela Prabowo. On the security of the modified Dual-Ouroboros PKE using Gabidulin codes. Appl. Algebra Eng. Commun. Comput., 32(6):681-699, 2021.
Ove05. Raphael Overbeck. A new structural attack for GPT and variants. In Mycrypt, volume 3715 of Lecture Notes in Computer Science, pages 50-63. Springer, 2005.
Ove08. R. Overbeck. Structural attacks for public key cryptosystems based on Gabidulin codes. Journal of Cryptology, 21(2):280-301, April 2008.
Pra62. E. Prange. The use of information sets in decoding cyclic codes. IRE Transactions on Information Theory, 8(5):5-9, 1962.
$\mathrm{S}^{+}$21. W. A. Stein et al. Sage Mathematics Software (Version (9.3)). The Sage Development Team, 2021. http://www.sagemath.org.
Sta11. Richard P. Stanley. Enumerative Combinatorics, volume 1 of Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2 edition, 2011.

# Distinguisher and Related-Key Attack on HALFLOOP-96 

Jinpeng Liu ${ }^{1,2}$ and Ling $\operatorname{Sun}(\square)^{1,2,3}$<br>${ }^{1}$ Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China<br>2 School of Cyber Science and Technology, Shandong University, Qingdao, China<br>${ }^{3}$ Quan Cheng Shandong Laboratory, Jinan, China<br>lingsun@sdu.edu.cn


#### Abstract

HALFLOOP-96 is a 96 -bit tweakable block cipher used in high frequency radio to secure automatic link establishment messages. In this paper, we concentrate on its differential properties in the contexts of conventional, related-tweak, and related-key differential attacks. Using automatic techniques, we determine the minimum number of active S-boxes and the maximum differential probability in each of the three configurations. The resistance of HALFLOOP-96 to differential attacks in the conventional and related-tweak configurations is good, and the longest distinguishers in both configurations consist of five rounds. In contrast, the security of the cipher against differential attacks in the related-key configuration is inadequate. The most effective related-key distinguisher we can find spans eight rounds. The 8-round related-key differential distinguisher is then utilised to initiate a 9 -round weak-key attack. With $2^{92.96}$ chosen-plaintexts, 38.77 -bit equivalent information about the keys can be recovered. Even though the attack does not pose a significant security threat to HALFLOOP-96, its security margin in the related-key configuration is exceedingly narrow. Therefore, improper use must be avoided in the application.


Keywords: Differential cryptanalysis • Related-tweak • Related-key • HALFLOOP-96.

## 1 Introduction

HALFLOOP is a family of tweakable block ciphers. It was created to encrypt protocol data units before transmission during automatic link establishment (ALE). HALFLOOP has been standardised in the most recent revision of MIL-STD-188141D [1], the interoperability and performance standards for medium and high frequency radio systems issued by the United States Department of Defence.

The three versions of HALFLOOP, namely HALFLOOP-24, HALFLOOP48, and HALFLOOP-96, possess the same key size of 128 bits while exhibiting differing state sizes of 24 bits, 48 bits, and 96 bits, correspondingly. The three variants of HALFLOOP are used in various generations of ALE systems: HALFLOOP-24 in the second generation (2G) system, HALFLOOP-48 in the
third generation (3G) system, and HALFLOOP-96 in the fourth generation (4G) system.

The announcement of HALFLOOP is not accompanied by a public cryptanalysis. Dansarie et al. [12] presented the first public cryptanalytic result on HALFLOOP-24 and proposed a number of differential attacks [5] for ciphertextonly, known-plaintext, chosen-plaintext, and chosen-ciphertext scenarios. Despite having a 128 -bit key size, the results of the attack indicate that HALFLOOP24 is incapable of providing 128 -bit security. Note that [12] only assesses the security of HALFLOOP-24 and does not examine the security of the other two variants.

Despite the fact that many HALFLOOP operations are derived from AES [2], HALFLOOP-96 is the most similar to AES of the three HALFLOOP variants. It is common knowledge that AES is susceptible to relate-key differential attacks, and full-round attacks on AES-192 and AES-256 are proposed in [6,7]. Consequently, the similarity between AES and HALFLOOP-96 drives us to investigate the security of HALFLOOP-96 in the context of related-key differential attacks.

### 1.1 Our Results

Motivated by recognising the resistance of HALFLOOP-96 to differential attack in the relate-key setting, we examine its differential property in the contexts of conventional, related-tweak, and related-key differential attacks. Automatic methods based on the Boolean satisfiability problem (SAT) are employed to find the lower bound on the number of active S-boxes and the upper bound on the differential probability for each of the three configurations.

* The resistance of HALFLOOP-96 to standard differential attacks is acceptable. The longest distinguisher with a probability above $2^{-95}$ covers five rounds. The probability of the optimal 5 -round differential characteristic is $2^{-92}$, whereas the accumulated probability of the best 5 -round differential we can discover is $2^{-89.18}$. Due to the limited accumulated effect of differential characteristics, there is no effective 6-round distinguisher.
* Comparing the security of HALFLOOP-96 in the related-tweak setting to the security of the cipher in the conventional differential setting, there is no significant decline. The bounds on the active S-boxes and differential probability in the related-tweak setting are identical to those in the conventional setting, commencing from the sixth round. For more than five rounds, the differential characteristics returned by the SAT solver are the same as those with zero tweak differences. Therefore, starting with the sixth round, the performance of related-tweak differential characteristics is not superior to that of traditional differential characteristics.
* In the related-key setting, HALFLOOP-96 has a low resistance to differential attack. The maximum number of rounds covered by a related-key differential characteristic is eight. The probability of the unique 8-round related-key differential characteristic is $2^{-124}$, whereas the probability of the key schedule
is $2^{-34}$ and the probability of the round function is $2^{-90}$. The security margin in this case is limited, considering the ten rounds of HALFLOOP-96.

Using the newly discovered 8-round related-key differential distinguisher, we launch a 9-round related-key differential attack to recover partial information about the key pair. It takes $2^{92.96}$ chosen-plaintexts and $2^{92.96} 9$-round encryptions to retrieve 38.77 bits of equivalent key information. The attack has a $90 \%$ success probability and is effective against $2^{94}$ key pairs with a specified difference. Although the attack does not pose an actual security threat to HALFLOOP-96, the security margin of the cipher in the setting for related-key attack is reduced to only one round. Hence, it is crucial to take measures to avoid the improper use of the application.

Outline. Section 2 goes over the target cipher HALFLOOP-96 as well as differential cryptanalysis. Section 3 describes the procedure for developing SAT models to seek for differential distinguishers of HALFLOOP-96. Section 4 provides the differential properties of the cipher in the conventional, related-tweak, and related-key configurations. The 9-round related-key differential on HALFLOOP96 is detailed in Section 5. Section 6 serves as the conclusion of the paper.

## 2 Preliminaries

In this section, the cipher examined in the paper is initially reviewed. Next, the primary concept of differential cryptanalysis is presented.

### 2.1 Description of HALFLOOP-96

HALFLOOP [1] is a tweakable block cipher family with three distinct variants. HALFLOOP- 96 employs 96 -bit blocks and has 128 -bit key $K$ and 64 -bit tweak $T$. Many operations in HALFLOOP-96 are derived from AES [2].

Initialisation After receiving the plaintext $m=m_{0}\left\|m_{1}\right\| \cdots \| m_{11}$, where $m_{i} \in$ $\mathbb{F}_{2}^{8}, 0 \leqslant i \leqslant 11$, the internal state IS is created by setting IS as

$$
I S=\left[\begin{array}{lll}
m_{0} & m_{4} & m_{8} \\
m_{1} & m_{5} & m_{9} \\
m_{2} & m_{6} & m_{10} \\
m_{3} & m_{7} & m_{11}
\end{array}\right] .
$$



Fig. 1. Round function of HALFLOOP-96.

A single encryption round consists of the four operations depicted in Fig. 1: AddRoundKey (ARK), SubBytes (SB), RotateRows (RR), and MixColumns (MC). The encryption process consists of $r=10$ rounds, with the last round replacing the MixColumns operation with AddRoundKey. The definitions of the four operations are as follows.

AddRoundKey (ARK) The round key $R K_{i}$ is bitwise added to the state in the $i$-th round.

SubBytes (SB) An 8-bit S-box $S$ is applied to each byte of the state, which is identical to the S-box used by AES (cf. [2]).

RotateRows (RR) As shown in Fig. 1, this operation rotates the rows of the state to the left by a variable number of bit positions.

MixColumns (MC) This operation is the same as the MixColumn transformation used in AES. The columns of the state are regarded as polynomials over the finite field $\mathbb{F}_{2^{8}}$, with the irreducible binary polynomial denoted as $m(x)=$ $x^{8}+x^{4}+x^{3}+x+1$. Each column is multiplied modulo $x^{4}+1$ by a fixed polynomial $c(x)$ given by $c(x)=3 \cdot x^{3}+x^{2}+x+2$. The aforementioned process can instead be represented as a matrix multiplication utilising the matrix $M$ over $\mathbb{F}_{2^{8}}$. In this case, the matrix $M$ is defined as

$$
M=\left[\begin{array}{llll}
2 & 3 & 1 & 1  \tag{1}\\
1 & 2 & 3 & 1 \\
1 & 1 & 2 & 3 \\
3 & 1 & 1 & 2
\end{array}\right]
$$



Fig. 2. Key schedule of HALFLOOP-96.

Key Schedule The key schedule resembles that of AES-128 closely. Denote $K$ and $T$ as $K_{0}\left\|K_{1}\right\| K_{2} \| K_{3}$ and $T_{0} \| T_{1}$, respectively, where $K_{i}(0 \leqslant i \leqslant 3)$ and $T_{j}(j=0,1)$ are 32 -bit words. $K$ and $T$ are utilised to generate a linear array of 4 -byte words $W_{0}, W_{1}, \ldots, W_{32}$, which are then employed to create the round keys. The first four words are initialised with

$$
W_{0}=K_{0} \oplus T_{0}, W_{1}=K_{1} \oplus T_{1}, W_{2}=K_{2}, W_{3}=K_{3} .
$$

The remaining words are derived using the subsequent two functions.
RotWord The function accepts the input word $a_{0}\left\|a_{1}\right\| a_{2} \| a_{3}$, performs a cyclic permutation, and returns the output word $a_{1}\left\|a_{2}\right\| a_{3} \| a_{0}$.
SubWord The function takes a 4-byte input word and applies the S-box $S$ to each of the four bytes to generate a 4 -byte output word.
Each subsequent word $W_{i}(4 \leqslant i \leqslant 16$ and $i \bmod 4 \neq 0)$ is the XOR of the two preceding words $W_{i-1}$ and $W_{i-4}$. For words in positions $i$ that are a multiple of four, $g=$ SubWord $\circ$ RotWord is applied to $W_{i-1}$ prior to the XOR, and a round constant $\mathrm{Rcon}_{i / 4}$ is XORed with the result. Eight round constants are involved in the key schedule of HALFLOOP-96, which are

$$
\begin{aligned}
& \mathrm{Rcon}_{1}=0 \times 01000000, \mathrm{Rcon}_{2}=0 \times 02000000, \mathrm{Rcon}_{3}=0 \times 04000000 \\
& \mathrm{Rcon}_{4}=0 \times 08000000, \mathrm{Rcon}_{5}=0 \times 10000000, \mathrm{Rcon}_{6}=0 \times 20000000, \\
& \mathrm{Rcon}_{7}=0 \times 40000000, \mathrm{Rcon}_{8}=0 \times 80000000 .
\end{aligned}
$$

To obtain the round keys $R K_{0}, R K_{1}, \ldots$, and $R K_{10}$ for HALFLOOP-96, it is necessary to repackage the 4 -byte words into 12 -byte words. The key schedule is illustrated in Fig. 2.

### 2.2 Differential Cryptanalysis

The concept of differential cryptanalysis was initially introduced by Biham and Shamir [5] at CRYPTO 1990. The fundamental methodology involves using plaintext pairs $\left(P, P^{\prime}\right)$ linked by a constant input difference $\Delta_{\text {in }}$, commonly described as the XOR operation between two plaintexts. The attacker subsequently calculates the difference between the two ciphertexts ( $C, C^{\prime}$ ) to identify a nonrandom occurrence of an output difference $\Delta_{\text {out }}$ with a certain likelihood.

The pair of differences $\left(\Delta_{\mathfrak{i n}}, \Delta_{\text {out }}\right)$ is called a differential. The differential probability of the differential over an $n$-bit primitive $E_{K}$ is computed as

$$
\operatorname{Pr}_{E_{K}}\left(\Delta_{\text {in }}, \Delta_{\text {out }}\right)=\frac{\left\{x \in \mathbb{F}_{2}^{n} \mid E_{K}(x) \oplus E_{K}\left(x \oplus \Delta_{\text {in }}\right)=\Delta_{\text {out }}\right\}}{2^{n}} .
$$

The weight of the differential is determined by taking the negative logarithm of its probability, using a base of two.

The task of evaluating the differential probability of a differential in order to discover a valid differential for a cryptographic algorithm with several iterations is known to be quite challenging. The differential is usually localised by constructing differential characteristics, which enable the tracking of differences occurring after each round. Let $\left(\Delta_{0}=\Delta_{\text {in }}, \Delta_{1}, \ldots, \Delta_{r}=\Delta_{\text {out }}\right)$ be an $r$-round differential characteristic of the given differential $\left(\Delta_{\text {in }}, \Delta_{\text {out }}\right)$. Suppose the $r$-round encryption $E_{K}$ can be represented as the composition of $r$ round functions denoted by $f_{k_{r-1}} \circ f_{k_{r-2}} \circ \cdots \circ f_{k_{0}}$. Given the premise that the round keys $k_{0}, k_{1}$, $\ldots$, and $k_{r-1}$ are independent and uniformly random, the differential probability of the differential characteristic can be calculated as

$$
\operatorname{Pr}_{E_{K}}\left(\Delta_{0}, \Delta_{1}, \ldots, \Delta_{r}\right)=\prod_{i=0}^{r-1} \operatorname{Pr}_{f_{k_{i}}}\left(\Delta_{i}, \Delta_{i+1}\right)
$$

As discussed in [14], a fixed differential might encompass several differential characteristics, and the probability of the differential is determined by aggregating the probabilities associated with each differential characteristic. This probability may be computed as

$$
\operatorname{Pr}_{E_{K}}\left(\Delta_{\text {in }}, \Delta_{\text {out }}\right)=\sum_{\Delta_{1}, \Delta_{2}, \ldots, \Delta_{r-1} \in \mathbb{F}_{2}^{n}} \operatorname{Pr}_{E_{K}}\left(\Delta_{\text {in }}, \Delta_{1}, \ldots, \Delta_{r-1}, \Delta_{\text {out }}\right) .
$$

In practical applications, the comprehensive search for all characteristics inside a differential and the precise calculation of their probabilities are unattainable due to the constraints imposed by limited computational resources. A common way of handling this is to find the differential characteristics with a higher
probability in the differential, and the summation of probabilities of these characteristics approximates the probability of the differential.

After finding an $r$-round differential $\left(\Delta_{\text {in }}, \Delta_{\text {out }}\right)$ with probability $p_{0}\left(p_{0}>\right.$ $\left.2^{1-n}\right)$, we can launch an attack against the $(r+1)$-round encryption $\widehat{E}_{K}=$ $f_{k_{r}} \circ E_{K}$. The following is a summary of the attack procedure.
(1) Select $N$ pairs of plaintexts $\left(P, P^{\prime}\right)$ whose difference $P \oplus P^{\prime}$ equals $\Delta_{\mathrm{in}}$. Query the encryption oracle to obtain pairs of corresponding ciphertexts $\left(C, C^{\prime}\right)$.
(2) Create a counter $\operatorname{Ctr}\left[k_{r}^{(i)}\right]$ for each possible value $k_{r}^{(i)}$ of the subkey $k_{r}, 0 \leqslant$ $i \leqslant 2^{n}-1$. For each pair $\left(C, C^{\prime}\right)$, determine the value of $f_{k_{r}^{(i)}}^{-1}(C) \oplus f_{k_{r}^{(i)}}^{-1}\left(C^{\prime}\right)$ for each $k_{r}^{(i)}$. If the equation $f_{k_{r}^{(i)}}^{-1}(C) \oplus f_{k_{r}^{(i)}}^{-1}\left(C^{\prime}\right)=\Delta_{\text {out }}$ is valid, increment the counter $\mathrm{C} \operatorname{tr}\left[k_{r}^{(i)}\right]$ by one.
(3) If the threshold is set to $\tau$, the key guess $k_{r}^{(i)}$ is sorted into a candidate list only if the counter value $\mathrm{C} \operatorname{tr}\left[k_{r}^{(i)}\right]$ is at least $\tau$.

The counter that keeps track of the number of pairs confirming the differential conforms to the binomial distribution $\mathcal{B}\left(N, p_{0}\right)$ when the correct key guess is made, as the attack procedure specifies. The counter under the wrong key guess follows a binomial distribution $\mathcal{B}(N, p)$, where $p$ is the probability of a pair matching the differential given a wrong key guess, which is equal to $p=2^{1-n}$.

As a statistical cryptanalysis, differential cryptanalysis is inevitably confronted with two errors. The symbol $\varepsilon_{0}$ denotes the likelihood that the candidate list does not include the right key. The likelihood of a key guess that is not correct remaining in the candidate list is represented by the symbol $\varepsilon_{1}$. Hence, the probability of success $\left(P_{S}\right)$ in the attack, denoting the likelihood of the right key being included in the candidate list, may be expressed as $1-\varepsilon_{0}$. When the value of $N$ is sufficiently large, the approximations for $\varepsilon_{0}$ and $\varepsilon_{1}$ may be derived using the methodology presented in [8] as

$$
\begin{align*}
& \varepsilon_{0} \approx \frac{p_{0} \cdot \sqrt{1-(\tau-1) / N}}{\left(p_{0}-(\tau-1) / N\right) \cdot \sqrt{2 \cdot \pi \cdot(\tau-1)}} \cdot \exp \left[-N \cdot D\left(\frac{\tau-1}{N} \| p_{0}\right)\right] \\
& \varepsilon_{1} \approx \frac{(1-p) \cdot \sqrt{\tau / N}}{(\tau / N-p) \cdot \sqrt{2 \cdot \pi \cdot N \cdot(1-\tau / N)}} \cdot \exp \left[-N \cdot D\left(\frac{\tau}{N} \| p\right)\right] \tag{2}
\end{align*}
$$

where $D(p \| q) \triangleq p \cdot \ln \left(\frac{p}{q}\right)+(1-p) \cdot \ln \left(\frac{1-p}{1-q}\right)$ represents the Kullback-Leibler divergence between two Bernoulli distributions with parameters $p$ and $q$.

### 2.3 Related-Key and Related-Tweak Differential Cryptanalysis

One notable distinction between differential cryptanalysis and related-key differential cryptanalysis is the utilisation of differential propagations. In related-key differential cryptanalysis, the focus is on exploiting the differential propagation
while encrypting plaintexts $P$ and $P^{\prime}$ with distinct keys, even if these plaintexts happen to be identical. The formal representation of an $r$-round relatedkey differential is denoted by the triple ( $\Delta_{\text {in }}, \Delta_{\text {out }}, \Delta_{\text {key }}$ ), where $\Delta_{\text {key }}$ signifies the difference between the keys. The probability is calculated as

$$
\operatorname{Pr}_{E_{K}}\left(\Delta_{\text {in }}, \Delta_{\text {out }}, \Delta_{\text {key }}\right)=\frac{\left\{x \in \mathbb{F}_{2}^{n} \mid E_{K}(x) \oplus E_{K \oplus \Delta_{\text {key }}}\left(x \oplus \Delta_{\text {in }}\right)=\Delta_{\text {out }}\right\}}{2^{n}} .
$$

AES is widely acknowledged as vulnerable to related-key differential attacks, as evidenced by the suggested full-round attacks on AES-192 and AES-256 in [6,7]. Given that HALFLOOP-96 has the highest degree of similarity to AES among the three HALFLOOP variations, our focus lies on examining its differential property in the context of a related-key attack.

It is also feasible to initialise related-tweak differential cryptanalysis for tweakable block ciphers. Differential propagation is utilised when $P$ and $P^{\prime}$, which might potentially be identical, are encrypted using the same key and distinct tweaks. The related-tweak differential is denoted by ( $\Delta_{\text {in }}, \Delta_{\text {out }}, \Delta_{\text {tweak }}$ ), where $\Delta_{\text {tweak }}$ signifies the difference between the tweaks. In contrast to related-key differential cryptanalysis, related-tweak differential cryptanalysis is considered a more feasible approach because the adversary knows the value of the tweak.

## 3 Automatic Search of Differential Distinguishers

Identifying a differential with a non-negligible probability is a pivotal and arduous stage in a differential attack. At the EUROCRYPT 1994, Matsui [18] introduced a pioneering approach called the branch and bound algorithm, which offered a systematic methodology for investigating the best differential characteristic. When considering tailored optimisations for certain ciphers, it is indisputable that branch and bound algorithms exhibit high efficiency [13]. However, the ability to prevent memory overflow through the precise selection of search nodes is a challenge requiring proficiency in cryptanalysis and programming.

The introduction of automatic search techniques [19] has dramatically simplified the process of identifying differential characteristics. The main aim is to transform the task of finding differential characteristics into some well-studied mathematical problems. With some publicly accessible solvers for these mathematical problems, the optimal differential characteristics can be identified. Due to its relatively straightforward implementation, automatic approaches have been widely employed in the search for distinguishers in various attacks.

The mathematical problems that are commonly encountered include mixed integer linear programming (MILP), Boolean satisfiability problem (SAT), satisfiability modulus theories (SMT), and constraint satisfaction problem (CSP). The classification of automatic search methods is based on the mathematical issues they address. The search for differential characteristics in ciphers with 8 -bit S-boxes may be conducted using MILP method as described in $[3,9,15]$, SAT method as described in [4,23], and SMT method as described in [16]. In
this study, the SAT method proposed in [23] is chosen for efficiently generating SAT models for S-boxes.

This section provides a comprehensive description of the SAT models necessary for searching for differential characteristics of HALFLOOP-96.

### 3.1 Boolean Satisfiability Problem

A Boolean formula is comprised of Boolean variables, the operations AND (conjunction, $\wedge$ ), OR (disjunction, $\vee$ ), and NOT (negation, $\cdot$ ), and brackets. The Boolean satisfiability problem (SAT) pertains to ascertaining the existence of a valid assignment for all Boolean variables such that the given Boolean formula holds. If this condition is met, the formula is known as satisfiable. In the absence of such a designated task, the formula in question is considered unsatisfiable. SAT is the first problem proven to be NP-complete [11]. However, significant advancements have been made in developing efficient solvers capable of handling a substantial volume of real-world SAT problems.

This work employs the solver CryptoMiniSat [21] for distinguisher search. CryptoMiniSat necessitates that Boolean formulae be expressed in conjunctive normal form (CNF), whereby many clauses are made in conjunction with each other, and each clause consists of a disjunction of variables, which may be negated. CryptoMiniSat additionally provides support for XOR clauses that are formed of XOR operations on variables. This feature greatly simplifies the process of constructing models for HALFLOOP-96. Converting distinguisher searching problems into Boolean formulae is critical in developing automatic models.

### 3.2 SAT Models for Linear Operations of HALFLOOP-96

For the $m$-bit vector $\Delta$, the $i$-th bit $(0 \leqslant i \leqslant m-1)$ is denoted by $\Delta[i]$, while $\Delta[0]$ represents the most significant bit.

Model 1 (XOR, [17]) For the m-bit XOR operation, the input differences are represented by $\Delta_{0}$ and $\Delta_{1}$, and the output difference is denoted by $\Delta_{2}$. Differential propagation is valid if and only if the values of $\Delta_{0}, \Delta_{1}$ and $\Delta_{2}$ validate all of the following XOR clauses.

$$
\Delta_{0}[i] \oplus \Delta_{1}[i] \oplus \Delta_{2}[i]=0,0 \leqslant i \leqslant m-1 .
$$

To build the model for the MC operation, we employ the procedure described in [24]. First, the primitive representation [22] $\mathbb{M}$ of the matrix $M$ (cf. Eqn. (1))
is created.
is the matrix representation of $M$ over $\mathbb{F}_{2}$. The notation $\mathbb{M}_{i, j}$ represents the element located in the $i$-th row and $j$-th column of the matrix $\mathbb{M}$. The SAT model can then be constructed using XOR clauses.

Model 2 (Matrix Multiplication) For matrix multiplication with the $32 \times$ 32 matrix $\mathbb{M}$, the input and output differences are represented by $\Delta_{0}$ and $\Delta_{1}$ respectively. Differential propagation is valid if and only if the values of $\Delta_{0}$ and $\Delta_{1}$ satisfy all the XOR clauses in the subsequent.


### 3.3 SAT Model for the S-box of HALFLOOP-48

The method in [23] is utilised to construct the SAT model for the S-box. We commence our analysis with the SAT model that is focused on active S-boxes. In addition to using 16 Boolean variables $\Delta_{0}=\left(\Delta_{0}[0], \Delta_{0}[1], \ldots, \Delta_{0}[7]\right)$ and
$\Delta_{1}=\left(\Delta_{1}[0], \Delta_{1}[1], \ldots, \Delta_{1}[7]\right)$ to represent the input and output differences of the S-box, it is necessary to incorporate an auxiliary Boolean variable denoted as $w$. The value assigned to $w$ is one for active S-boxes and zero for inactive S-boxes, assuming the propagation $\Delta_{0} \rightarrow \Delta_{1}$ is possible. Based on the given criteria, the set
encompasses potential values for $\Delta_{0}\left\|\Delta_{1}\right\| w$. In order to maintain the constraint that $\Delta_{0}\left\|\Delta_{1}\right\| w$ remains within the bounds of the set $\mathcal{V}_{1}$, a clause is generated for each 17 -bit vector $v \notin \mathcal{V}_{1}$,

$$
\bigvee_{i=0}^{7}\left(\Delta_{0}[i] \oplus v[i]\right) \vee \bigvee_{i=0}^{7}\left(\Delta_{1}[i] \oplus v[i+8]\right) \vee(w \oplus v[16])=1
$$

which may serve as a candidate for the SAT model of the S-box. These clauses comprise an initial version of the SAT model for the search oriented to active S-boxes. The use of the initial version of the SAT model without modification would impede the search process of the automatic method due to the large size of the set $\mathbb{F}_{2}^{17} \backslash \mathcal{V}_{1}$, which is $2^{17}-32386=98686$. To reduce the size of the $S$-box model, we employ the Espresso algorithm [10] to simplify the model ${ }^{4}$. The final SAT model oriented to active S-boxes is composed of 7967 clauses.

The SAT model oriented to differential probability can be created similarly. The probabilities of possible differential propagations $\Delta_{0} \rightarrow \Delta_{1}$ for the 8-bit S-box $S$ can take values from the set $\left\{2^{-7}, 2^{-6}, 1\right\}$. Motivated by the two-step encoding method described in [23], we introduce two Boolean variables $u_{0}$ and $u_{1}$ for each S-box to encode the differential probability of possible propagations.

$$
\mathcal{V}_{2}=\left\{\begin{array}{l|l}
\Delta_{0}\left\|\Delta_{1}\right\| u_{0} \| u_{1} & \left.\begin{array}{l}
\Delta_{0}, \Delta_{1} \in \mathbb{F}_{2}^{8}, u_{0}, u_{1} \in \mathbb{F}_{2} \\
u_{0} \| u_{1}= \begin{cases}1 \| 1, & \text { if } \operatorname{Pr}_{S}\left(\Delta_{0}, \Delta_{1}\right)=2^{-7} \\
0 \| 1, & \text { if } \operatorname{Pr}_{S}\left(\Delta_{0}, \Delta_{1}\right)=2^{-6} \\
0 \| 0, & \text { if } \operatorname{Pr}_{S}\left(\Delta_{0}, \Delta_{1}\right)=1\end{cases}
\end{array}\right\}
\end{array}\right.
$$

is an optional set of values that may be assigned to the vector $\Delta_{0}\left\|\Delta_{1}\right\| u_{0} \| u_{1}$. Thus, the weight of a potential propagation can be determined by $u_{0}+6 \cdot u_{1}$. To ensure that $\Delta_{0}\left\|\Delta_{1}\right\| u_{0} \| u_{1}$ never takes values outside of the set $\mathcal{V}_{2}$, we should generate a clause for each 18 -bit $\nu \notin \mathcal{V}_{2}$,

$$
\bigvee_{i=0}^{7}\left(\Delta_{0}[i] \oplus \nu[i]\right) \vee \bigvee_{i=0}^{7}\left(\Delta_{1}[i] \oplus \nu[i+8]\right) \vee\left(u_{0} \oplus \nu[16]\right) \vee\left(u_{1} \oplus \nu[17]\right)=1
$$

[^29]These clauses constitute an initial version of the SAT model oriented to differential probability. ESPRESSO algorithm is once again employed to reduce the size of the model. The final S-box model oriented to differential probability is composed of 8728 clauses.

### 3.4 SAT Model for the Objective Function

We aim to identify differential characteristics that exhibit fewer active S-boxes and high probability. The objective function can be mathematically expressed as $\sum_{i=0}^{\ell} u_{i} \leqslant \vartheta$, where $u_{i}(0 \leqslant i \leqslant \ell)$ are Boolean variables that indicate the activation status of the S-boxes or encode the differential probability of possible propagations for the S-boxes. Let $\vartheta$ denote a predetermined upper limit for either the number of active S-boxes or the weight of the differential characteristics. The sequential encoding method [20] is utilised to transform this inequality into clauses.

Model 3 (Objective Function, [20]) The following clauses provide validity assurance for the objective function $\sum_{i=0}^{\ell} u_{i} \leqslant 0$.

$$
\overline{u_{i}}=1,0 \leqslant i \leqslant \ell .
$$

For the objective function $\sum_{i=0}^{\ell} u_{i} \leqslant \vartheta$ with $\vartheta>0$, it is necessary to incorporate auxiliary Boolean variables $a_{i, j}(0 \leqslant i \leqslant \ell-1,0 \leqslant j \leqslant \vartheta-1)$. The objective function is valid if and only if the following clauses hold.

$$
\left.\left.\begin{array}{l}
\overline{u_{0}} \vee a_{0,0}=1 \\
\overline{a_{0, j}}=1,1 \leqslant j \leqslant \vartheta-1 \\
\overline{u_{i}} \vee a_{i, 0}=1 \\
\overline{a_{i-1,0}} \vee a_{i, 0}=1 \\
\overline{u_{i}} \vee \overline{a_{i-1, j-1}} \vee a_{i, j}=1 \\
\overline{a_{i-1, j}} \vee a_{i, j}=1 \\
\overline{u_{i}} \vee \overline{a_{i-1, \vartheta-1}}=1 \\
\overline{u_{\ell}} \vee \overline{a_{\ell-1, \vartheta-1}}=1
\end{array}\right\} 1 \leqslant j \leqslant \vartheta-1\right\} 1 \leqslant i \leqslant \ell-2 .
$$

### 3.5 Finding More Differential Characteristics

Using the models presented in Sections 3.2 to 3.4 , we can identify differential characteristics with fewer active S-boxes and high probabilities. To improve the probability evaluation of the differential, we should fix the input and output differences in the automatic model and find as many other differential characteristics as feasible. To prevent the solver from returning the same solution after
obtaining a single differential characteristic, we should add a clause to the SAT problem. Assume that $v \in \mathbb{F}_{2}^{\omega}$ is a solution for the $\omega$ Boolean variables $x_{0}, x_{1}$, $\ldots, x_{\omega-1}$ returned by the SAT solver. Two index sets
$\left.v\right|_{0}=\{i \mid 0 \leqslant i \leqslant \omega-1$ s.t. $v[i]=0\}$, and $\left.v\right|_{1}=\{i \mid 0 \leqslant i \leqslant \omega-1$ s.t. $v[i]=1\}$. are generated based on the value of $v$. Adding the clause

$$
\bigvee_{\left.i \in v\right|_{0}} x_{i} \vee \bigvee_{\left.i \in v\right|_{1}} \overline{x_{i}}=1
$$

to the SAT problem guarantees that the solver will not find $v$ again.

## 4 Differential Distinguishers of HALFLOOP-96

This section presents an analysis of the differential characteristics of HALFLOOP96 in three attack settings: conventional, related-tweak, and related-key. These characteristics are determined using the methodology in Section 3.

Table 1. Differential properties of HALFLOOP-96.

| Round | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\# \mathrm{~S}$ | 1 | 5 | 8 | 11 | 14 | 17 | 20 | 23 | 26 | 29 |
| $\# \mathrm{~S}_{\mathrm{T}}$ | 0 | 1 | 3 | 8 | 14 | 17 | 20 | 23 | 26 | 29 |
| $\# \mathrm{~S}_{\mathrm{K}}$ | 0 | 0 | 1 | 5 | 11 | 14 | 16 | 19 | 24 | 29 |
| P | $2^{-6}$ | $2^{-30}$ | $2^{-48}$ | $2^{-70}$ | $2^{-92}$ | $2^{-113}$ | $2^{-134}$ | $2^{-155}$ | $2^{-176}$ | $2^{-197}$ |
| $\mathrm{P}_{\mathrm{T}}$ | 1 | $2^{-6}$ | $2^{-18}$ | $2^{-53}$ | $2^{-91}$ | $2^{-113}$ | $2^{-134}$ | $2^{-155}$ | $2^{-176}$ | $2^{-197}$ |
| $\mathrm{P}_{\mathrm{K}}$ | 1 | 1 | $2^{-6}$ | $2^{-31}$ | $2^{-66}$ | $2^{-87}$ | $2^{-106}$ | $2^{-124}$ | $2^{-154}$ | $2^{-197}$ |

\#S, \#S $\mathrm{S}_{\mathrm{T}}$, and \#S : The number of active S-boxes in conventional, related-tweak, and related-key settings. $P, P_{T}$, and $P_{K}$ : Differential probabilities in conventional, related-tweak, and related-key settings.

### 4.1 Conventional Differential Distinguishers of HALFLOOP-96

The lower bound on the number of active S-boxes and the upper bound on the differential probability are calculated in the standard differential attack scenario. The outcomes of 1 to 10 rounds of HALFLOOP-96 are displayed in Table 1.

Table 2. Information about three 5 -round differentials with probability $2^{-89.18}$.

| Index | Input difference | Output difference |
| :---: | :---: | :---: |
| 1 | $0 \times 000000580600000000660000$ | $0 \times 101030205 f 6 a 3535 \mathrm{e} 8 \mathrm{c} 09 \mathrm{~d} 2 \mathrm{e}$ |
| 2 | $0 \times 060000000066000000000058$ | $0 \times 5 \mathrm{f} 6 \mathrm{a} 3535 \mathrm{e} 8 \mathrm{c} 09 \mathrm{~d} 2 \mathrm{e} 10103020$ |
| 3 | $0 \times 006600000000005806000000$ | $0 \times 88 \mathrm{c} 09 \mathrm{~d} 2 \mathrm{e} 101030205 \mathrm{f} 6 \mathrm{a} 3535$ |

The longest differential characteristic with a probability greater than $2^{-95}$ spans five rounds, and the SAT solver indicates that there are 32075 -round differential characteristics with probability $2^{-92}$. A thorough analysis reveals that the 3207 characteristics stem from 2214 distinct differentials. We search for all differential characteristics in the 2214 differentials with probabilities more significant than $2^{-110}$ by fixing the input and output differences in the automatic search. The largest accumulated probability of the differential is $2^{-89.18}$, and there are three differentials with the highest probability, whose input and output differences are shown in Table 2. Six 5-round characteristics exist in the first differential with the highest probability of $2^{-92}$, as depicted in Fig. 3.


Fig. 3. Six dominated characteristics in the first 5-round differential.

Even though the probability of the optimal 6-round differential characteristic of HALFLOOP-96 is less than $2^{-95}$, we question the existence of 6 -round differentials with accumulated probabilities greater than $2^{-95}$. To find the answer, we first search for all 6 -round differential characteristics with a probability of $2^{-113}$ and determine that 1272 characteristics meet the condition. Note that the 1272 characteristics come from 1017 different differentials. Then, we fix the input and output differences in the automatic search and discover all differential characteristics with probabilities greater than $2^{-135}$ for each of the 1017 differentials. The maximal accumulated probability of 6 -round differentials reaches $2^{-110.87}$, indicating that these differentials cannot support a valid differential attack. The longest differential distinguisher for HALFLOOP-96 comprises five rounds.

### 4.2 Related-Tweak Differential Distinguishers of HALFLOOP-96

The evaluation of active S-boxes and differential probabilities should include the key schedule in the context of a related-tweak attack. Table 1 displays the minimum number of active S-boxes and maximum differential probabilities for one to ten rounds of HALFLOOP-96 in the related-tweak attack configuration.

From the sixth round, the bounds on the active S-boxes and probabilities in the related-tweak setting are identical to those in the conventional setting, as shown in Table 1. The differential characteristics returned by the SAT solver for more than five rounds do not have non-zero tweak differences. Accordingly, beginning with the sixth round, related-tweak differential characteristics do not perform better than conventional ones. Given that the optimal differential in the conventional differential attack setting has already reached five rounds, the advantage of the adversary in the related-tweak setting is insignificant.


Fig. 4. Two 5-round related-tweak differential characteristics with probability $2^{-91}$.

The minor advantage resides in the existence of 5 -round related-tweak differential characteristics with a probability of $2^{-91}$, whereas the probability of the optimal 5 -round characteristic in the conventional setting is $2^{-92}$. We find two 5 -round related-tweak differential characteristics with a probability of $2^{-91}$ using the SAT solver. The probability in the key schedule is $2^{-12}$ and the probability in the round function is $2^{-79}$ for both characteristics. In addition, after searching exhaustively with the automatic procedure for all characteristics with probabilities greater than $2^{-120}$, we are unable to identify a clustering effect for the two characteristics. Figure 4 exhibits the two characteristics.

### 4.3 Related-Key Differential Distinguishers of HALFLOOP-96

In the context of a related-key attack, the calculation of active S-boxes and differential probabilities must consider the key schedule. Table 1 displays the bounds on the active S-boxes and differential probabilities from one to ten cycles of HALFLOOP-96.

Note that in the related-key attack configuration, the characteristics may be utilised in an attack if the probability is greater than $2^{-127}$. According to Table 1, the effective related-key differential characteristic with the most rounds is eight. We verify using the SAT solver that there is only one 8-round related-key differential characteristic with probability $2^{-124}$. Figure 5 illustrates the 8 -round characteristic. The probability in the key schedule is $2^{-34}$, and the probability in the round function is $2^{-90}$. In addition, we do not identify the clustering effect for the 8 -round distinguisher after exhaustively searching for all characteristics with probability no less than $2^{-150}$.


Fig. 5. 8-round related-key differential characteristics with probability $2^{-124}$.

## 5 Related-Key Differential Attack on HALFLOOP-96

In this section, we employ the 8-round related-key differential distinguisher in Section 4.3 to launch a 9-round related-key differential attack on HALFLOOP96. Note that the attack is a weak-key attack, as the probability of the key schedule shown in Fig. 5 is $2^{-34}$. In other words, only one pair of keys out of $2^{34}$ pairs of keys with a difference of $\Delta_{k e y}=0 x a d 0000 f 65 a f 6 f 6 f 75 a f 6 f 60100000000$ is susceptible to the following attack. In this circumstance, a valid attack must ensure the time complexity is less than $2^{94}$.

In the attack, one round is appended after the distinguisher, and the keyrecovery procedure is depicted in Fig. 6. $\mathcal{S}$ structures are prepared for the attack. Each structure contains $2^{80}$ plaintexts, where ten bytes $P[0,3-11]$ of the plaintext $P$ traverse all possible values while the remaining two are fixed to random constants. Then, a single structure can be used to create $2^{79}$ pairs with a difference of $\Delta P=0 x a d 0000 f 6 c 05 \mathrm{df} 6 \mathrm{f} 7 \mathrm{f} 7 \mathrm{f} 6 \mathrm{f} 001$, bringing the total number of pairs to $N=\mathcal{S} \cdot 2^{79}$. Therefore, the data complexity of the attack is $\mathcal{S} \cdot 2^{80}$ chosen-plaintexts.


Fig. 6. 9-round related-key differential attack on HALFLOOP-96.

In the attack, an empty hash table $\mathbb{H}$ is created. For each output pair $\left(O, O^{\prime}\right)$ returned by the encryption oracle, if the conditions

$$
\Delta O[0-2]=0 \mathrm{x} 5 \mathrm{af} 6 \mathrm{f} 6, \Delta O[5] \oplus \Delta O[9]=\Delta O[6] \oplus \Delta O[10]=0 \mathrm{xf} 6
$$

are fulfilled, the quadruple ( $P, P^{\prime}, O[3,4,7,11], O^{\prime}[3,4,7,11]$ ) will be inserted into $\mathbb{H}$ at index $\Delta O[8-10]$. Consequently, $\mathbb{H}$ contains approximately $N \cdot 2^{-40}$ quadruples, and each index $\Delta O[8-10]$ corresponds to approximately $N \cdot 2^{-64}$ quadruples. The index $\Delta O[8-10]$ that renders differential propagation of either $0 x f 6 \rightarrow \Delta O[8] \oplus$ ad or $0 x f 6 \rightarrow \Delta O[9]$ impossible for the S-box is then eliminated from $\mathbb{H}$. After this stage, there are approximately $2^{24} \cdot(127 / 256)^{2}=2^{21.98}$ indexes remaining in $\mathbb{H}$. The time complexity of this phase is dominated by the
time to query the encryption oracle, which corresponds to line 3 of Algorithm 1 and is equivalent to $\mathrm{T}_{\mathrm{L3}}=\mathcal{S} \cdot 2^{79} \cdot 2=\mathcal{S} \cdot 2^{80} 9$-round encryptions.

For each index $\Delta O[8-10]$ in $\mathbb{H}$, we guess the value of $R K_{9}[4]$ and initialise an empty table $\mathbb{T}_{1}$. After deriving the value of $R K_{9}^{\prime}[4]=R K_{9}[4] \oplus \Delta O[8] \oplus 5 \mathrm{a}$, the value of $\Delta X_{8}[4]$ for each quadruple at index $\Delta O[8-10]$ can be computed.

```
Algorithm 1: 9-round related-key differential attack
    Create \(\mathcal{S} \cdot 2^{79}\) pairs \(\left(P, P^{\prime}\right)\) from \(\mathcal{S}\) structures
    Initialise an empty hash table \(\mathbb{H}\)
    Obtain the value of ( \(O, O^{\prime}\) ) for each ( \(P, P^{\prime}\) ) by querying the encryption oracle
    if \(\Delta O[0-2]=0 \times 5\) af6f6 and \(\Delta O[5] \oplus \Delta O[9]=\Delta O[6] \oplus \Delta O[10]=0 \times f 6\) then
        \(\left(P, P^{\prime}, O[3,4,7,11], O^{\prime}[3,4,7,11]\right)\) is inserted into \(\mathbb{H}\) at index \(\Delta O[8-10]\)
    end
    foreach index \(\Delta O[8-10]\) of \(\mathbb{H}\) do
        if \(0 \times \mathrm{xf} 6 \rightarrow \Delta O[8] \oplus\) ad or \(0 \mathrm{xf} 6 \rightarrow \Delta O[9]\) are impossible propagations then
            Remove the index \(\Delta O[8-10]\) from \(\mathbb{H}\)
        else
            foreach 8-bit possible values of \(R K_{9}[4]\) do
            Initialise an empty table \(\mathbb{T}_{1}\)
            Derive \(R K_{9}^{\prime}[4]=R K_{9}[4] \oplus \Delta O[8] \oplus 0 \times 5 \mathrm{a}\)
            foreach ( \(\left.P, P^{\prime}, O[3,4,7,11], O^{\prime}[3,4,7,11]\right)\) at index \(\Delta O[8-10]\) do
                        Compute \(\Delta X_{8}[4]\)
                        if \(\Delta X_{8}[4]=0 x a d\) then
                            Inserted \(\left(P, P^{\prime}, O[3,7,11], O^{\prime}[3,7,11]\right)\) into table \(\mathbb{T}_{1}\)
                            end
            end
            foreach 63 possible values of \(\alpha^{\prime}\) and 127 possible values of \(\zeta\) do
                        foreach 24 -bit possible values of \(R K_{9}[3,7,11]\) do
                            Initialise an empty table \(\mathbb{T}_{2}\)
                                Derive \(R K_{9}^{\prime}[3,7,11]=R K_{9}[3,7,11] \oplus\left(\alpha^{\prime}\left\|\left(\alpha^{\prime} \oplus \zeta\right)\right\| \zeta\right)\)
                                foreach ( \(\left.P, P^{\prime}, O[3,7,11], O^{\prime}[3,7,11]\right)\) in \(\mathbb{T}_{1}\) do
                                    Compute \(\Delta X_{8}[3,7,11]\)
                                    if \(\Delta X_{8}[3]=\Delta X_{8}[7]=\Delta X_{8}[11]=\alpha^{\prime} \oplus 0 \times 01\) then
                            Inserted \(\left(P, P^{\prime}\right)\) into table \(\mathbb{T}_{2}\)
                                    end
                                    end
                                    Count the number of pairs \(\operatorname{Ctr}\) in \(\mathbb{T}_{2}\)
                                    if \(\mathrm{Ctr} \geqslant \tau\) then
                                    Derive candidates for \(R K_{0}[4,5,8,10]\) with \(\left(P, P^{\prime}\right)\) in \(\mathbb{T}_{2}\)
                                    Output \(R K_{0}[4,5,8,10]\left\|R K_{9}[3,4,7,11]\right\| \alpha^{\prime}\|\zeta\| \Delta[8-10]\)
                            end
                end
            end
            end
        end
    end
```

If $\Delta X_{8}[4]=0 x a d$, the quadruple $\left(P, P^{\prime}, O[3,7,11], O^{\prime}[3,7,11]\right)$ is inserted into table $\mathbb{T}_{1}$. The approximate number of quadruples in $\mathbb{T}_{1}$ is $N \cdot 2^{-64} \cdot 2^{-8}=N \cdot 2^{-72}$. This phase, which corresponds to line 14 of Algorithm 1, has a time complexity of $\mathrm{T}_{\mathrm{L} 14}=2^{21.98} \cdot 2^{8} \cdot N \cdot 2^{-64} \cdot 2 / 12=\mathcal{S} \cdot 2^{42.40}$ one-round encryptions.

Since the difference $\Delta R K_{9}[3,7,11]$ is related to undetermined values $\alpha^{\prime}$ and $\zeta$, the following attack should enumerate the values of $\alpha^{\prime}$ and $\zeta$. Noting that $5 \mathrm{a} \rightarrow \zeta$ is a possible propagation for the S-box, $\zeta$ can take on one of 127 possible values. Since ad $\rightarrow \alpha^{\prime} \oplus 0 \mathrm{x} 01$ and $\alpha^{\prime} \rightarrow \Delta O[10]$ must be possible propagations for the S-box, the probability that a random 8 -bit vector validates the two constraints for the case of $\alpha^{\prime}$ is $(127 / 256)^{2}=2^{-2.02}$. Therefore, $\alpha^{\prime}$ has an average of 63 possible values. Then, for all 63 possible values of $\alpha^{\prime}$ and 127 possible values of $\zeta$, we estimate the value of $R K_{9}[3,7,11]$ and create an empty table $\mathbb{T}_{2}$. After deriving the values of $R K_{9}^{\prime}[3]=R K_{9}[3] \oplus \alpha^{\prime}, R K_{9}^{\prime}[7]=R K_{9}[7] \oplus \alpha^{\prime} \oplus \zeta$, and $R K_{9}^{\prime}[11]=R K_{9}[11] \oplus \zeta$, it is possible to calculate the value of $\Delta X_{8}[3,7,11]$. If $\Delta X_{8}[3]=\Delta X_{8}[7]=\Delta X_{8}[11]=\alpha^{\prime} \oplus 0 \mathrm{x} 01$, the quadruple in $\mathbb{T}_{1}$ will be inserted into $\mathbb{T}_{2}$. Consequently, $\mathbb{T}_{2}$ contains approximately $N \cdot 2^{-72} \cdot 2^{-24}=N \cdot 2^{-96}$ quadruples. This step, which corresponds to line 24 of Algorithm 1, has a time complexity of $\mathrm{T}_{\mathrm{L} 24}=2^{21.98} \cdot 2^{8} \cdot 63 \cdot 127 \cdot 2^{24} \cdot N \cdot 2^{-72} \cdot 3 \cdot 2 / 12=\mathcal{S} \cdot 2^{72.95}$ one-round encryptions.

We set a counter Cnt in order to remember the number of quadruples in $\mathbb{T}_{2}$. Based on the analysis presented above, the value of Cnt follows a binomial distribution with parameters $\mathcal{B}\left(N, p_{0}=2^{-90}\right)$ for a correct key guess and $\mathcal{B}\left(N, p=2^{-96}\right)$ otherwise. The threshold $\tau$ is set to two correct pairs, and the success probability $P_{S}$ is set to $90.00 \%$. Using Eqn. (2), we determine $\mathcal{S}=2^{12.96}$ and $\varepsilon_{1}=2^{-14.77}$. Therefore, there are $\varepsilon_{1} \cdot 2^{21.98} \cdot 2^{32} \cdot 63 \cdot 127=\varepsilon_{1} \cdot 2^{66.95}$ candidates for $R K_{9}[3,4,7,11]\left\|\alpha^{\prime}\right\| \zeta \| \Delta[8-10]$ that satisfy the condition at line 30 of Algorithm 1. Utilising the property of the four active S-boxes in the first round, as depicted in Fig. 6(b), and relying on right pairs, additional information about the key can be recovered. Take the S-box at $X_{0}[4]$ as an illustration. Since the input difference $0 x 9$ a must be propagated to the output difference $0 x d b$, there are only four possible values for $X_{0}[4]$ and $X_{0}^{\prime}[4]$, which are $0 x 00,0 x 72,0 x 9 a$, and 0 xe8. This restriction allows us to screen out candidates for $R K_{0}[4]$ with a probability of $2^{-6}$. Likewise, the constraints on $X_{0}[5], X_{0}[8]$ and $X_{0}[10]$ yield a sieving probability of $2^{-18}$. There are a total of $\varepsilon_{1} \cdot 2^{66.95} \cdot 4^{4}=\varepsilon_{1} \cdot 2^{74.95}$ candidates for $R K_{0}[4,5,8,10]\left\|R K_{9}[3,4,7,11]\right\| \alpha^{\prime}\|\zeta\| \Delta[8-10]$. This phase, corresponding to line 31 of Algorithm 1, has a maximal time complexity of $T_{L 31}=\varepsilon_{1} \cdot 2^{66.95}$ one-round encryptions. We recover equivalently $1 / \varepsilon_{1}+6 \cdot 4=38.77$ bits of information about the key pair. As a result, the total time complexity of the attack is $\mathrm{T}_{\mathrm{L} 3}+\left(\mathrm{T}_{\mathrm{L} 14}+\mathrm{T}_{\mathrm{L} 24}+\mathrm{T}_{\mathrm{L} 31}\right) / 9=2^{92.96} 9$-round encryptions. Given that the hash table $\mathbb{H}$ dominates memory consumption, the memory complexity of the attack is $2^{56.96}$ bytes.

Remark 1. We attempt to recover the remaining key bits as well. However, the time required to seek the remaining key bits exhaustively exceeds $2^{94}$. The recovery of complete information about the key is an intriguing future endeavour.

## 6 Conclusion

This paper focuses on the differential distinguishers and related-key differential attacks on HALFLOOP-96. SAT problems are utilised to model the search for differential distinguishers. We use the SAT solver to determine the minimum number of active S-boxes and the maximum differential probability for the conventional, related-tweak, and related-key differential attack configurations. By applying the newly discovered 8 -round related-key differential distinguisher, we launch a 9 -round related-key differential attack against the cipher. The attack is weak-key and effective against $2^{94}$ key pairs with a specified difference. Although the attack does not pose a real security threat to HALFLOOP-96, the security margin of the cipher in the setting for related-key attacks is minimal. Consequently, care must be taken to avoid misuse.

Acknowledgements The research leading to these results has received funding from the National Natural Science Foundation of China (Grant No. 62272273, Grant No. 62002201, Grant No. 62032014), the National Key Research and Development Program of China (Grant No. 2018YFA0704702), and the Major Basic Research Project of Natural Science Foundation of Shandong Province, China (Grant No. ZR202010220025). Ling Sun gratefully acknowledges the support by the Program of TaiShan Scholars Special Fund for young scholars (Grant No. tsqn202306043).

## References

1. Interoperability and performance standards for medium and high frequency radio systems. United States Department of Defense Interface Standard MIL-STD-188141D
2. Specification for the advanced encryption standard (aes). Federal Information Processing Standards Publication 197 (2001), http://csrc.nist.gov/publications/ fips/fips197/fips-197.pdf
3. Abdelkhalek, A., Sasaki, Y., Todo, Y., Tolba, M., Youssef, A.M.: MILP modeling for (large) s-boxes to optimize probability of differential characteristics. IACR Trans. Symmetric Cryptol. 2017(4), 99-129 (2017). https://doi.org/10.13154/ tosc.v2017.i4.99-129
4. Ankele, R., Kölbl, S.: Mind the gap - A closer look at the security of block ciphers against differential cryptanalysis. In: Cid, C., Jr., M.J.J. (eds.) Selected Areas in Cryptography - SAC 2018-25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected Papers. Lecture Notes in Computer Science, vol. 11349, pp. 163-190. Springer (2018). https://doi.org/10. 1007/978-3-030-10970-7_8
5. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings. Lecture Notes in Computer Science, vol. 537, pp. 2-21. Springer (1990). https://doi.org/10.1007/3-540-38424-3_1
6. Biryukov, A., Khovratovich, D.: Related-key cryptanalysis of the full AES-192 and AES-256. In: Matsui, M. (ed.) Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5912, pp. 1-18. Springer (2009). https://doi. org/10.1007/978-3-642-10366-7_1
7. Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and related-key attack on the full AES-256. In: Halevi, S. (ed.) Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 1620, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5677, pp. 231-249. Springer (2009). https://doi.org/10.1007/978-3-642-03356-8_14
8. Blondeau, C., Gérard, B., Tillich, J.: Accurate estimates of the data complexity and success probability for various cryptanalyses. Des. Codes Cryptogr. 59(1-3), 3-34 (2011). https://doi.org/10.1007/s10623-010-9452-2
9. Boura, C., Coggia, D.: Efficient MILP modelings for sboxes and linear layers of SPN ciphers. IACR Trans. Symmetric Cryptol. 2020(3), 327-361 (2020). https: //doi.org/10.13154/tosc.v2020.i3.327-361
10. Brayton, R.K., Hachtel, G.D., McMullen, C.T., Sangiovanni-Vincentelli, A.L.: Logic Minimization Algorithms for VLSI Synthesis, The Kluwer International Series in Engineering and Computer Science, vol. 2. Springer (1984). https: //doi.org/10.1007/978-1-4613-2821-6
11. Cook, S.A.: The complexity of theorem-proving procedures. In: Harrison, M.A., Banerji, R.B., Ullman, J.D. (eds.) Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA. pp. 151-158. ACM (1971). https://doi.org/10.1145/800157. 805047
12. Dansarie, M., Derbez, P., Leander, G., Stennes, L.: Breaking HALFLOOP-24. IACR Trans. Symmetric Cryptol. 2022(3), 217-238 (2022). https://doi.org/10. 46586/tosc.v2022.i3.217-238
13. Kim, S., Hong, D., Sung, J., Hong, S.: Accelerating the best trail search on aeslike ciphers. IACR Trans. Symmetric Cryptol. 2022(2), 201-252 (2022). https: //doi.org/10.46586/tosc.v2022.i2.201-252
14. Lai, X., Massey, J.L., Murphy, S.: Markov ciphers and differential cryptanalysis. In: Davies, D.W. (ed.) Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings. Lecture Notes in Computer Science, vol. 547, pp. 17-38. Springer (1991). https://doi.org/10.1007/3-540-46416-6_2
15. Li, T., Sun, Y.: Superball: A new approach for MILP modelings of boolean functions. IACR Trans. Symmetric Cryptol. 2022(3), 341-367 (2022). https: //doi.org/10.46586/tosc.v2022.i3.341-367
16. Liu, Y., Liang, H., Li, M., Huang, L., Hu, K., Yang, C., Wang, M.: STP models of optimal differential and linear trail for s-box based ciphers. Sci. China Inf. Sci. 64(5) (2021). https://doi.org/10.1007/s11432-018-9772-0
17. Liu, Y., Wang, Q., Rijmen, V.: Automatic search of linear trails in ARX with applications to SPECK and chaskey. In: Manulis, M., Sadeghi, A., Schneider, S.A. (eds.) Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings. Lecture Notes in Computer Science, vol. 9696, pp. 485-499. Springer (2016). https: //doi.org/10.1007/978-3-319-39555-5_26
18. Matsui, M.: On correlation between the order of s-boxes and the strength of DES. In: Santis, A.D. (ed.) Advances in Cryptology - EUROCRYPT '94, Workshop on
the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 912, 1994, Proceedings. Lecture Notes in Computer Science, vol. 950, pp. 366-375. Springer (1994). https://doi.org/10.1007/BFb0053451
19. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Wu, C., Yung, M., Lin, D. (eds.) Information Security and Cryptology - 7th International Conference, Inscrypt 2011, Beijing, China, November 30 - December 3, 2011. Revised Selected Papers. Lecture Notes in Computer Science, vol. 7537, pp. 57-76. Springer (2011). https://doi. org/10.1007/978-3-642-34704-7_5
20. Sinz, C.: Towards an optimal CNF encoding of boolean cardinality constraints. In: van Beek, P. (ed.) Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3709, pp. 827-831. Springer (2005). https://doi.org/10.1007/11564751_73
21. Soos, M., Nohl, K., Castelluccia, C.: Extending SAT solvers to cryptographic problems. In: Kullmann, O. (ed.) Theory and Applications of Satisfiability Testing SAT 2009, 12 th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5584, pp. 244-257. Springer (2009). https://doi.org/10.1007/978-3-642-02777-2_24
22. Sun, B., Liu, Z., Rijmen, V., Li, R., Cheng, L., Wang, Q., AlKhzaimi, H., Li, C.: Links among impossible differential, integral and zero correlation linear cryptanalysis. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology - CRYPTO 2015-35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 1620, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 95-115. Springer (2015). https://doi.org/10.1007/978-3-662-47989-6_5
23. Sun, L., Wang, M.: Sok: Modeling for large s-boxes oriented to differential probabilities and linear correlations. IACR Trans. Symmetric Cryptol. 2023(1), 111-151 (2023). https://doi.org/10.46586/tosc.v2023.i1.111-151
24. Sun, L., Wang, W., Wang, M.: More accurate differential properties of LED64 and midori64. IACR Trans. Symmetric Cryptol. 2018(3), 93-123 (2018). https: //doi.org/10.13154/tosc.v2018.i3.93-123

# Not optimal but efficient: a distinguisher based on the Kruskal-Wallis test 

Yan Yan ${ }^{1}$, Elisabeth Oswald ${ }^{1}$, and Arnab Roy ${ }^{1,2}$<br>${ }^{1}$ University of Klagenfurt<br>${ }^{2}$ University of Innsbruck<br>yanyansmajesty@outlook.com; elizabeth.oswald@aau.at; arnab.roy@uibk.ac.at


#### Abstract

Research about the theoretical properties of side channel distinguishers revealed the rules by which to maximise the probability of first order success ("optimal distinguishers") under different assumptions about the leakage model and noise distribution. Simultaneously, research into bounding first order success (as a function of the number of observations) has revealed universal bounds, which suggest that (even optimal) distinguishers are not able to reach theoretically possible success rates. Is this gap a proof artefact (aka the bounds are not tight) or does a distinguisher exist that is more trace efficient than the "optimal" one? We show that in the context of an unknown (and not linear) leakage model there is indeed a distinguisher that outperforms the "optimal" distinguisher in terms of trace efficiency: it is based on the Kruskal-Wallis test.


Keywords: Distinguisher • Side Channel

## 1 Introduction

To exploit the information contained in side channels we use distinguishers: these are key-guess dependent functions, which are applied to the side channel observations and some auxiliary input (plaintext or ciphertext information), that attribute scores to key guesses. Optimal distinguishers [HRG14] are distinguishing rules derived by the process of maximising the likelihood of ranking the key guess that corresponds to the true secret value first (via their respective scores). The mathematical setup to derive optimal distinguishers is agnostic to estimation and trace efficiency, and thus an optimal distinguisher is not per construction the most trace efficient one. However, the optimal distinguishing rules that were derived in [HRG14] outperformed (experimentally) other distinguishers, or when not, [HRG14] showed mathematical equivalence between an optimal distinguishing rule and a classical rule. For instance, the correlation distinguisher turned out to be equivalent to the optimal rule in the situation where the leakage function is known and the noise is Gaussian.

The situation in which an adversary is confronted with a new device that contains an unknown key is interesting because it corresponds to the "hardest challenge" for the adversary: they should recover the key with only information about the cryptographic implementation. Framing this in the context of
side channel distinguishers, this leads to a type of distinguisher that neither requires assumptions about the noise distribution nor information about the device leakage distribution. Previous research has looked at distinguishers such as mutual information [GBTP08], Spearman's rank correlation [BGL08], and the Kolmogorov-Smirnov (KS) test [WOM11] in this context - these papers pre-date the seminal paper [HRG14] that establishes how to derive an optimal distinguishing rule.

Relatively recently only it was argued that the mutual information can be recovered as the optimal distinguishing rule [dCGHR18] if no assumptions about the device leakage distribution can be made. They also show experimentally that mutual information is the most trace efficient distinguisher in this setting. Next, better bounds for the estimation of the first order success rate (i.e. the probability to rank the key guess that corresponds to the true secret key first based on distinguishing scores) were derived in [dCGRP19]. The idea here was to derive these bounds independently of any specific distinguisher, purely based on the mutual information between the observed leakage and the key. The bounds were then compared to the respective optimal distinguishing rule. It turned out that there is a considerable gap between the optimal distinguisher and the bounds. This begs the question: could there indeed be a distinguisher that is more trace efficient than the one recovered as the optimal distinguishing rule?

### 1.1 Our contributions

We find a more trace efficient distinguisher by switching to rank based statistics. Previous work has once touched on rank based statistics before (Spearman's rank correlation) but we seek out a method that works even if the relationship between the intermediate values and the device leakage is not monotonic: this leads us to explore the Kruskal-Wallis method. We show how to translate it to the side channel context (the important trick here is to rank the traces itself prior to any partitioning) and we demonstrate how to estimate the number of needed traces for statistical attack success. We extend the existing work here by developing a lower bound for the number of needed traces.

Following established practice we then provide experimental results that enable us to conclude also from a practical point of view that the anticipated theoretical advantages show in practice. We cover a range of situations where we explore different target functions and different device leakage functions. In terms of target functions, we use non-injective target functions (as required by the assumptions in [HRG14,dCGHR18]), and also injective target functions with the bit-dropping trick. For device leakage functions we cover functions that range from highly non-linear to linear. We investigate Gaussian and Laplacian noise. Our philosophy is to include settings from prevoius work and more. We also consider implementations based on shared out intermediate values. Experiments that vary all these factors are necessarily based on simulations. We also demonstrate that our observations translate to real device data by using traces from two AES implementations: one with and one without masking.

Our research exhibits, for the first time, in the setting where no information about the device leakage distribution is available, a distinguishing rule that is more trace efficient than the optimal distinguishing rule (MI). Our research also shows for the first time that a purely rank based distinguisher is effective in the context of masking.

We provide the necessary background about (rank based) distinguishers, and our notation in Sect 2. Then we introduce the Kruskal-Wallis method and turn it into a distinguisher (alongside the analysis for the number of needed traces from a statistical point of view) in Sect. 3. In Sect. 4 we show and discuss the simulation results, and in Sect. 5 we show and discuss the results for the real traces. We conclude in Sect. 6.

## 2 Background

We try and use notation that is uncluttered whenever we refer to well established background, in particular, when it comes to known facts about distinguishers, and we "overload" variables so that they simultaneously refer to sets and random variables. For instance, we use $L$ to refer to the set of observed traces, which we also know to have a distribution.

### 2.1 Side channel attacks and notation

We assume that the side-channel leakage $L$ can be expressed as a sum of a key dependent function $M$ and some independent noise $\varepsilon$ :

$$
L=M\left(V_{k^{*}}\right)+\varepsilon .
$$

The device leakage model $M$ is not known in practice. It is a function of $V$, an intermediate value, which depends on some input word $X$ and a fixed and unknown secret key word $k^{*}$. We assume that the noise follows a Gaussian distribution $\varepsilon \sim \mathcal{N}(0, \sigma) .{ }^{3}$ The intermediate $V$ is derived by the keyed cryptographic function $f_{k^{*}}$ :

$$
V_{k^{*}}=f_{k^{*}}(X)
$$

In a side-channel attack, the adversary is given a set of leakages $L$ and their corresponding inputs $X^{4}$. To recover the correct (secret) key $k^{*}$ embedded within the device, the adversary first computes the (predicted) intermediates $V_{k}$ under all possible guesses of $k$, from the given input $X$. Then they compute the hypothetical leakage value $L_{\mathcal{H}, k}=\mathcal{H}\left(V_{k}\right)$ by assuming a leakage function $\mathcal{H}$. In side-channel attacks that rely on a direct or proportional approximation of the device leakage, the quality of $\mathcal{H}$ determines the success or efficiency of the corresponding attacks. When no model is known, then $\mathcal{H}$ is simply the identity function.

[^30]A distinguisher $D$ is used to compute the distinguishing score $d_{k}$ from the predicted intermediates $V_{k}$ and the observed leakage $L$. In a successful sidechannel attack, the correct key $k^{*}$ is determined as the maximum distinguishing score(s):

$$
k^{*}=\arg \max _{k} d_{k}=\arg \max _{k} D\left(L_{\mathcal{H}, k}, L\right)
$$

It is important to bear in mind that distinguishers are based on estimators of statistical quantities, thus in the formulas below we indicate this fact by placing a hat above the respective quantity. Distinguishers may or may not be based on some either assumed or known properties of the observed leakage $L$. In statistical jargon, statistics that require assumptions about the distribution are called "parametric" and statistics that do not require assumptions about the distribution are called "non-parametric". In this paper we work on the assumption that we are in a "first contact" scenario where the adversary utilises no information about $L$ in their initial attack attempt: this hence requires them to use non-parametric statistics, thus a non-parametric distinguisher.

In all practical side-channel attacks, the targeted intermediate $V_{k}$ is normally a part of operands being processed by the device during the cryptographic algorithms, and the key $k$ is a chunk of the cryptographic key. The complete key recovery is done via performing multiple side-channel attacks on each of the key chunks (thus we use a divide and conquer strategy). Also observable leakage often is given as a real-valued vector: e.g. power traces consist of many measurement points. Distinguishers are either applied to individual trace points, or to specific subsets of trace points. Therefore, in our aim to keep the notation uncluttered, we do not include any variables for indices for trace points or the like. We implicitly understand that the distinguisher is applied to (many) trace points or sets of trace points individually.

### 2.2 Rank transformations

Many statistical techniques that do not require assumptions about the underlying distributions have been developed by working on ranked data. Suppose that we have a set of leakages $L$ : there are several ways in which ranks can be assigned to the leakages in the set. The two most natural types of assigning ranks are the following:

Type 1: The entire set is ranked from smallest to largest (or vice versa), and the smallest leakage having rank 1 , the second smallest having rank 2 , etc.
Type 2: The set $L$ is partitioned according to some rule into subsets, then each subset is ranked independently of all other subset, by ordering the elements within a set (either from smallest to largest or vice versa).

Ties are resolved by assigning the average of the ranks that the ties would have received.

Any monotonic increasing function that is applied to the data does not change the ranking of the data. In our text we indicate that ranking takes place by applying the $\operatorname{rank}()$ function to the resp. variables. The type of ranking will be clear from the context.

### 2.3 Non-parametric side-channel distinguishers

For the sake of completeness we provide a very brief description of the nonparametric side-channel distinguishers that we use as comparisons with are new distinguisher.

Difference of Means The Difference of Means (DoM) [KJJ99] is often used as a baseline distinguisher, and it can be defined such that it makes minimal assumptions about the leakage distribution. For its' computation, the traces are divided into two groups $L_{V_{k}=0}$ and $L_{V_{k}=1}$ depending on whether a predicted single bit of a targeted intermediate is zero or one ( $V_{k}=0$ or $V_{k}=1$ ). The distinguishing score is defined as the estimated difference of means (often one takes the absolute value)):

$$
d_{k}=\left|\hat{\mathbb{E}}\left(L_{V_{k}=0}\right)-\hat{\mathbb{E}}\left(L_{V_{k}=1}\right)\right| .
$$

Spearman's Rank Correlation This is a non-parametric alternative to Pearson's correlation, and it was investigated in [BGL08] against an AES implementation. It was shown to be significantly more efficient (in terms of success rate) compared to Pearson's correlation-based attack [BCO04] (a.k.a. CPA). In this attack, the adversary computes the hypothetical leakage from $V_{k}$ by computing $L_{\mathcal{H}, k}$ where $\mathcal{H}$ is guessed/assumed by the adversary. Then $L_{\mathcal{H}, k}$ and $L$ are ranked and the (absolute value of the) correlation coefficient is estimated as follows

$$
d_{k}=\left|\frac{\hat{\operatorname{Cov}}\left(\operatorname{rank}(L), \operatorname{rank}\left(L_{\mathcal{H}, k}\right)\right)}{\hat{\sigma}_{\operatorname{rank}(L)} \hat{\sigma}_{\operatorname{rank}\left(L_{\mathcal{H}, k}\right)}}\right| .
$$

Notice that although the adversay must "guess" a hypothetical leakage model, there is no requirement for the device leakage to follow a Gaussian distribution.

Mutual Information Mutual Information [GBTP08] analysis is a distinguishing method that can be used without the need for $\mathcal{H}$. The MI distinguishing score is computed by estimating the mutual information from a set of collected traces and the corresponding inputs or plaintexts:

$$
d_{k}=\hat{I}\left(L, V_{k}\right)=\hat{H}(L)-\hat{H}\left(L \mid V_{k}\right)
$$

where $\hat{H}$ and $\hat{I}$ denote the (estimated) Shannon's entropy and mutual information respectively. For estimating MI, different entropy estimation methods have been studied, but the most commonly applied and efficient method (over $\mathbb{R}$ ) is the so-called binning method that is used in the original proposal of MIA [GBTP08]. We also use this same estimation method in our experiments.

Note that MI requires that the target function $f_{k}$ is not a bijection as discussed in [WOS14,dCGHR18]. When MIA is applied to cryptographic target that is a bijection, then the bit dropping technique [RGV14] that simply chops off a selected number bits from the output, is used. Although it is not necessary to supply MI with a hypothetical leakage model $\mathcal{H}$ this is frequently done in the literature, in particular by selecting the Hamming weight as $\mathcal{H}$.

Kolmogorov-Smirnov (KS) The KS test-based distinguisher [WOM11] is suggested as an alternative to using MI. The distinguishing score (of a key) is defined as the average of KS distances between the leakage distribution of $L$ and leakage distributions of $L_{V_{k}}$ for each predicted intermediate $V_{k}$ i.e.

$$
d_{k}=\hat{\mathbb{E}}_{V_{k}}\left(\sup _{l}\left|F_{L}(l)-F_{L_{V_{k}}}(l)\right|\right)
$$

where $F_{L}(l)$ and $F_{L_{V_{k}}}(l)$ are the Cumulative Distribution Functions (CDFs) of $L$ and $L_{V_{k}}$ respectively. From a finite sample set $A$ the empirical CDF is computed by $F_{A}(x)=\frac{1}{n} \sum_{a \in A} I_{a \leq x}$ where $I$ is the indicator function and $|A|=n$.

## 3 The Kruskal-Wallis test as side-channel distinguisher

The Kruskal-Wallis test (KW) [KW52] is a non-parametric method for the analysis of variance (ANOVA): this means it does not require any distributional assumption about the leakage $L$. The KW test is based on the ranks of the observed data and it is often used to check whether (or not) multiple groups of samples are from the same distribution. In this section we explain how to construct a KW based distinguisher, and we discuss the salient properties of the resulting distinguisher.

### 3.1 The KW statistic as a distinguisher

In this section we describe how to compute the KW statistic in a side-channel setting, and we argue why it gives a sound side channel distinguisher. For a generic description of the KW statistic we refer the readers to appendix A.

Informally, the KW test statistic is derived by first ranking the observed data, and second by grouping the data according to the resp. (key dependent) intermediate values. Then the tests checks if the groups can be distinguished from another or not, by comparing the variances between the groups and within the groups.

More formally, let us assume that we have $N$ side channel leakages. We apply the type 1 rank transformation to the side channel leaks, and then work with the ranked data: $\operatorname{rank}(L)$. For each key guess $k$, the ranked data is grouped according to the respective intermediate $V_{k}$. Thus the set $R_{k}^{i}=\left\{\operatorname{rank}(L) \mid V_{k}=i\right\}$ contains the ranks of leakages where the intermediate $V_{k}$ equals $i$. Let $R_{k}^{i, j}$ refers to the $j$-th element in $R_{k}^{i}$. Suppose that we have $t$ groups and the size of group $R_{k}^{i}$ is $n^{i}$ and so $N=\sum_{i=1}^{t} n^{i}$.

Let us assume that the group $R_{k}^{i}$ has distribution $F^{i}$. The null hypothesis is that all the groups have the same distribution, and alternative hypotheses of KW test is that the groups can be distinguished:

$$
\begin{align*}
& H_{0}: F^{0}=F^{1}=\ldots=F^{t-1}  \tag{1}\\
& H_{a}: F^{i} \neq F^{j} \quad \text { for some } \quad i, j \text { s.t } \quad i \neq j
\end{align*}
$$

The average of the ranks in $R_{i}$ is given as:

$$
\bar{R}_{k}^{i}=1 / n_{i} \sum_{j=1}^{n^{i}} R_{k}^{i, j}
$$

and $\overline{R_{k}}=(N+1) / 2$ the average of all $R_{k}^{i, j}$. The KW test statistic is defined [KW52] as:

$$
\begin{equation*}
d_{k}=(N-1) \frac{\sum_{i=1}^{t} n_{i}\left(\overline{R_{k}^{i}}-\bar{R}_{k}\right)^{2}}{\sum_{i=1}^{t} \sum_{j=1}^{n_{i}}\left(R_{k}^{i, j}-\overline{R_{k}}\right)^{2}} \tag{2}
\end{equation*}
$$

If the elements in $R_{k}^{i}$ are all from the same distribution, then all $\bar{R}^{i}{ }_{k}$ are expected to be close to $\overline{R_{k}}$ and thus the statistic $d_{k}$ should be smaller, than when the elements in $R_{k}^{i}$ are from different distributions. Thus large values of the test statistic imply that we reject the null hypothesis of the KW test (i.e. we have enough data to conclude that there are meaningful groups). We can use this test statistic readily as a side channel distinguisher: the groups are given by the key dependent intermediate values $V_{k}$. Thus, for $k=k^{*}$ we have a meaningful grouping of the ranked leakages, and thus the test statistic is large. If $k \neq k^{*}$, then the ranked side channel leaks are randomly assigned to different groups, which will lead to a small test statistic. Consequently the value of $d_{k^{*}}>=d_{k}$ for $\forall k$, which implies that it is a sound side channel distinguisher.

### 3.2 Properties of the KW distinguisher

Side channel distinguisher are most useful if they can be applied in different settings, including higher order attacks. It is also beneficial to be able to derive sample size estimates. For some of the existing non-parametric, in particular in the case of MI, this is hard to achieved. We now explain what is possible for the KW distinguisher.

Application to higher order attack scenarios. In masked implementations, an intermediate value is represented as a tuple of shares. The leakage of a single share is uninformative, but a statistic that exploits the distribution of the entire tuple enables key recovery. The canonical way of applying distinguishers to masked implementations is via processing the observed leakage traces: a popular (processing) function is the multiplication of (mean-free) trace points [PRB09]. Such trace processing produces a new trace in which each point now is based on the joint leakage of multiple points (aka shares). Using the mean-free product to produce joint leakage is compatible with the Kruskal-Wallis distinguisher (if the mean-free product of two values is larger than the mean-free product of another two values then this property is preserved by ranking: it is a monotonically increasing function), and we show how well it performs in the experimental sections.

Computational cost. The KW test is often compared to the Wilcoxon-WhitneyMann test (MWW) [MW47] with respect to computation costs, which is another rank based non-parametric test. The major difference between the two is that MWW is applied to paired data against two values, whereas KW is applied to multiple groups. The latter thus naturally fits with the side channel setting where the intermediate values fall naturally in multiple (independent) groups. Applying MWW in the side channel setting increases the computational cost. For example, in case of $t$ groups we need to apply MWW in the worst case $\binom{t}{2}$ times. Thus the KW test is a natural choice over the MWW test. We found that the computational cost of KW is of the same order as other generic distinguishers (MI, KS).

Number of samples. For the KW statistic, the theoretical analysis [FZZ11, Theorem 1] shows how to estimate the sample size. The main result necessary for estimating the sample size in a KW test is that under the alternative hypothesis the KW statistic follow a non-central $\chi^{2}$ distribution. Let $\lambda_{i}=n_{i} / N \geq \lambda_{0}$ for all $i$ and a fixed $\lambda_{0}>0$. And let $\alpha$ be the confidence level and $\beta$ be the power of the test. Then the estimated sample size is given as

$$
\begin{equation*}
\tilde{N}=\frac{\tau_{\alpha, \beta}}{12 \sum_{i=1}^{t} \lambda_{i}\left(\sum_{s \neq i} \lambda_{s}\left(\hat{p}_{i s}-1 / 2\right)\right)^{2}} . \tag{3}
\end{equation*}
$$

For each pair $i, s$ s.t. $i \neq s$, the probability estimates $\hat{p}_{i s}$ can be computed from the given data sample of size $N$ as follows

$$
\hat{p}_{i s}=\frac{1}{N_{i} N_{k}} \sum_{j=1}^{N_{i}} \sum_{\ell=1}^{N_{s}}\left(\mathcal{I}\left(X_{s \ell}<X_{i j}\right)+\mathcal{I}\left(X_{s \ell}=X_{i j}\right) / 2\right)
$$

where $\mathcal{I}$ is the indicator function, and $i, s \in\{1,2, \ldots t\}$. Note that the second part of the above expression corresponds to the ties in ranking. In eq. (3) $\tau_{\alpha, \beta}$ is solution to $\mathbb{P}\left(\chi_{t-1}^{2}(\tau)>\chi_{t-1,1-\alpha}^{2}\right)=1-\beta$ for some fixed $\alpha, \beta$, and $\chi_{t-1,1-\alpha}^{2}$ is the $(1-\alpha)$ quantile of central $\chi^{2}$ distribution with $t-1$ degrees of freedom.

The estimation of sample size following equation eq. (3) is biased and needs to be adjusted. As explained in [FZZ11], an adjusted estimator $\widehat{N}$ is defined as follows

$$
\begin{equation*}
\widehat{N}=\widetilde{N} \cdot \frac{\operatorname{median}\left\{\chi_{t-1}^{2}(\hat{\tau})\right\}}{\hat{\tau}} \tag{4}
\end{equation*}
$$

where $\hat{\tau}=N \cdot 12 \sum_{i=1}^{t} \lambda_{i}\left(\sum_{s \neq i} \lambda_{s}\left(\hat{p}_{i s}-1 / 2\right)\right)^{2}$.
Considering correct and incorrect key hypotheses. The application of sample size estimation technique requires care in the context of side-channel key recovery attack. Recall that in a statistical (hypothesis) testing there are two types of errors namely

1. Type I error $\alpha$ where the null hypothesis $H_{0}$ is rejected when the hypothesis $H_{0}$ is true, and
2. Type II error $\beta$ where the null hypothesis $H_{0}$ is not rejected when the alternate hypothesis $H_{a}$ is true.

In a successful attack the null hypothesis should not be rejected for any $k$ where $k \neq k^{*}$ (thus we want $\alpha$ to be small). However, under the correct key guess $k=k^{*}$ the alternative hypothesis $H_{a}$ is true and we should not fail to reject $H_{0}$. Hence, $\beta$ should be small so that the power of the test $1-\beta$ is large. In fact we wish to have a high power for both cases.

Thus we should perform the sample size estimation for both cases (correct and incorrect keys) and then take the maximum of these sample sizes as a conservative estimate. In statistical hypothesis testing typically it is ensured that the value of $\mathbb{P}$ (Type I error $) \leq 0.1$ and $\mathbb{P}$ (Type II error $) \leq 0.2$.

Example 1. In this example we show the sample size estimation for $N=1000$ using simulated Hamming weight traces of AES Sbox where the Gaussian noise has $\sigma=6$.

We choose $\alpha=0.025$ (corresponding to the confidence level) and $\beta=0.05$ (corresponding to the power of the test). First, using the technique as described above, we find the generic estimate of the sample size as per eq. (3). For applying the leakage estimation (or KW attack) we extract the 4 Least Significant Bits (LSB) from the output of the Sbox.

For this experiment the degrees of freedom of the $\chi^{2}$ distributions is $16-1=$ 15 (the number of different groups are 16 corresponding to the 4 -bit output values obtained). Note that $\tau_{\alpha, \beta}$ depends only on the degrees of freedom, $\alpha$ and $\beta$. In this case $\tau_{\alpha, \beta}=1.8506$. We compute $\widetilde{N}$ for different key choices. Here we only show the computation for one key that corresponds to the maximum $\widetilde{N}$. The estimation process is carried out in the same way for other keys.

Estimating $\lambda_{i}$ and $\hat{p}_{i s}$ from 1000 data points we obtain the $\widetilde{N}=\frac{1.8506}{.0041} \approx 451$. Since this is a biased estimate we obtain the adjusted estimate as

$$
\begin{equation*}
\widehat{N}=\widetilde{N} \cdot \frac{\operatorname{median}\left\{\chi_{t-1}^{2}(\hat{\tau})\right\}}{\hat{\tau}}=451 \cdot \frac{\operatorname{median}\left\{\chi_{15}^{2}(4.1)\right\}}{4.1} \approx 2015 \tag{5}
\end{equation*}
$$

Remark 1. For estimating the sample size in the context of side-channel attack, $\lambda_{i}$ can be estimated from the target cryptographic function (instead of estimating it from the data). Suppose, the target function is 8 -bit Sbox, and say 4 bits of the output is chosen for the attack. In this case, for all $2^{8}$ input values, the number of elements $n_{i}$ in each $2^{4}$ groups can be computed.

Example 2. In this example we show the sample size estimation when traces are simulated from ARX function with a HW leakage model and Gaussian noise with $\sigma=6$. We fix $N=1000$ and follow the same process as in Example 1. Here we choose $\alpha=0.001$ and $\beta=0.1$.

We consider a key recovery attack (using KW statistic) which recovers 4-bit key chunk from each $k_{1}$ and $k_{2}$, in the usual divide and conquer process used for side-channel attack. The ARX function is defined as

$$
A(x)=\left(x \oplus k_{1}\right) \boxplus\left(y \oplus k_{2}\right) .
$$

$\left(\oplus\right.$ denotes the bit-wise exclusive-or and $\boxplus$ the addition in $G F\left(2^{16}\right)$ ). So, the degrees of freedom for the $\chi^{2}$ distribution remains $16-1=15$. The biased sample size estimation gives $\widetilde{N} \approx 992$. After adjusting the bias as in Example 1 we get $\widehat{N} \approx 2212$.

Corollary 1. The generic estimate $\tilde{N}$ in eq. (3) (and bias adjusted estimate $\widehat{N}$ in eq. (4)) gives estimated lower bound on sample size.

Proof. The sample estimate is derived from the fact that $\hat{\tau} \approx \tau_{\alpha, \beta}$. Recall that $\tau_{\alpha, \beta}$ is the solution to the equation

$$
\mathbb{P}\left(\chi_{t-1}^{2}(\tau)>\chi_{t-1,1-\alpha}^{2}\right)=1-\beta
$$

for some fixed $\beta$. Now, if we obtain a $\hat{\tau}_{1}$ from the fixed sized data such that $\hat{\tau}_{1} \geq \tau_{\alpha, \beta}$, then $\mathbb{P}\left(\chi_{t-1}^{2}\left(\tau_{1}\right)>\chi_{t-1,1-\alpha}^{2}\right)$ will be more than $1-\beta$. This is favourable since we want to maximise the power of the test. Thus we have

$$
\hat{\tau} \geq \tau_{\alpha, \beta} \quad \Longrightarrow \quad \tilde{N}^{*} \geq \frac{\tau_{\alpha, \beta}}{12 \sum_{i=1}^{t} \lambda_{i}\left(\sum_{s \neq i} \lambda_{s}\left(\hat{p}_{i s}-1 / 2\right)\right)^{2}}=\tilde{N}
$$

The lower bound on the bias adjusted estimate $\widehat{N}^{*}$ follows from this.

## 4 Experiments based on simulated leakage

We now detail a range of experiments that are based on simulating side channel data. Experiments based on simulated data offer the advantage, over experiments based on data from devices, that we can efficiently vary implementation characteristics such as the leakage function, the cryptographic target function, and the signal to noise ratio. Therefore the inclusion of simulations is standard in research on distinguishers.

We display simulation outcomes in terms of the success rate as function of an increasing number of side channel observations. Our comparisons include the KW test, mutual information analysis (MI) with an identity leakage model, mutual information analysis with a Hamming weight leakage model (MI-HW), the Kolmogorov-Smirnov test and Spearman's test. We included MI-HW because of its wide use in the literature (and despite the obvious fact that it is no longer assumption free).

Before we discuss the outcomes, we provide an informal but detailed description of the choices for the cryptographic target functions $V_{k}$ as well as the device leakage functions $M$.

### 4.1 Simulation setup

Our choice of target functions $V_{k}$ is informed by best-practice: it is well known that properties of the target function impact on distinguishability and therefore we aimed to select a function that is known to be "poor" target, to challenge all distinguishers. Our selection observed a further requirement imposed by the use of MI (as main comparison) that MI is only a sound distinguisher for non-injective target functions (if a target function is injective, then MI cannot distinguish any key candidates)[HRG14], and the bit-dropping trick must be used. Therefore, we selected as a poor non-injective target function $V_{n i}$ the non-injective target function is the modular addition that is part of many ARX constructions, which is also the basis of modern permutation based ciphers such as SPARKLE:

$$
V_{n i}\left(x_{l}, x_{r}, k_{l}, k_{r}\right)=\left(x_{l} \oplus k_{l}\right) \boxplus\left(x_{r} \oplus k_{r}\right)
$$

where $x_{l} \| x_{r} \in\{0,1\}^{32}$ is a state element, and $k_{l} \| k_{r} \in\{0,1\}^{32}$ is the key, and $\boxplus$ is the addition modulo $2^{16}$.

We also experimented with a function that is known to be an excellent target function, namely the AES SubBytes operation, which is injective, and thus the bit-dropping trick must be applied. To aid the flow of this submission, we include the results of this in the appendix (they are aligned with the results for the injective target function).

Our choice of leakage functions $M$ is also informed by best-practice: leakage functions are also well known to impact on distinguisher performance. Linear leakage functions help distinguishers that are based on distributional assumptions or simple hypothetical leakage models. Highly non-linear leakage functions are representative of of complex leakage originating in combinational logic ([LBS19] and [GMPO20]) are a motivating factor for studing "assumption free" distinguishers like MI, KS and KW.

In our experiments we thus use a range of device leakage functions, which are defined as follows. Let $y_{i}$ be the $i$ th bit of $y$. Then we consider two linear device leakage functions (Hamming weight and Randomly weighted bits), and two non-linear leakage functions (Strongly non-linear and Binary), as follows:

Hamming weight: $M(y)=\sum_{i=1}^{n} y_{i}$
Randomly weighted bits: $M(y)=\sum_{i=1}^{n} w_{i} y_{i}$ with $w \in[-1,1]$
Strongly non-linear: $M(y)=S(y)$, with $S(y)$ defined to be the Present S-Box Binary: $M(y)=\sum_{i} S(y)_{i}(\bmod 2)$, with $S(y)$ defined to be the Present S-Box

### 4.2 First order attack simulations

Figure 1a shows that the Spearman rank correlation has indeed a significant advantage (because it uses the correct hypothetical leakage model), compared to the other distinguishers. Note that the KW test-based attack outperforms the other generic distinguishers with a clear margin that is more significant in the lower SNRs.

(b) Rrandomly weighted bits leakage model

SNR $=\mathbf{2}^{\mathbf{0}}$


SNR $=\mathbf{2}^{-5}$


(c) Strongly non-linear (PRESENT S-Box) leakage model


SNR $=2^{-3}$
SNR $=2^{-5}$


(d) Binary leakage model

KW:- MI:- MI-HW:+ KS:× Spearman:-
Fig. 1: Simulations for Modular Addition as a Target
 KW:— MI:- MI-HW:+ KS:× Spearman:-

Fig. 2: 2-share Boolean masking of ARX with different leakage models

Figure 1b shows that the Spearman rank correlation fails: more traces reduce the success rate, which is a clear indication that that the "built in leakage model" is incompatible with $M$. This is a useful reminder that linear models are not necessarily compatible with a Hamming weight assumption. All modelfree distinguishers succeed, and KW turns out to be the most trace efficient in all SNR settings. The MI and MI-HW distinguishers show similar performance while KS is the least trace efficient one among the successful attacks.

In the non-linear simulation (Figure 1c) We expect that Spearmans rank correlation will fail because the leakage model is not compatible. However MI with the same model works very well, alongside MI without model and KS. These three distinguishers show a very similar performance in all SNR settings. KW shows a clear margin to the other distinguishers, which is evidence that it is the preferable distinguisher in this setting.

The last simulation (Figure 1d) is a binary leakage model that represents an extreme case where the leakage is either 0 or 1 such that only a minimum resolution exists in the leakage values. In a high SNR setting, all assumption-free distinguishers recover the key. In low SNR seetings, the KW distinguisher show the quickest convergence to a high success rate, which is evidence that it is the preferable distinguisher in this setting.

### 4.3 Masked implementation

We further extend our simulations to a masked implementation by simulating the leakages of a 2 -shares Boolean masking scheme using the same leakage models as before. To perform an attack we use the a well understood, and frequently adopted approach of combining the leakages from all independent shares via the centred product-combining function, [PRB09], which was also used in $\left[\mathrm{BGP}^{+} 11\right]$. 5

[^31]
KW:- MI:- MI-HW:+ KS:×

Fig. 3: Higher order Boolean masking of AES with Hamming weight leakage

The results of the simulations for the 2-share Boolean masking scheme are shown in Figure 2a, Figure 2b and Figure 2c. For succinctness, we excluded the very low SNR settings of $2^{-3}$ and $2^{-5}$ (because the observations are the same as for the higher SNRs), and the results of binary leakage model (because all distinguishers failed in this setting). As is evident from the graphs, Spearman fails in all settings; among the successful attacks, KW turned out to be the most trace efficient distinguisher.

We then turn our attention to masking for the AES SubBytes operation, where Figures 3a-3c show that KW provides a clear advantage for low order masking.

## 5 Experiments based on Device Data

To complement our simulation results we also show experiments that were performed based on measurements from two processors. These processors are based on the ARM Cortex M0 and the ARM Cortex M3 architecture. We implemented the same target functions as before in the simulations.

To work with the masked implementation, we perform the same mean-free product combining pre-processing as in the simulations. Before showing the outcomes, we discuss the implementation characteristics in some more detail.

### 5.1 Implementation characteristics and experimental setup

Our simulated experiments ranged from unprotected implementations to implementations based on sharing out intermediate values. For implementations that are unprotected we only ensure functional correctness of our implementation. In the case of the non-injective target function, we utilise the modular addition
outcomes are highly sensitive to various factors, including leakage models, noise levels and methods for pre-processing, etc.
in C and let the compiler translate this into Assbembly code. In the case of the AES SubBytes implementation we use a simple table-based lookup. For the masked SubBytes implementation we use a custom Thumb-16 Assembly implementation of a two share ISW multiplication gadget. This implementation is specifically crafted to ensure that there are no first-order leaks.

Both processors are mounted in a special purpose measurement rig ${ }^{6}$. We have a state of the art scope and probe, but do not perform any filtering or de-noising before applying the distinguishers. The devices that we use are well characterised, and we know that they exhibit a range of leakage functions, which all have a strong linear component (thus they resemeble the two linear leakage functions that we considered in the simulations).

We apply the distinguishers to all trace points, and perform repeat experiments to determine the first order success rate. We then select the best point and plot the success rate graphs for this point only.

### 5.2 Experimental results

Non-injective target function. Figure 4a shows the results of repeat attacks on the modular addition on the M0. In the corresponding simulated experiments, we supplied Spearman with the Hamming weight leakage model and as a result it outperformed the other distinguishers when the device leakage model was also the Hamming weight. To demonstrate that Spearmans succeess in the Hamming weight simulation really was because we supplied it with the Hamming weight model, we now supply it with only 4 bits of the intermediate values. We give the same 4 bit intermediate values also to MI, MI-HW, KS and KW.

Lacking the correct leakage model, Spearman now completely fails. All other side-channel distinguishers successfully recover the key. KW shows again a better success rate than the competitors.

Injective target function. Figure 4b shows the results of repeat attacks on the SubBytes operation on the M0. Now we supply Spearman once more with the Hamming weight leakage model, which gives it a significant advantage over the other distinguishers (because the device features signifant linear leakage in all trace points).

KW is the most trace efficient distinguisher among the other distinguishers. DoM is the least efficient one which might due to the fact that DoM can only exploit a single bit leakage whereas other distinguishers exploit all 4 bit leakages.

Masked implementation. Figure $4 \mathrm{c}^{7}$ shows a familiar picture: KW achieves a higher success rate by a clear margin over the other distinguishers. Spearman failed, so we did not include it anymore. The picture also shows that MI-HW

[^32]

Fig. 4: Experiments based on real device data
no longer shows any advantage over MI, which one should expect given that pre-processing is applied to the trace points as part of attacking the masking scheme.

## 6 Discussion and Conclusion

Of the distinguishers that we compared in this submission, Spearman and MIHW are supplied with the Hamming weight leakage model. Theoretically, this gives them an advantage in situations where there is strong Hamming weight device leakage. We can see this advantage also experimentally: in all Hamming weight simulations, Spearman outperforms all other distinguishers, including MI-HW. This particular simulation showcases that iff the device leakage model is "simple" then there is no point in using MI, KS or KW.

In situations where the leakage model is unknown and HW based attack fail, they are the premise of our work, MI, KS, and KW are considerably better than Spearman (and MI-HW). When looking carefully at the experimental outcomes, then we can observe that the gap between the distinguishers decreases with lower SNR values. This behaviour is expected because of [MOS11], according to which they must, asymptotically speaking, get closer in terms of trace efficiency the lower the SNR.

All together our experiments provide strong evidence that MI is not the most trace efficient distinguisher setting where no leakage model is available, which is in contrast to [dCGHR18], who selected different distinguishers for comparison with MI.

Our results help clarify that "optimal distinguishers" are not necessarily the most trace efficient distinguishers, despite that in previous work they have always been identified as being more trace efficient (in their respective categories) than their "normal" counterparts.

## 7 Acknowledgment

Elisabeth Oswald and Yan Yan have been supported in part by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 725042).

## References

BCO04. Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings, volume 3156 of Lecture Notes in Computer Science, pages 16-29. Springer, 2004.
BGL08. Lejla Batina, Benedikt Gierlichs, and Kerstin Lemke-Rust. Comparative evaluation of rank correlation based DPA on an AES prototype chip. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, Information Security, 11th International Conference, ISC 2008, Taipei, Taiwan, September 15-18, 2008. Proceedings, volume 5222 of Lecture Notes in Computer Science, pages 341-354. Springer, 2008.
BGP $^{+}$11. Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Mutual information analysis: a comprehensive study. J. Cryptol., 24(2):269-291, 2011.
dCGHR18. Eloi de Chérisey, Sylvain Guilley, Annelie Heuser, and Olivier Rioul. On the optimality and practicability of mutual information analysis in some scenarios. Cryptogr. Commun., 10(1):101-121, 2018.
dCGRP19. Eloi de Chérisey, Sylvain Guilley, Olivier Rioul, and Pablo Piantanida. Best information is most successful mutual information and success rate in side-channel analysis. IACR Trans. Cryptogr. Hardw. Embed. Syst., 2019(2):49-79, 2019.
FZZ11. Chunpeng Fan, Donghui Zhang, and Cun-Hui Zhang. On sample size of the kruskal-wallis test with application to a mouse peritoneal cavity study. Biometrics, 67 1:213-24, 2011.
GBTP08. Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In Elisabeth Oswald and Pankaj Rohatgi, editors, Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings, volume 5154 of Lecture Notes in Computer Science, pages 426-442. Springer, 2008.
GMPO20. Si Gao, Ben Marshall, Dan Page, and Elisabeth Oswald. Share-slicing: Friend or foe? IACR Trans. Cryptogr. Hardw. Embed. Syst., 2020(1):152174, 2020.
HRG14. Annelie Heuser, Olivier Rioul, and Sylvain Guilley. Good Is Not Good Enough - Deriving Optimal Distinguishers from Communication Theory. In Lejla Batina and Matthew Robshaw, editors, Cryptographic Hardware and Embedded Systems - CHES 2014-16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings, volume 8731 of Lecture Notes in Computer Science, pages 55-74. Springer, 2014.
KJJ99. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings, volume 1666 of Lecture Notes in Computer Science, pages 388-397. Springer, 1999.
KW52. William H. Kruskal and W. Allen Wallis. Use of ranks in one-criterion variance analysis. Journal of the American Statistical Association, 47(260):583-621, 1952.

LBS19. Itamar Levi, Davide Bellizia, and François-Xavier Standaert. Reducing a masked implementation's effective security order with setup manipulations and an explanation based on externally-amplified couplings. IACR Trans. Cryptogr. Hardw. Embed. Syst., 2019(2):293-317, 2019.
MOS11. Stefan Mangard, Elisabeth Oswald, and François-Xavier Standaert. One for all - all for one: unifying standard differential power analysis attacks. IET Inf. Secur., 5(2):100-110, 2011.
MW47. H. B. Mann and D. R. Whitney. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. The Annals of Mathematical Statistics, 18(1):50-60, 1947.
PRB09. Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical analysis of second order differential power analysis. IEEE Trans. Computers, 58(6):799-811, 2009.
RGV14. Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. Generic DPA attacks: Curse or blessing? In Emmanuel Prouff, editor, Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers, volume 8622 of Lecture Notes in Computer Science, pages 98-111. Springer, 2014.
WOM11. Carolyn Whitnall, Elisabeth Oswald, and Luke Mather. An exploration of the kolmogorov-smirnov test as a competitor to mutual information analysis. In Emmanuel Prouff, editor, Smart Card Research and Advanced Applications - 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers, volume 7079 of Lecture Notes in Computer Science, pages 234-251. Springer, 2011.
WOS14. Carolyn Whitnall, Elisabeth Oswald, and François-Xavier Standaert. The myth of generic dpa...and the magic of learning. In Josh Benaloh, editor, Topics in Cryptology - CT-RSA 2014 - The Cryptographer's Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings, volume 8366 of Lecture Notes in Computer Science, pages 183-205. Springer, 2014.

## A The KW Statistic

Let $X_{i j}$ where $i=1, \ldots, t, j=1, \ldots, n_{i}$ be independent random samples collected from a population having $t$ groups and the sample size for group $i$ is $n_{i}$. Let us assume that the random variables $X_{i j}$ have distribution $F_{i}$. The generic null and alternative hypotheses of KW test are

$$
\begin{align*}
& H_{0}: F_{1}=F_{2}=\ldots=F_{t}  \tag{6}\\
& H_{a}: F_{i} \neq F_{j} \quad \text { for some } \quad i, j \quad \text { s.t } \quad i \neq j
\end{align*}
$$

The observations are combined into one sample of size $N$ where

$$
N=\sum_{i=1}^{t} n_{i}
$$

This combined sample is ranked. Suppose, $R_{i, j}$ is the ranking of the $j$-th sample from the group $i, \bar{R}_{i}$ the average rank of all samples from group $i$ :

$$
\bar{R}_{i}=n_{i}^{-1} \sum_{j=1}^{n_{i}} R_{i, j}
$$

and $\bar{R}=(N+1) / 2$ the average of all $R_{i, j}$.
The KW test statistic $H_{K W}$ is defined [KW52] as:

$$
\begin{equation*}
H_{K W}=(N-1) \frac{\sum_{i=1}^{t} n_{i}\left(\bar{R}_{i}-\bar{R}\right)^{2}}{\sum_{i=1}^{t} \sum_{j=1}^{n_{i}}\left(R_{i, j}-\bar{R}\right)^{2}} \tag{7}
\end{equation*}
$$

In eq. (7) the denominator $\sum_{i=1}^{t} n_{i}\left(\bar{R}_{i}-\bar{R}\right)^{2}$ describes the variation of ranks between groups, and the numerator $\sum_{i=1}^{t} \sum_{j=1}^{n_{i}}\left(R_{i, j}-\bar{R}\right)^{2}$ describes the variation of ranks in the combined sample. Intuitively, if $X_{i j}$ are all sampled from the same distribution, then all $\bar{R}_{i}$ are expected to be close to $\bar{R}$ and thus the statistics $H_{K W}$ should be smaller, and vice versa. Large values of the test statistic results in rejecting the null hypothesis of the KW test.

## B Further experimental results



KW:- MI:-_ MI-HW:+ KS:× Spearman:-_ DoM: o
Fig. 5: Attacking the AES SubBytes target, dropping 4 most significant bits

# Feasibility Analysis and Performance Optimization of the Conflict Test Algorithms for Searching Eviction Sets 

Zhenzhen $\mathrm{Li}^{1,2}$, Zihan Xue ${ }^{1,2}$, and Wei Song ${ }^{1,2}$<br>${ }^{1}$ Institute of Information Engineering Chinese Academy of Sciences, Beijing, China<br>${ }^{2}$ School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China \{lizhenzhen1, xuezihan, songwei\}@iie.ac.cn


#### Abstract

Cache side-channel attacks have been widely utilized as an intermediate step in some comprehensive attacks. Eviction sets, especially the minimal eviction sets, are essential components of the conflictbased cache side-channel attacks. It is important to develop efficient search algorithms that incur the lowest latency with the highest success rate. Several fast search algorithms have been proposed in recent years, among which conflict test (CT) achieves the highest success rate with the lowest latency. In this paper, we have conducted the first systematic feasibility analysis of the CT algorithm. Besides failing on the commonly known cache architectures where the last-level cache (LLC) is exclusive or non-inclusive, CT is also found and verified failing on two inclusive LLC architectures if it is running in single-core mode. We have further explored three optimizations for improving the speed performance of the CT algorithm, two of which are newly proposed in this paper.


Keywords: Computer micro-architecture • Cache architecture • Cache side-channel attack • Eviction set construction

## 1 Introduction

As an effective way of obtaining sensitive information from the cache system [10, $18,24]$, cache side-channel attacks have been widely utilized as an intermediate step in some comprehensive attacks, such as reconstructing cryptographic keys $[1,6,11,29,30,37]$, disarming the address space randomization $[7,8]$ in control-flow attacks, retrieving the leaked information at the end of a transient execution attack $[14,15]$, and constantly striking a row of the off-chip memory in a rowhammer attack [9].

Eviction sets, especially the minimal eviction sets [32], are essential components of the conflict-based cache side-channel attacks [34]. In such attacks, an attacker and her victim share the same cache space, typically certain cache sets in the last-level cache (LLC). The attacker needs to control the state of these shared cache sets to monitor the memory accesses of her victim, which are then used to infer security-critical information. To be specific, the attacker occupies (primes) a cache set by accessing an eviction set [9]; therefore, her victim's access
to this cache set must incur a cache miss, refilling of the missing cache block, evicting an address from the eviction set, and eventually a prolonged access. Both the address eviction and the prolonged access latency might be observable and used to infer the access of her victim.

All addresses in a minimal eviction set are congruent with (mapping to) the targeted cache set [32]. At least $W$ addresses are required for a $W$-way set-associative cache. Obviously, the key for constructing an eviction set is to find enough congruent addresses. Unfortunately, this is not an easy task on modern processors. LLC is indexed by physical addresses but attackers control only virtual addresses. A complex addressing scheme is utilized by modern Intel processors [17] to randomize the mapping from physical addresses to LLC slices. Attackers are usually forced to search eviction sets at runtime from a large amount of random addresses. It is important to develop efficient search algorithms that incur the lowest latency with the highest success rate. Several fast search algorithms have been proposed in recent years, including group elimination (GE) [16, 27, 32], prime, prune and probe (PPP) [19, 22], conflict test (CT) [23] and write-after-write (W+W) [28]. Among these algorithms, CT achieves the highest success rate with the lowest latency (see Table 1), and becomes one of the most widely utilized search algorithms [20,21]. However, there lacks a systematic analysis on the feasibility and the potential optimization of CT while similar analyses have been done for GE [27] and PPP [19].

In this paper, we have conducted the first systematic feasibility analysis of the CT algorithm. Besides sharing a commonly known limitation with other algorithms, that CT fails to work on exclusive or non-inclusive LLCs, CT also fails on two inclusive LLC architectures if the algorithm is running in single-core mode. Based on the result of the feasibility analysis, we have further explored three techniques for further optimizing the CT algorithm, two of which are newly proposed in this paper. Overall, this paper makes the following contributions:

- Conduct a systematic feasibility analysis on CT. For the first time, two inclusive cache architectures are identified as infeasible for single-core CT.
- Optimize the performance of CT by improving the efficiency of the cacheback technique and propose two new techniques.
- Practically evaluate the optimization techniques on both real processors and a behavioral-level cache model.


## 2 Background

Modern processors are multicore processors adopting a two/three-level cache structure. Each processing core contains a pair of private level-one (L1) instruction and data caches. Some processors, especially the Intel ones, equip each core with a uniformed level-two (L2) cache. A large LLC (L2 or L3) is shared among all cores. This LLC might be divided into multiple slices, whose mapping with physical addresses is decided by an undisclosed hash function (complex addressing scheme [17]) in Intel processors. All levels of caches are set-associative
writeback allocated caches. According to [31], all cache levels in the early generations (Haswell and earlier) and the L1 caches in recent Intel processors utilize the pseudo-LRU (PLRU) replacement policy [5], while L2 and LLC in recent Intel processors adopt some policies derived from RRIP [13]. The situation is similar for most other commercial processors, such as AMD ones. In some rare cases, random replacement policy is used in embedded-level processors [26]. In all cache architectures, LLC acts as the coherence hub. LLC and the private L1/L2 caches maintain either an inclusive relation (Intel's consumer processors), where all cache blocks in the private caches are also stored in the LLC, or a noninclusive relation (Intel's Xeon and AMD's Ryzen), where cache blocks stored in private caches may not be concurrently stored in the LLC.

Cache side-channel attacks normally fall in two categories: flush-based and conflict-based attacks. Flush-based attacks use explicit flush instructions (clflush on x 86 [36]) to invalidate a targeted data out of the cache architecture. These attacks are accurate but require the targeted data is accessible by the attacker, which is a rather strict requirement infeasible in most cross-process side-channel attacks. As an alternative, conflict-based attacks can achieve the similar effect. They evict the targeted data out of the LLC by occupying the corresponding LLC cache set with a collection of attacker's controlled cache blocks, typically called an eviction set. An eviction set is a collection of addresses (cache blocks) that contain enough addresses congruent with the targeted data. A sufficiently large number of addresses are also an eviction set as they can evict any cache block by priming the whole caches [33]. However, this type of untargeted eviction introduces undesirable noise [9] and brings down the attack speed [7]. What is really desirable is a minimal eviction containing only the congruent addresses. For simplicity, an "eviction set" beyond this point refers to a minimal eviction set. This paper concentrates on the algorithms for searching eviction sets.

Existing search algorithms for eviction sets can be classified into two categories: pruning algorithms which begin with an untargeted eviction set containing a large number of random addresses and prune it into a minimal one, and inserting algorithms which begin with an empty collection and gradually fill it with newly found congruent addresses until it becomes an eviction set.

GE and PPP are the two widely utilized pruning algorithms. GE prunes the initial large eviction set in a multi-round process. In each round, the remaining $N$ addresses are divided into $W+1$ groups. Since a minimal eviction set contains only $W$ addresses, at least one group contains none of the $W$ addresses and should be removed. By sequentially testing whether the address collection remains an eviction set without a certain group, the removable group is found and removed. The prune process continues until a minimal set is produced. GE is robust in tolerating environment noise, as indicated by the high success rate shown in Table 1, but the multi-round prune is slow.

PPP reduces the prune latency by manipulating the PLRU replacement policy $[22,23]$. It first tries to store addresses of the initial large eviction set into the LLC concurrently by gradually removing the addresses causing self-evictions. The resulted (reduced) eviction set is still untargeted but may fully occupy the

Table 1. Speed comparison of different search algorithms for eviction sets.

| CPU | GE |  | PPP |  | $\mathbf{W}+\mathbf{W}$ |  | CT |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | latency | rate | latency | rate | latency | rate | latency | rate |
| i7-3770 | $58 \pm 32 \mathrm{~ms}$ | 74\% | $0.69 \pm 1.7 \mathrm{~ms}$ | 8.8\% | $33 \pm 41 \mathrm{~ms}$ | 6.1\% | $6.0 \pm 3.4 \mathrm{~ms}$ | 69\% |
| i7-6700 | $82 \pm 66 \mathrm{~ms}$ | $79 \%$ | $1.0 \pm 2.9 \mathrm{~ms}$ | 0.9\% | $10 \pm 5.3 \mathrm{~ms}$ | 5.3\% | $23 \pm 21 \mathrm{~ms}$ | 16\% |
| i7-9700 | $115 \pm 92 \mathrm{~ms}$ | 85\% | $0.65 \pm 0.68 \mathrm{~ms}$ | 11\% | $159 \pm 8.5 \mathrm{~ms}$ | 2.0\% | $20 \pm 17 \mathrm{~ms}$ | 21\% |
| i7-11700 | $642 \pm 586 \mathrm{~ms}$ | 24\% | $0.81 \pm 0.04 \mathrm{~ms}$ | 7.0\% | $3 \pm 1.4 \mathrm{~ms}$ | 0.4\% | $12 \pm 4.4 \mathrm{~ms}$ | 2.1\% |

targeted cache set. Then the attacker incurs an eviction in the targeted cache set by accessing the targeted address, following with a timed re-access of the reduced eviction set. Due to the PLRU replacement policy, it is likely that exactly $W$ addresses (just enough for an eviction set) are found missing in the LLC. However, the probability that the reduced eviction set occupying the targeted cache set is actually low in a large LLC with many cache sets. As shown in Table 1, the success rate of PPP is much lower than GE.

CT is the mostly utilized inserting algorithm. It was initially proposed only for LLCs adopting the random replacement policy [23]. In this case, a congruent address has a $1 / W$ probability to evict the targeted cache block. As a random address is a congruent address by a probability of $1 / S$, where $S$ is the number of cache sets, one congruent address can be found by probing around $S W$ random addresses. Finding eviction set with $W$ congruent addresses therefore requires probing $\mathcal{O}\left(S W^{2}\right)$ random addresses. This algorithm is also effective for permutation-based replacement, such as LRU and RRIP. Instead of finding congruent addresses by detecting the eviction of the targeted cache block, detecting the prolonged write latency due to the LLC enforced serialization of parallel writes to the same cache set was also found effective [28]. The resulted algorithm, namely $W+W$, was claimed faster than the GE algorithm. However, the accuracy of such serialization detection is found extremely noisy and unstable, which results in low success rates as shown in Table 1.

To compare the speed performance of these algorithms, they are ported to four Intel processors and the result is shown in Table 1. CT seems to provide the most balanced performance in latency and success rate. The search latency is significantly lower than GE while the success rate is much higher than PPP (except for $\mathrm{i} 7-11700$ ). The search latency of $\mathrm{W}+\mathrm{W}$ is shorter than CT only on i7-6700 and i7-11700 but the success rate is much lower on both processors. This paper concentrates on improving the CT algorithm.

## 3 Feasibility Analysis

This section conducts a systematic analysis on the feasibility of the CT algorithm on different cache architectures. For the first time, the CT algorithm is found infeasible on two inclusive cache architectures.

### 3.1 Threat Model

For an eviction set search algorithm, we define a successful attack as finding an eviction set. We assume that the search algorithm is run by an attacker in a restricted user mode environment with the following capabilities and limitations:

- The targeted LLC is shared between the attacker and her victim.
- The amount of memory acquirable by the attacker is not limited by the system, so the attacker can access an arbitraily large range of addresses.
- The attacker either runs in the same core with her victim or occupies a separate core.
- The attacker can flush her own data out of the LLC.
- The attacker can accurately trick her victim into accessing a target address without incurring a large amount of noise.
- Some parameters regarding the cache system are made available, such as the replacement policy, the inclusiveness relation, and the number of sets and ways of each cache level, but neither the virtual to physical page mapping nor the Intel complex addressing scheme [17] is reverse-engineered.


### 3.2 Necessary Working Conditions

Algorithm 1 illustrates the baseline CT algorithm. Different with other papers $[20,23]$, we explicitly specify the cores running the victim and the attacker. When $C_{a}=C_{v}$, the attacker and her victim are running on the same core or even in the same process/thread. This is the typical case for cache side-channel attacks that tries to break the user-mode address randomization [8], leak information through transient execution [14, 15], and constantly hammer a targeted DRAM row [9]. We call this the single-core case while the traditional cross-core (process) attack as the cross-core case. As we will soon discover in Section 3.3, CT may fail to work on some inclusive cache architectures when running in the single-core case while remains feasible for cross-core.

According to Algorithm 1, a random address $a$ is found congruent with the targeted address $x$ only if accessing $a$ (line 6) causes a miss in the targeted cache

```
Algorithm 1: The baseline CT algorithm
    Input: \(x\), target address; \(W\), number of ways; \(\left(C_{a}, C_{v}\right)\), cores running the attacker and
        her victim.
    Output: \(\mathcal{E}\), an eviction set for \(x\).
    function \(\operatorname{ct}\left(x, W, C_{a}, C_{v}\right)\)
        \(\mathcal{E} \leftarrow \varnothing / /\) eviction set
        \(C_{v}: \operatorname{access}(x)\)
        while \(|\mathcal{E}|<W\) do
            \(a \leftarrow \operatorname{random}\) ()
            \(C_{a}\) : \(\operatorname{access}(a)\)
            if not \(C_{v}: \operatorname{probe}(x)\) then
                \(\mathcal{E} \cup\{a\}\)
            end
        end
        return \(\mathcal{E}\)
    end
```


(a) Before accessing $a_{7}$.

Fig. 1. Purging $x$ (cross-core case) after accessing $a_{7}$ by $C_{a}$ in a 2-level inclusive cache architecture. $\left(W_{\mathrm{L} 1}=4, W_{\mathrm{LLC}}=8\right.$, all caches use LRU $)$
set and the cache block containing $x$ is evicted for refilling $a$. In addition, the eviction of $x$ can be observed by probing $x$ (code highlighted in blue): a timed access of $x$. If the probe latency is longer than a pre-defined threshold, $x$ is assumed missing and $a$ is identified as congruent. Two necessary conditions for the success of CT can be derived from Algorithm 1:

Condition 1: Inclusion victim effect. When an LLC is the targeted cache, the targeted cache block stored in a private L1 cache (the potential inclusion victim [12]), such as the $x$ stored in $C_{v}: L 1$ depicted in Fig. 1a, must be purged from the cache architecture when its copy in the LLC is evicted due to a conflict, such as the access of $a_{7}$ by $C_{a}$ shown in Fig. 1b. In other words, CT works only when the targeted LLC is inclusive. Note that this condition is required for the single-core case as well, since $x$ is also purged by a conflict in the LLC.

Condition 2: Cache filter effect. When the CT algorithm is used to target an LLC adopting LRU/RRIP replacement policies, the probing of $x$ is observed by the LLC only after $x$ is successfully evicted in the LLC, such as probing $x$ after accessing $a_{7}$ as shown in Fig. 1b. The cache filter effect is a by-product of the hierarchical cache architecture where memory accesses hitting in private caches are invisible to the LLC. When the LLC adopts PLRU/RRIP replacement policies, the target address $x$ is possible to be evicted by a fresh access of a new random address $a$ only when $x$ is pushed to the LRU position, as shown in Fig. 1a, by a number of accesses (random addresses) to the cache set after the previous access of $x$ is observed by the LLC. According to Algorithm 1, $x$ is accessed once in the probe for each random address. All of these accesses must be filtered from the LLC (served by private L1/L2 caches); otherwise, $x$ is repeatedly accessed in the LLC and cannot be pushed to the LRU position. This is the first time that such condition has been discovered and we will show in the next section (Section 3.3) why CT fails on some inclusive cache architectures (satisfying condition 1) due to the lack of this cache filter effect.

### 3.3 Feasibility on Different Cache Architectures

Utilizing the two necessary conditions discovered in Section 3.2, we have conducted a systematic survey on the feasibility of CT on different cache architectures. We consider the following cache parameters:

- Cache levels: cache architectures that have two or three levels of caches.
- Inclusiveness: inclusive ( $L 1 \subseteq L L C$ ), exclusive ( $L 1 \neq L L C$ ) or noninclusive ( $L 1 \nsubseteq L L C$ ) relation between cache levels.

Table 2. Feasibility on different cache architectures.

| Architecture | Example | Attack | Feasible |
| :---: | :---: | :---: | :---: |
| Exclusive or Non-inclusive LLC ${ }^{a}$. | AMD Zen 2 and later (Ryzen-7 5700G) | cross-core <br> single-core | $\begin{aligned} & \text { No } \\ & \text { No } \end{aligned}$ |
| Inclusive LLC $^{b}$ with private caches using LRU/RRIP. | Intel Processors (i7-6700 and Xeon $4110^{b}$ ) | cross-core <br> single-core | Yes Yes |
| Three levels of inclusive caches using LRU/RRIP. | Early quad/hexa-core processors (Intel Dunnington [2, 25]) | cross-core <br> single-core | Yes <br> No |
| Inclusive LLC using LRU/RRIP with private caches using random. | A customized Rocket-Chip processor (Section 5.1) | cross-core <br> single-core | Yes <br> No |

${ }^{a}$ A non-inclusive LLC may adopt an inclusive directory and CT becomes feasible, such as the Intel
Xeon processors [35]. These cache architectures are counted as inclusive LLCs without differentiating the directory from the cache.
${ }^{b}$ Include the non-inclusive LLCs adopting inclusive directories.

(a) Before accessing $a_{7}$.

(b) After accessing $a_{7}$.

Fig. 2. A failing example of single-core CT in a 3-level inclusive cache using LRU. $\left(W_{\mathrm{L} 1}=4, W_{\mathrm{L} 2}=8, W_{\mathrm{LLC}}=16\right)$

- Cache sets and ways: when the LLC is inclusive, it is assumed that the number of ways in the LLC ( $W_{\text {LLC }}$ ) is no less than it in in private caches: $W_{\mathrm{LLC}} \geq W_{\mathrm{L} 2}$ if $L 1 \subseteq L 2$ or $W_{\mathrm{LLC}} \geq W_{\mathrm{L} 1}+W_{\mathrm{L} 2}$ otherwise.
- Replacement policy: the replacement policy of individual cache can be independently selected among LRU, RRIP or random.
- Attack scenario: both cross-core and single-core attacks are considered.

In total, we have surveyed 168 different cache architectures (scenarios) and identified four categories of representative cache architectures as revealed in Table 2:

Exclusive or Non-inclusive LLC: When the LLC is exclusive or noninclusive, the target $x$ stored in L1 cannot become an inclusion victim and CT fails to work. Nearly all recent AMD Zen 2 and later processors fall in this category and are naturally immune to the CT algorithm. Intel Xeon processors adopt a non-inclusive LLC but utilize an inclusive directory. Due the inclusiveness of the directory, they are still vulnerable to CT. We count them as inclusive LLCs.

Inclusive LLC with private caches using LRU/RRIP: This is the common category for nearly all Intel processors. The inclusive LLC ensures the inclusion victim effect. As for the cache fitler effect, since LRU/RRIP is adopted by the private caches, the repeatedly probing of $x$ ensures that $x$ is pinned in the L1 and all accesses to $x$ are invisible to LLC until $x$ is evicted from the LLC. Note that we deliberately leave an exception here for simplicity. As described by the next category, a three-level inclusive LLC using LRU/RRIP can break the condition for the cache filter effect.

Three levels of inclusive caches using LRU/RRIP: This is an exception of the previous category. CT works in the cross-core case but fails in the
single-core case due to the lack of the filter effect. A failing example is presented in Fig. 2. After accessing seven congruent addresses ( $a_{0}$ to $a_{6}$ ) and probing $x$, the state of the three-level cache is depicted in Fig. 2a. Note the repeated probing of $x$ is filtered by L1 and invisible to both L2 and LLC. As a result, $x$ is pushed to the LRU position in L2. As demonstrated in Fig. 2b, the following access of $a_{7}$ thus evicts $x$ from L2, which consequently purges $x$ also from L1 as it is an inclusion victim. As $x$ is purges from both L1 and L2, the probing of $x$ is observed by LLC, which moves $x$ to the MRU position in LLC and CT fails. The rooting cause is that the probing of $x$ is invisible to the inclusive L2 while $W_{\mathrm{L} 2}<W_{\mathrm{LLC}}$. Early generations of the Intel quad/hexa-core multiprocessors, such as the Intel Dunnington architecture [2, 25] adopts such a three-level inclusive cache architecture. The L2 cache in later generations becomes exclusive, which unfortunately makes them vulnerable to single-core CT.

Inclusive LLC using LRU/RRIP with private caches using random: This architecture is uncommon as most L1 caches adopt LRU/RRIP replacement policies. However, the single-core CT fails in such an architecture as $x$ is likely evicted from the private caches before it is evicted from the LLC due to the random replacement, which makes the following probing of $x$ observed by the LLC. CT therefore fails due to the lack of the filter effect. In Section 5.1, we have configured the cache architecture of a dual-core Rocket-Chip accordingly as a demonstrative example for the failing of single-core CT.

## 4 Performance Optimization

This section begins with a performance analysis of the baseline CT algorithm. Based on the analysis, three optimization techniques are proposed to improve the efficiency of the CT algorithm.

### 4.1 Performance Analysis of the Baseline Algorithm

Let us consider a cross-core attack on a two-level inclusive cache using the LRU replacement policy. The latency $(L)$ of searching one eviction set of $W$ congruent addresses can be estimated as:

$$
\begin{equation*}
L=\left(N_{\mathrm{RA}}+W\right) \cdot t_{\mathrm{mem}}+\left(N_{\mathrm{v}}-W\right) \cdot t_{\mathrm{L} 1}+N_{\mathrm{v}} \cdot \Delta_{\text {cross }} \tag{1}
\end{equation*}
$$

where $N_{\mathrm{RA}}$ and $N_{\mathrm{v}}$ are the numbers of accessing random addresses and the victim address $x$, respectively, while $t_{\mathrm{mem}}, t_{\mathrm{L} 1}$ and $\Delta_{\text {cross }}$ are the time for one memory access, the time for one access hitting in L1, and the time overhead for one cross-core access, respectively. The total number of LLC misses is $N_{\mathrm{RA}}+W$ and $N_{\mathrm{v}}-W$ times of probing $x$ should hit in L1 due to the perfect filter effect.

Due to the LRU replacement policy, the target address $x$ is evicted from the LLC every time when $W$ congruent random addresses are accessed. A total of $W^{2}$ congruent random addresses are searched before obtaining an eviction set.

We call this number $N_{\mathrm{CA}}$. Since random address is a congruent address with $x$ by a probability of $1 / S, N_{\mathrm{RA}}$ and $N_{\mathrm{v}}$ can be estimated as:

$$
\begin{equation*}
N_{\mathrm{RA}}=N_{\mathrm{v}}=N_{\mathrm{CA}} \cdot S=S W^{2} \tag{2}
\end{equation*}
$$

where $S$ is the number of LLC sets. Using Equation 1, $L$ is rewritten to:

$$
\begin{align*}
L & =\left(S W^{2}+W\right) \cdot t_{\mathrm{mem}}+\left(S W^{2}-W\right) \cdot t_{\mathrm{L} 1}+S W^{2} \cdot t_{\text {cross }}  \tag{3}\\
& =S W^{2} \cdot\left[t_{\mathrm{mem}}+\left(t_{\mathrm{L} 1}+\Delta_{\text {cross }}\right)\right]+W \cdot\left(t_{\mathrm{mem}}-t_{\mathrm{L} 1}\right)  \tag{4}\\
& =S \cdot N_{\mathrm{CA}} \cdot\left(t_{\mathrm{mem}}+t_{\mathrm{v}}\right)+W \cdot \Delta_{\mathrm{miss}} \tag{5}
\end{align*}
$$

where $t_{\mathrm{v}}$ and $\Delta_{\text {miss }}$ are the time for one (cross-core) probing of $x$ and the time overhead of one cache (both L1 and LLC) miss, respectively. According to Equation 5, the key for reducing $L$ is to decrease $N_{\mathrm{CA}}$, the number of congruent random addresses requiring to be accessed, as all others are constants.

Equation 5 holds true for cross-core attacks on all feasible cache architectures, even when the LLC adopts the random replacement policy. In this case, Equation 2 remains the same as a random address is a congruent address with $x$ by a probability of $1 / S$, accessing a congruent address evicts $x$ by a probability of $1 / W$, and $x$ is evicted for $W$ times during the whole search. Consequently, Equation 4 and 5 remain untouched

For single-core attacks, Equation 5 remains valid as long as the L1 adopts LRU/RRIP replacement policies because LRU/RRIP guarantees the perfect filter effect. $t_{\mathrm{v}}$ is reduced to $t_{\mathrm{L} 1}$ as the cross-core overhead is removed. When both L1 and LLC adopt the random replacement policy, accessing a random address evicts $x$ from the L1 cache by a probability of $1 /\left(S_{\mathrm{L} 1} \cdot W_{\mathrm{L} 1}\right)$. Therefore, extra latency is introduced in Equation 4 and 5:

$$
\begin{align*}
L & =S W^{2} \cdot\left(t_{\mathrm{mem}}+t_{\mathrm{L} 1}\right)+W \cdot\left(t_{\mathrm{mem}}-t_{\mathrm{L} 1}\right)+\frac{S W^{2}}{S_{\mathrm{L} 1} \cdot W_{\mathrm{L} 1}} \cdot\left(t_{\mathrm{LLC}}-t_{\mathrm{L} 1}\right)  \tag{6}\\
& =S \cdot N_{\mathrm{CA}} \cdot\left(t_{\mathrm{mem}}+t_{\mathrm{v}}\right)+W \cdot \Delta_{\mathrm{miss}}+\frac{S \cdot N_{\mathrm{CA}}}{S_{\mathrm{L} 1} \cdot W_{\mathrm{L} 1}} \cdot \Delta_{\mathrm{L} 1-\mathrm{miss}} \tag{7}
\end{align*}
$$

where $t_{\mathrm{v}}=t_{\mathrm{L} 1}$ and $\Delta_{\mathrm{L} 1-\text { miss }}=t_{\mathrm{LLC}}-t_{\mathrm{L} 1}$, which is the time overhead of accessing LLC when L1 misses. Similarly, the key for reducing $L$ is to decrease $N_{\mathrm{CA}}$ as all others are constants.

### 4.2 Cacheback: Reducing the Number of Random Accesses

Cacheback is an optimization capable of reducing $N_{\mathrm{CA}}$ when the LLC adopts an LRU/RRIP replacement policy. In the baseline CT algorithm, every time the target address $x$ is evicted from the LLC, a total of $W$ congruent addresses are accessed but only the last one is identified by the algorithm, because it finally evicts $x$. When a number of congruent addresses are identified and stored in $\mathcal{E}$ (line 8 in Algorithm 1), these addresses can be used to push $x$ to the LRU position and reduce the total number of congruent addresses ( $N_{\mathrm{CA}}$ ) needed in the

```
Algorithm 2: Cacheback after a successful probe
    if not \(C_{v}\) :probe (x) then
        \(\mathcal{E} \bigcup\{a\}\)
        for \(e\) in \(\mathcal{E}\) do
            \(C_{a}\) :access (e)
        end
    end
```



Fig. 3. Problem of cacheback when the order observed by LLC (L2) mismatching with the program order. (According to i7-6700, $W_{\mathrm{L} 1}=8, W_{\mathrm{L} 2}=4, W_{\mathrm{LLC}}=16, \mathrm{~L} 2$ is exclusive, L2 and LLC adopt RRIP)
search. This cacheback procedure is described in the code extraction of the probe illustrated in Algorithm 2 (replacing the code highlighted blue in Algorithm 1) with the optimization highlighted in red. After the $i$-th congruent address $\left(e_{i}\right)$ is identified by CT, the number of congruent addresses needed for identifying the next one is reduced to $W-i$. Therefore, $N_{\mathrm{CA}}$ is reduced to:

$$
\begin{equation*}
N_{\mathrm{CA}}=\sum_{i=0}^{W-1}(W-i)=\frac{W^{2}+W}{2} \tag{8}
\end{equation*}
$$

Compared with Equation 2, the total number of congruent addresses needed in the search is roughly reduced by half, so does the search latency.

This optimization is first proposed in the Prime+Scope attack [20]. By further investigation, we find out that the optimization works but not as effective as it should be. There are two reasons for this reduced efficiency: mismatching access order and broken filter effect. Let us consider an example of single-core attack on a three-level cache depicted in Fig. 3. The access order observed by the LLC might not match with the access order issued by the program. As a result, when the target address $x$ is evicted by accessing address $a_{3}$ in Fig. 3a, the access order observed (more importantly the replacement order) by L2 and LLC mismatches with the program order for address $e_{2},{ }^{3}$ assuming 12 congruent addresses ( $e_{0}$ to $e_{11}$ ) have been identified, stored in $\mathcal{E}$, and used for cacheback. $a_{3}$ is identified as a congruent address and stored in $\mathcal{E}$ after probing $x$ (refill $x$ to L1 and LLC as well). According to the cacheback optimization, $e_{0}$ to $e_{11}$, along with $a_{3}$ are accessed

[^33]```
Algorithm 3: Flush before cacheback
    if not \(C_{v}: \operatorname{probe}(x)\) then
        \(\mathcal{E} \bigcup\{a\}\)
        for \(e\) in \(\mathcal{E}\) do
            \(C_{a}:\) flush(e)
        end
        for \(e\) in \(\mathcal{E}\) do
            \(C_{a}: \operatorname{access}(e)\)
        end
    end
```

```
Algorithm 4: Interleavedly re-access target during cacheback
```

Algorithm 4: Interleavedly re-access target during cacheback
if not $C_{v}$ : probe ( $x$ ) then
if not $C_{v}$ : probe ( $x$ ) then
$\mathcal{E} \cup\{a\}$
$\mathcal{E} \cup\{a\}$
for $e$ in $\mathcal{E}$ do
for $e$ in $\mathcal{E}$ do
$C_{a}:$ flush(e)
$C_{a}:$ flush(e)
end
end
for $e$ in $\mathcal{E}$ do
for $e$ in $\mathcal{E}$ do
$C_{a}$ : access (e)
$C_{a}$ : access (e)
$C_{v}: \operatorname{access}(x) / /$ single-core, $C_{v}=C_{a}$
$C_{v}: \operatorname{access}(x) / /$ single-core, $C_{v}=C_{a}$
end
end
end

```
    end
```

to push $x$ towards the LRU position in LLC. When the cacheback proceeds to $e_{2}$, this address hits in L2 and is swapped to L1 rather than accessing from LLC due to the order mismatch. Consequently, the access of $e_{2}$ is invisible to LLC, reducing the effectiveness of the cacheback and the access order in L2 and LLC diverse further away from the program order. To avoid the mismatching access order, we propose to flush all the addresses stored in $\mathcal{E}$ before caching them back, as highlighted in the code extraction of the probe part illustrated in Algorithm 3 (replacing the code highlighted blue in Algorithm 1). In this way, each accessing of $e_{i}$ forces an insertion at the MRU position in the LLC.

The other problem is the broken filter effect in single-core CT attack when the number of addresses in $\mathcal{E}$ is larger than the associativity of the inner caches: $|\mathcal{E}| \geq W_{\mathrm{L} 1}+W_{\mathrm{L} 2}$ for the cache architecture shown in Fig. 3. Let us consider the situation after the cacheback process is finished, the target $x$ is actually evicted from L1 and L2, because the total number of addresses in $\mathcal{E}$ becomes 13 after adding $a_{3}$. As a result, LLC observes a re-access of $x$ soon after probing for the next random address. This would put $x$ to the unfavorable MRU position if LLC adopts the LRU replacement. It is even worse for the RRIP replacement policy as a re-access of $x$ promotes it to higher replacement priority [13], which would fail the CT algorithm. To avoid such problem, we propose to interleavedly re-access the target address $x$ during the cacheback process, as shown in Algorithm 4 (replacing the code highlighted blue in Algorithm 1). In this way, CT ensures that $x$ is never evicted from L1.

### 4.3 Extended Probing: Increasing the Probability of Probing

Cacheback is effective only when the LLC adopts an LRU/RRIP replacement policy. If the policy is random, the probability of evicting the target address $x$ is independent for every random address being tested. Cacheback is therefore

```
Algorithm 5: Extended probing
    \(r=\) TRUE
    \(r=r\) and \(C_{v}: \operatorname{probe}(x)\)
    for \(e\) in \(\mathcal{E}\) do
        \(r=r\) and \(C_{a}: \operatorname{probe}(e)\)
    end
    if not \(r\) then
        \(\mathcal{E} \cup\{a\}\)
    end
```

useless. In this situation, we propose to directly improve the probability of identifying a congruent address in the probing. Instead of probing only the target address $x$, an attacker can additionally probe all the found congruent addresses stored in $\mathcal{E}$, as they all stored in the same LLC cache set. Algorithm 5 demonstrates the probe (code highlighted blue in Algorithm 1) optimized with the extend probing. If any of the target address $x$ or the addresses stored in $\mathcal{E}$ is probed missing in the LLC, $r$ becomes FALSE, and the random address $a$ is then identified as congruent and added to $\mathcal{E}$.

Assuming the size of $\mathcal{E}$ is $|\mathcal{E}|$, the probability of identifying a congruent address increases from $\frac{1}{W}$ to $\frac{1+|\mathcal{E}|}{W}$, which approaches to $64 \%$ when $|\mathcal{E}|=15$ for a 16 -way LLC. Consequently, $N_{\mathrm{CA}}$ is reduced from 256 to 54.1 , achieving a $79 \%$ reduction. However, the search latency does not drop proportionally to the reduction of $N_{\mathrm{CA}}$. In fact, the latency benefit eventually drops to negative with the increasing of $|\mathcal{E}|$, because the total number of accesses issued by probes rises proportionally to $|\mathcal{E}|$. They incur a significant latency overhead when $|\mathcal{E}| \rightarrow W$. When $|\mathcal{E}|<W_{\text {L1 }}$, addresses in $\mathcal{E}$ likely hit in L1. The extra accesses introduced by the extended probing are served by L1 and the latency overhead is small. When $|\mathcal{E}| \geq W_{\mathrm{L} 1}$, the extended probing begins to experience significant amount of L1 misses. The latency overhead would gradually becomes intolerable. There should be an optimal number of addresses applied with the extended probing.

### 4.4 Surrogate Targets: Reducing Victim Accesses

The final optimization is related to reduce the number of probing the target address $x$. In certain attack scenarios, tricking the victim to probe the target address $x$ (normally cross-core) is a time consuming and noisy procedure ( $\Delta_{\text {cross }} \gg t_{\mathrm{L} 1}$ in Equation 4), especially when the victim is non-cooperative or the victim probe is likely bulky (containing unrelated code). As a result, the total time required for constructing an eviction might not be decided by the complexity of the search algorithm but largely by the number of victim accesses [22].

According to Equation 2, the number of victim access $N_{\mathrm{v}}=N_{\mathrm{CA}} \cdot S$, which is a fairly large number. We would like to significantly reduce $N_{\mathrm{v}}$. Instead of probing the target address $x$, an attacker can replace $x$ with a found congruent address as the surrogate target, such as $e_{0}$ stored in $\mathcal{E}$. The number of total victim access $N_{\mathrm{v}}$ is therefore reduced to the number of victim accesses required for identifying the first congruent address, which is only $S \cdot W$. Note that this effectively convert an originally cross-core attack into a single-core one. Therefore, it is viable only

Table 3. Cache misses incurred by testing 1000 random addresses.

| Scenario | $C_{0} \mathbf{L 1}$ miss | $C_{0} \mathbf{L 2}$ miss | $C_{1} \mathbf{L} 1 \mathbf{m i s s}$ | $C_{1} \mathbf{L 2}$ miss | LLC miss |
| :--- | :---: | :---: | :---: | :---: | :---: |
| cross-core | $1000 \pm 0.0$ | $1000 \pm 0.0$ | $1.3 \pm 0.5$ | $1.3 \pm 0.5$ | $1001 \pm 0.5$ |
| single-core | $1016 \pm 1.3$ | $1016 \pm 1.3$ | $0 \pm 0$ | $0 \pm 0$ | $1000 \pm 0.03$ |

for the cache architectures feasible for the single-core case. It is also worthwhile to point out, this technique is universally effective for all inserting algorithms.

## 5 Performance Evaluation

The performance of CT with various optimizations is evaluated by running them on actual processors whenever possible. The two assumed failing cache architectures for the single-core case (Section 3.3) are first verified. Consequently, the speed benefits of the optimizations proposed in Section 4 are measured.

### 5.1 Feasibility Verification

It is widely understand that CT fails on exclusive/non-inclusive cache architectures. This section concentrates on verifying of the two inclusive cache architectures identified in Table 2 (Section 3.3) where the single-core CT fails.

For the three levels of inclusive caches using LRU/RRIP, we verify the failing single-core case using a behavioral cache model [27] as we do not have any of the early Intel machines or any open processor implementation adopting a three-level cache architecture. The cache model is configured with two cores. $\left(C_{a}, C_{v}\right)=\left(C_{0}, C 1\right)$ for the cross-core case. Each core contains a 64 -set 8 -way L1 and an private 512 -set 8 -way L2, while a 4096 -set 12 -way LLC is shared between cores. All caches adopt the LRU replacement policy.

The baseline CT is used to test 1000 random addresses for both cross-core and single-core cases. Complying with the analysis provided in Table 2, 1~2 congruent addresses are found in the cross-core case but none in the single-core case. Table 3 reveals the cache misses recorded in all caches. In the cross-core case, testing 1000 random addresses incurs $\sim 1001$ misses in the LLC, where the extra $1 \sim 2$ misses are caused by the eviction of the target addresses $x$ from the LLC, which is confirmed by the matching missing number on core $C_{1}\left(C_{v}\right)$. In the single-core case, the number of cache misses incurred by testing 1000 random addresses is $\sim 1016$ on L1 but exactly 1000 on LLC. The 16 extra misses on L1 is caused by the eviction of the target address $x$ from the L2, which would lead to re-accessing $x$ on the LLC (broken filter effect). As a result, $x$ is never pushed to the LRU position in the LLC, and CT fails.

For the inclusive LLC using LRU/RRIP with private caches using random, we manage to configure a dual-core Rocket-Chip [3,4] with a two-level cache architecture where the 1024 -set 16 -way LLC is inclusive using LRU while the 64 -set 8 -way L1 uses random. The Rocket-Chip is ported to a FPGA dev board, runs at 50 MHz , and boots with a Linux kernel (ver. 5.11.0).


Fig. 4. The latency distribution of probing the target address $x$.


Fig. 5. Search latency and success rate on Intel i7-6700 when various cacheback optimizations are applied.

The baseline CT algorithm runs on this dual-core Rocket-Chip for both crosscore and single-core cases. The cross-core case successfully finds eviction sets with a probability of $13 \%$ while the single-core case fails, complying with Table 2. To verify this result, the latency distribution of probing the target address $x$ has been collected and depicted in Fig. 4. For the cross-core case, $99.5 \%$ probes hit in L1 ( $\sim 4$ cycles), while $\sim 0.4 \%$ probes miss in LLC ( $>45$ cycles). The tested CT algorithm uses random addresses sharing the same page offset with the target address $x$, providing a theoretical conflicting rate of $1 / 256(0.391 \%)$. The $0.4 \%$ LLC miss rate matches perfectly with the theory. For the single-core case, only $87.4 \%$ probes hit in L1, $12.4 \%$ probes hit in LLC ( $\sim 25$ cycles), and none misses in the LLC. Due to the random replacement policy used in L1, the target address $x$ shall be evicted from the L1 by a probability of $1 / 8(12.5 \%)$, in theory. This matches with the $12.4 \%$ probes hitting in the LLC. Due to this effect, $x$ is never pushed to the LRU position in LLC, and CT fails.

### 5.2 Speed Optimization Results

Cacheback (Section 4.2) reduces $N_{\mathrm{AC}}$ along with the search latency in inclusive LLCs adopting LRU/RRIP replacement policies. We use Intel i7-6700 as our default processor for analyzing the different techniques for improving the efficiency of cacheback while the final performance of CT with the optimized cacheback is compared on all the four Intel processors.

Fig. 5 demonstrates the search latency and success rate on Inel i7-6700 when different optimization techniques are applied to the cacheback process. In the cross-core case, applying the basic cacheback alone without flushing before cacheback (labeled as "flush") or interleavedly re-access (labeled as "int-re-acc") already raises the success rate from $22 \%$ to $28 \%$ and reduces the search latency from 49 ms to 46 ms . Since the target address $x$ is accessed by $C_{v}$ rather than


Fig. 6. Search latency and success rate on Intel processors when cacheback is applied.
Table 4. Cross-core CT on Intel processors using cacheback and surrogate targets.

| CPU | Baseline |  | Cacheback |  | Surrogate Target |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | latency | rate | latency | rate | latency | rate | -acc. | ction |
| i7-3770 | $13 \pm 4.5 \mathrm{~ms}$ | 80\% | $9.5 \pm 7.6 \mathrm{~ms}$ | 81\% | $6.4 \pm 2.6 \mathrm{~ms}$ | 64\% | $3.4 K$ | 89\% |
| i7-6700 | $49 \pm 47 \mathrm{~ms}$ | 22\% | $46 \pm 54 \mathrm{~ms}$ | 28\% | $31 \pm 44 \mathrm{~ms}$ | 16\% | 44 K | 68\% |
| i7-9700 | $44 \pm 39 \mathrm{~ms}$ | 20\% | $33 \pm 39 \mathrm{~ms}$ | $24 \%$ | $29 \pm 36 \mathrm{~ms}$ | 22\% | 43 K | 62\% |
| i7-11700 | $72 \pm 54 \mathrm{~ms}$ | 4.8\% | $69 \pm 58 \mathrm{~ms}$ | 6.6\% | $63 \pm 50 \mathrm{~ms}$ | 2.3\% | 125 K | 37\% |

$C_{a}$, caching back $\mathcal{E}$ would not evict $x$ out of the $C_{v}: \mathrm{L} 1$ and the thrashing access pattern observed by the private L1 and L2 caches means the benefit of flush is marginal. As shown in Fig. 5, flush reduces the search latency but also incurs a drop on the success rate. int-re-acc is unnecessary for the cross-core case. We therefore choose the basic cacheback (without flush or int-re-acc) as the default cacheback optimized CT algorithm. In the single-core case, caching back $\mathcal{E}$ has a much higher probability to evict the target address $x$ out of the private L1 and L2 caches than in the cross-core case. Consequently, applying cacheback itself leads to an substantial drop on the success rate. By applying both flush and int-re-acc, the success rate is raised from $16 \%$ to $19 \%$ while the search latency drops from 23 ms to 15 ms . We consequently define the cacheback with both flush and int-re-acc as the default cacheback optimized CT algorithm for single-core.

Fig. 6 demonstrates the performance improvement of cacheback optimized CT compared with the baseline CT on all the four Intel processors. The detail performance figures are also revealed in Table 4 for the cross-care case and Table 5 for the single-core case. The success rate is improved substantially on the more recent processors (later than the 6th generation) and this increase is most visible for the latest i7-11700 where the success rate is raised by $90 \%$ for the single-core case. As for the search latency, cacheback is able to reduce the search latency for all processors earlier than the 9th generation. Overall, cacheback is able to significantly improve the speed performance of CT on all the four tested Inel processors.

The surrogate targets (Section 4.4) optimization can significantly reduce the number of victim accesses by replacing the probing target from the target address $x$ to the first found congruent address $e_{0}$ stored in $\mathcal{E}$. We have tested the CT using surrogate targets on the four Intel processors and the detailed result is revealed in the right-most columns in Table 4. The number of victim accesses is reduced by $37 \%$ to, as high as, $89 \%$. This reduction proves the effectiveness of the optimization. The search latency is also significantly reduced to the range achieved by the single-core case. The reason is simply, once the probe target is

Table 5. Single-core CT on Intel processors using cacheback.

| CPU | Baseline |  |  | Cacheback |  |
| :--- | :--- | ---: | :--- | :--- | :--- |
|  | latency | rate |  | latency | rate |
| $\mathrm{i} 7-3770$ | $6.0 \pm 3.4 \mathrm{~ms}$ | $69 \%$ |  | $5.1 \pm 5.8 \mathrm{~ms}$ | $65 \%$ |
| $\mathrm{i} 7-6700$ | $23 \pm 21 \mathrm{~ms}$ | $16 \%$ |  | $15 \pm 19 \mathrm{~ms}$ | $19 \%$ |
| $\mathrm{i} 7-9700$ | $20 \pm 17 \mathrm{~ms}$ | $21 \%$ |  | $16 \pm 18 \mathrm{~ms}$ | $25 \%$ |
| $\mathrm{i} 7-11700$ | $12 \pm 4.4 \mathrm{~ms}$ | $2.1 \%$ |  | $13 \pm 9.8 \mathrm{~ms}$ | $3.9 \%$ |



Fig. 7. Success rate and search latency (both single-test and accumulated) of CT running a dual-core Rocket-Chip (L1 LRU and LLC random) with extended probing.
replaced with $e_{0}$, the time consuming cross-core probe becomes the much faster single-core probe. However, the success rate drops to slightly lower than the single-core case. The success rate of single-core case is typically lower than the cross-core case due to its higher noise level. In addition, probing the surrogate targets suffers from a slightly reduced success rate as the found $e_{0}$ might not be congruent with $x$ by a small probability due to false-positive errors.

Finally, we demonstrate the performance benefit of the extended probing (Section 4.3) again using a dual-core Rocket-Chip and configuring the replacement policies of the L1 cache to LRU and the LLC to random. Fig. 7 depicts the success rate and the search latency when the probing target is extended with 0 to 16 found congruent addresses stored in $\mathcal{E}$. The search latency is labeled as the "single-test latency" while the accumulated latency for eventually finding an eviction set (latency divided by success rate) is labeled as the "accu. latency". For both cross-core and single-core cases, extending the probe with found congruent addresses reduces the single-test search latency by increasing the success probability of probes. However, the success rate gradually drops with the number of extended probed addresses due to the increased probability of self-evicting the probe targets. The overall impact of applying extended probing is better presented by the accumulated search latency for finding an eviction set. Extending the probe with 2 to 6 addresses reduces the accumulated latency by around $20 \%$ for the cross-core case while extending the probe with 4 addresses reduces the accumulated latency by $18 \%$ for the single-core case. The result confirms that extending the probe with a small number of found congruent addresses can improve speed when the LLC adopts the random replacement policy.

## 6 Conclusion

In this paper, we have conducted the first systematic feasibility analysis of the CT algorithm. Besides the commonly known failing case where the LLC is exclusive or non-inclusive, two inclusive cache architectures are identified and verified as failing cases for the single-core CT. Three optimizations have been studied. The performance of the cacheback optimization has been significantly improved (especially for the single-core CT) by introducing flushing before cache back and interleaved re-access during the cacheback. The other two are newly proposed in this paper. Extended probing is effective in reducing the search latency by increasing the success probability of probes on cache architectures where the LLC adopts the random replacement policy. Surrogate targets is effective in reducing the number of victim accesses, which is hugely beneficial when the cross-core probing of the victim address is time consuming.

Acknowledgements. This research was supported by the National Natural Science Foundation of China (No. 62172406) and the CAS Pioneer Hundred Talents Program. Any opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding parties.

## References

1. Acriçmez, O., Schindler, W., Koç, Ç.K.: Cache based remote timing attack on the AES. In: Topics in Cryptology - CT-RSA 2007. pp. 271-286 (2007)
2. Alfs, G., Knupffer, N.: Intel fact sheet: Intel corporation's multicore architecture briefing (Mar 2008), https://www.intel.com/pressroom/archive/releases/2008/20080317fact.htm
3. Amid, A., Biancolin, D., Gonzalez, A., Grubb, D., Karandikar, S., Liew, H., Magyar, A., Mao, H., Ou, A., Pemberton, N., Rigge, P., Schmidt, C., Wright, J., Zhao, J., Shao, Y.S., Asanović, K., Nikolić, B.: Chipyard: Integrated design, simulation, and implementation framework for custom SoCs. IEEE Micro 40(4), 10-21 (2020)
4. Asanović, K., Avizienis, R., Bachrach, J., Beamer, S., Biancolin, D., Celio, C., Cook, H., Dabbelt, D., Hauser, J., Izraelevitz, A., Karandikar, S., Keller, B., Kim, D., Koenig, J., Lee, Y., Love, E., Maas, M., Magyar, A., Mao, H., Moreto, M., Ou, A., Patterson, D.A., Richards, B., Schmidt, C., Twigg, S., Vo, H., Waterman, A.: The Rocket chip generator. Tech. Rep. UCB/EECS-2016-17, University of California, Berkeley (Apr 2016)
5. Berg, C.: PLRU cache domino effects. In: Proceedings of the International Workshop on Worst-Case Execution Time Analysis (WCET) (2006)
6. Bernstein, D.J.: Cache-timing attacks on AES (2005), https://cr.yp.to/antiforgery/cachetiming-20050414.pdf
7. Genkin, D., Pachmanov, L., Tromer, E., Yarom, Y.: Drive-by key-extraction cache attacks from portable code. In: Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS). pp. 83-102. Springer (Jul 2018)
8. Gras, B., Razavi, K., Bosman, E., Bos, H., Giuffrida, C.: ASLR on the line: Practical cache attacks on the MMU. In: Proceedings of the Network and Distributed System Security Symposium (NDSS). Internet Society (Feb 2017)
9. Gruss, D., Maurice, C., Mangard, S.: Rowhammer.js: A remote software-induced fault attack in JavaScript. In: Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMV), pp. 300-321. Springer (Jul 2016)
10. İnci, M.S., Gulmezoglu, B., Irazoqui, G., Eisenbarth, T., Sunar, B.: Cache attacks enable bulk key recovery on the cloud. In: Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems (CHES). pp. 368-388. ICAR (Aug 2016)
11. Irazoqui, G., Inci, M.S., Eisenbarth, T., Sunar, B.: Wait a Minute! A fast, CrossVM Attack on AES. In: Stavrou, A., Bos, H., Portokalidis, G. (eds.) Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses (RAID). pp. 299-319 (2014)
12. Jaleel, A., Borch, E., Bhandaru, M., Jr., S.C.S., Emer, J.: Achieving non-inclusive cache performance with inclusive caches: Temporal locality aware (TLA) cache management policies. In: Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE (May 2020)
13. Jaleel, A., Theobald, K.B., Steely Jr., S.C., Emer, J.S.: High performance cache replacement using re-reference interval prediction (RRIP). In: Proceedings of the International Symposium on Computer Architecture (ISCA). pp. 60-71. ACM (Jun 2010)
14. Kocher, P., Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T., Schwarz, M., Yarom, Y.: Spectre attacks: Exploiting speculative execution. In: Proceedings of the IEEE Symposium on Security and Privacy (S\&P). pp. 19-37 (May 2019)
15. Lipp, M., Schwarz, M., Gruss, D., Prescher, T., Haas, W., Fogh, A., Horn, J., Mangard, S., Kocher, P., Genkin, D., Yarom, Y., Hamburg, M.: Meltdown: Reading kernel memory from user space. In: Proceedings of the USENIX Security Symposium (Security). pp. 973-990. USENIX Association (Aug 2018)
16. Liu, F., Yarom, Y., Ge, Q., Heiser, G., Lee, R.B.: Last-level cache side-channel attacks are practical. In: Proceedings of the IEEE Symposium on Security and Privacy (S\&P). IEEE (May 2015)
17. Maurice, C., Scouarnec, N.L., Neumann, C., Heen, O., Francillon, A.: Reverse engineering intel last-level cache complex addressing using performance counters. In: Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses (RAID). pp. 48-65. Springer (Nov 2015)
18. Percival, C.: Cache missing for fun and profit (2005)
19. Purnal, A., Giner, L., Gruß, D., Verbauwhede, I.: Systematic analysis of randomization-based protected cache architectures. In: Proceedings of the IEEE Symposium on Security and Privacy (S\&P). pp. 987-1002. IEEE (May 2021)
20. Purnal, A., Turan, F., Verbauwhede, I.: Prime+Scope: Overcoming the observer effect for high-precision cache contention attacks. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS). pp. 29062920. ACM (Nov 2021)
21. Purnal, A., Turan, F., Verbauwhede, I.: Double trouble: Combined heterogeneous attacks on non-inclusive cache hierarchies. In: Proceedings of the USENIX Security Symposium (Security). pp. 3647-3664. USENIX Association (Aug 2022)
22. Purnal, A., Verbauwhede, I.: Advanced profiling for probabilistic Prime+Probe attacks and covert channels in ScatterCache. ArXiv: cs.CR (Aug 2019)
23. Qureshi, M.K.: New attacks and defense for encrypted-address cache. In: Proceedings of the International Symposium on Computer Architecture (ISCA). pp. 360-371. ACM (Jun 2019)
24. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS). pp. 199-212. ACM (Nov 2009)
25. Savage, J.E., Zubair, M.: A unified model for multicore architectures. In: Proceedings of the International Forum on Next-Generation Multicore/Manycore Technologies (IFMT). p. 12. ACM (Nov 2008)
26. Shen, S., Li, Z., Song, W.: Methods of extracting parameters of the processor caches. In: Proceedings of the International Workshop on Security. pp. 47-65. Springer (Aug 2022)
27. Song, W., Liu, P.: Dynamically finding minimal eviction sets can be quicker than you think for side-channel attacks against the LLC. In: Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses (RAID). pp. 427-442. USENIX Association (Sep 2019)
28. Thoma, J.P., GÃijneysu, T.: Write me and I'll tell you secrets - write-after-write effects on Intel CPUs. In: Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses (RAID). ACM (Oct 2022)
29. Tromer, E., Osvik, D.A., Shamir, A.: Efficient Cache Attacks on AES, and Countermeasures. Journal of Cryptology 23(1), 37-71 (Jan 2010)
30. TÃşth, R., Faigl, Z., Szalay, M., Imre, S.: An advanced timing attack scheme on RSA. In: Networks 2008 - The 13th International Telecommunications Network Strategy and Planning Symposium. vol. Supplement, pp. 1-9 (2008)
31. Vila, P., Ganty, P., Guarnieri, M., KÃüpf, B.: CacheQuery: learning replacement policies from hardware caches. In: Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI). ACM (Jun 2020)
32. Vila, P., Köpf, B., Morales, J.F.: Theory and practice of finding eviction sets. In: Proceedings of the IEEE Symposium on Security and Privacy (S\&P). pp. 39-54. IEEE (May 2019)
33. Wong, H.: Intel Ivy Bridge cache replacement policy (Jan 2013), http://blog.stuffedcow.net/2013/01/ivb-cache-replacement/
34. Yan, M., Gopireddy, B., Shull, T., Torrellas, J.: Secure hierarchy-aware cache replacement policy (SHARP): defending against cache-based side channel atacks. In: Proceedings of the Annual International Symposium on Computer Architecture (ISCA). pp. 347-360. ACM (Jun 2017)
35. Yan, M., Sprabery, R., Gopireddy, B., Fletcher, C.W., Campbell, R.H., Torrellas, J.: Attack directories, not caches: Side-channel attacks in a non-inclusive world. In: Proceedings of the IEEE Symposium on Security and Privacy (S\&P). pp. 888-904. IEEE (May 2019)
36. Yarom, Y., Falkner, K.: FLUSH+RELOAD: A high resolution, low noise, L3 cache side-channel attack. In: Proceedings of the USENIX Security Symposium (Security). pp. 719-732. USENIX Association (Aug 2014)
37. Zhou, Y., Feng, D.: Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing (2005), http://eprint.iacr.org/2005/388

# Revisiting Key Switching Techniques with Applications to Light-Key FHE 

Ruida Wang ${ }^{1,2}$, Zhihao $\mathrm{Li}^{1,2}$, Benqiang Wei ${ }^{1,2}$, Chunling Chen ${ }^{1,2}$, Xianhui $\mathrm{Lu}^{1,2 *}$, and Kunpeng Wang ${ }^{1,2 *}$<br>${ }^{1}$ State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China<br>${ }^{2}$ School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China<br>luxianhui@iie.ac.cn; wangkunpeng@iie.ac.cn


#### Abstract

Fully Homomorphic Encryption (FHE) allows for data processing while it remains encrypted, enabling privacy-preserving outsourced computation. However, FHE faces challenges in real-world applications, such as communication overhead and storage limitations, due to the large size of its evaluation key. This paper revisits existing key switching algorithms widely used in FHE, which may account for over $90 \%$ of the total evaluation key size. Although these algorithms work towards the same goal, they differ significantly in functionality, computational complexity, noise management and key size. We close their functional gap and reanalyze them under a common standard, proposing theorems and comparative results to provide a flexible time-space trade-off when designing FHE applications. To validate the efficacy of our theoretical results, we propose a light-key bootstrapping method using a lower-sized key switching variant. This approach reduces the key size of the well-known GINX bootstrapping by a factor of $88.8 \%$. It also outperforms the state-of-the-art light-key FHE by reducing $48.4 \%$ bootstrapping key size and $8 \%$ transfer key size.


Keywords: FHE • Key Switching • Light-Key Bootstrapping.

## 1 Introduction

Fully Homomorphic Encryption (FHE) allows data to be processed while encrypted, enabling users to delegate computation to an untrusted party without the risk of data leakage. This opens up the potential for privacy-preserving outsourced computation in various applications, such as cloud computing [1,19], the internet of things (IoT) [26,29] and machine learning [20,10]. The process involves the client (data owner) encrypting their sensitive data, generating the necessary evaluation keys for homomorphic operations, and transmitting them to the server (computing party). The server performs homomorphic evaluations on the ciphertext and returns the encrypted results, as shown in fig.1.

One issue faced by Fully Homomorphic Encryption (FHE) is the storage and the communication cost. FHE is based on lattice encryption schemes, resulting in


Fig. 1. Client-server model of FHE applications. Evk denotes the evaluation keys.
large ciphertext and key sizes. In word-wise encryption schemes, the evaluation keys often have sizes of gigabytes $[14,4,16,17]$. While FHEW-like bit-wise encryption schemes reduce the evaluation key size by one order of magnitude, they still face limitations in real-world applications due to their key size of about 200 MB . More precisely, there is a strong preference for clients to generate and transmit keys with the smallest possible size. This is due to the fact that clients typically operate on devices with constrained computing power and limited storage space, sometimes even on mobile devices $[13,28]$.

From the server's perspective, research has demonstrated that hardware acceleration can yield over a ten times boost in the efficiency of homomorphic encryption operations $[15,30,25]$. However, these solutions are memory-constrained due to their limited on-chip storage. These challenges promote us to explore techniques to reduce the size of evaluation keys.

This paper concentrates on the key switching algorithm, whose key size may account for over $90 \%$ of the total evaluation key in FHEW-like schemes, as shown in tab.1.

| Methods | Evaluation key size | Key switching key size | Transfer key size |
| :---: | :---: | :---: | :---: |
| GINX ([21]) | 250 MB | $229.1 \mathrm{MB}(91.6 \%)$ | 16.48 MB |
| LFHE ([18]) | 175 MB | $84.6 \mathrm{MB}(48.3 \%)$ | 881 KB |
| GINX $_{\text {our }}$ | 27.91 MB | $27.2 \mathrm{~KB}(0.1 \%)$ | 13.96 MB |
| LFHE $_{\text {our }}$ | 90.38 MB | $54.2 \mathrm{~KB}(0.06 \%)$ | 810.1 KB |

Table 1. The proportion of key switching key size in the total evaluation key size of different bootstrapping methods. The parameters resources is within brackets. In the transfer model [18], the transfer key is a seed of the evaluation key. Sec.6.4 provides a detailed description of the transfer model.

Key switching is an essential operation in FHEW-like cryptosystems that enables changing the encryption key without revealing the plaintext. Various types of key switching have been described in this literature, including LWE-to-(R)LWE key switching, and RLWE-to-RLWE key switching. Chillotti et al.
shows that the former scheme can evaluate a linear Lipschitz morphism on ciphertext almost for free during switching keys [7]. Depending on the confidentiality of the morphism, it can be further divided into public functional key switching and private functional key switching. Even for the same switching type, there are different computation methods, these algorithms differ in functionality, key size, computational complexity, and noise management. A unified comparison is currently lacking, and there is no theoretical basis for selecting proper key switching algorithms when designing FHE applications. This motivates us to comprehensively revisit known key switching algorithms.

Functional Key Switching Algorithms. Our first contribution is to fill the functional gap in key switching algorithms. TFHE's key switching can compute a linear Lipschitz morphism while switching keys [7]. This property is not presented in the LWE-to-LWE key switching proposed by Chen et al. [3], or the commonly used RLWE-to-RLWE key switching algorithm. We fill this gap by decomposing all key switching algorithms into gadget products ${ }^{3}$, and embedding the linear Lipschitz morphism in it. The linear property ensures that the morphism can be correctly calculated by scalar multiplication, while the Lipschitz property helps manage noise growth. As a result, we provide functional variants of all known key switching algorithms, which may have independent interests beyond this paper. For instance, we demonstrate that the scheme switching algorithm [9] (or the same EvalSquareMult algorithm [18]) can be regarded as a specific case of our proposed RLWE-to-RLWE private functional key switching algorithm for the morphism $f(x)=\mathbf{s k} \cdot x$, where $\mathbf{s k}$ is the secret key.

Comparison Between Key Switching Algorithms. Comparing key switching algorithms can be challenging since they are proposed and analyzed using different baselines, such as algebraic structures, the key distributions, and the gadget decomposition methods ${ }^{4}$. In this work, we present a comprehensive reanalysis of the existing key switching algorithms and our proposed functional variants under a common standard. We use the power of two cyclotomic ring, which is commonly used in FHE schemes, binary key distribution, and the canonical approximate gadget decomposition [7]. We propose noise growth formulas and provide performance data in terms of key sizes and computational complexity. Our work serves as a theoretical basis for the practical selection of key switching algorithms when designing FHE applications.

Light-Key Bootstrapping Algorithm. To validate the efficacy of our theoretical results, we propose the light-key bootstrapping variants using a lowersized key switching algorithm. For the well-known GINX bootstrapping, this approach reduces the bootstrapping key size by $88.8 \%$ and the transfer key

[^34]size by 15.3 \%. For the state-of-the-art light-key bootstrapping, this approach outperforms Kim et al.'s LFHE method [18] by reducing 48.4 \% bootstrapping key size and $8 \%$ transfer key size.

Related Work. Fig. 1 illustrates that the client must generate and transmit two components: the ciphertext and the evaluation keys. This paper focuses on reducing the size of the evaluation key. However, the ciphertext size is also significantly larger than plaintext due to the lattice-based encryption. Currently, Naehrig et al. have introduced techniques named hybrid homomorphic encryption (HHE or transciphering) [24,2,11,8]. This technique allows the client to encrypt messages with a symmetric cipher. The server then evaluates the decryption circuit homomorphically to obtain the ciphertext under HE form for further processing. Our future work involves integrating HHE with our research, to develop fully homomorphic encryption applications with minimal transmission size.

Organization. The rest of the paper is organized as follows: sec. 2 reviews the notations and crypto primitives; sec. 3 revisits the gadget product as the basic computational unit of key switching algorithms; sec. 4 and sec. 5 analyzes the LWE-to-LWE key switching algorithms and RLWE-to-RLWE key switching algorithms, respectively; sec. 6 constructs the light-key bootstrapping algorithm based on the analysis results; sec. 7 concludes the paper.

## 2 Preliminaries

### 2.1 Notations

Let $\mathbb{A}$ be a set. Define $\mathbb{A}^{n}$ as the set of vectors with $n$ elements in $\mathbb{A}, \mathbb{A}_{q}$ as the set $\mathbb{A}$ module $q$, where the elements' scope is $[-q / 2, q / 2) \cap \mathbb{A}$. Use $\mathbb{Z}$ to denote the set of integers, $\mathbb{R}$ to denote the set of real numbers, and $\mathbb{B}=\mathbb{Z}_{2}$ represents the set of binary numbers. Denote $\mathcal{R}$ as the set of integer coefficient polynomials modulo $X^{N}+1$, where $N$ is a power of 2 Then $\mathcal{R}$ is the $2 N$-th cyclotomic ring.

Use regular letters to represent (modular) integers like $a \in \mathbb{Z}_{q}$, while bold letters to represent polynomials $\mathbf{a} \in \mathcal{R}$ or vectors $\mathbf{a} \in \mathbb{Z}^{n}$. The notation $a_{i}$ refers to the $i$-th coefficient/term of $\mathbf{a}$. The floor, ceiling, and rounding functions are written as $\lfloor\cdot\rfloor,\lceil\cdot\rceil\lfloor\cdot\rceil$, respectively. A function $f$ is R-Lipschitz means that it satisfies $\|f(x)-f(y)\|_{\infty} \leq R\|x-y\|_{\infty}$, where $\|\cdot\|_{\infty}$ is the infinity norm.

### 2.2 Gadget Decomposition

Given a gadget vector $\mathbf{v}=\left(v_{0}, v_{1}, \ldots, v_{l-1}\right)$, the gadget decomposition of a ring element $\mathbf{t} \in R$ is to find $\left(\mathbf{t}_{0}, \ldots, \mathbf{t}_{l-1}\right)$ to minimize the decomposition error $\varepsilon_{\text {gadget }}(\mathbf{t})=\sum_{i} v_{i} \mathbf{t}_{i}-\mathbf{t} . \epsilon$ denotes its infinite norm, that is, $\left\|\sum_{i} v_{i} \mathbf{t}_{i}-\mathbf{t}\right\|_{\infty} \leq \epsilon$. In this paper, we use the canonical approximate gadget decomposition, where $\left.\mathbf{v}=\left(\left\lceil\frac{q}{\left.B^{l}\right\rceil}\right\rceil \frac{q}{B^{l}}\right\rceil B, \ldots,\left\lceil\frac{q}{B^{l}}\right\rceil B^{l-1}\right)$, thus $\epsilon \leq \frac{1}{2}\left\lceil\frac{q}{B^{l}}\right\rceil$. We say $B$ is the gadget base and $l$ is the gadget length.

### 2.3 Learning with Errors

The security of FHEW-like cryptosystem is based on the (ring) learning with errors problem $[27,22]$. We summarize the three kinds of ciphertexts as follow:

- LWE: Giving positive integers $n$ and $q$, the LWE encryption of the message $m \in \mathbb{Z}$ is a vector $(\mathbf{a}, b) \in \mathbb{Z}_{q}^{n+1}$, where $b=-\mathbf{a} \cdot \mathbf{s k}+m+e$. The vector a is uniformly sampled from $\mathbb{Z}_{q}^{n}$, the secret key sk is sampled from a key distribution $\chi$, the error $e$ is sampled from an error distribution $\chi^{\prime}$.
- RLWE: RLWE is a ring version of LWE on $\mathcal{R}_{q}$. The RLWE encryption of the message $\mathbf{m} \in \mathcal{R}_{q}$ is a pair $(\mathbf{a}, \mathbf{b}) \in \mathcal{R}_{q}^{n+1}$, where $\mathbf{b}=-\mathbf{a} \cdot \mathbf{s k}+\mathbf{m}+\mathbf{e}$. The vector a is uniformly sampled from $\mathcal{R}_{q}$, the secret key sk is sampled from a key distribution $\chi$, and each coefficient of the error $e_{i}$ is sampled from $\chi^{\prime}$.
- RGSW: The RGSW encryption of the message $\mathbf{m} \in \mathcal{R}_{q}$ can be expressed as: $\operatorname{RGSW}_{\mathbf{s k}}(\mathbf{m})=\left(\operatorname{RLWE}_{\mathbf{s k}}^{\prime}(\mathbf{s k} \cdot \mathbf{m}), \operatorname{RLWE}_{\mathbf{s k}}^{\prime}(\mathbf{m})\right)$, where RLWE' is the gadget RLWE ciphertext defined as follows:

Given a gadget vector $\mathbf{v}=\left(v_{0}, v_{1}, \ldots, v_{l-1}\right)$, the notion (R)LWE' refers to the gadget (R)LWE ciphertext is defined as:

$$
\begin{aligned}
& \operatorname{LWE}_{\mathbf{s k}}^{\prime}(m)=\left(\operatorname{LWE}_{\mathbf{s k}}\left(v_{0} \cdot m\right), \operatorname{LWE}_{\mathbf{s k}}\left(v_{1} \cdot m\right), \ldots, \operatorname{LWE}_{\mathbf{s k}}\left(v_{l-1} \cdot m\right)\right) \\
& \operatorname{RLWE}_{\mathbf{s k}}^{\prime}(\mathbf{m})=\left(\operatorname{RLWE}_{\mathbf{s k}}\left(v_{0} \cdot \mathbf{m}\right), \operatorname{RLWE}_{\mathbf{s k}}\left(v_{1} \cdot \mathbf{m}\right), \ldots, \operatorname{RLWE}_{\mathbf{s k}}\left(v_{l-1} \cdot \mathbf{m}\right)\right)
\end{aligned}
$$

Remark 1. These definitions (following Micciancio and Polyakov [23]) use different notions compared to the original TFHE papers [5,6,7]. Specifically, TFHE uses real torus $\mathbb{T}=\mathbb{R} / \mathbb{Z}$ and $\mathbb{T}_{N}[X]=\mathcal{R} / \mathbb{Z}$ to describe the message and ciphertext spaces, but implements $\mathbb{T}$ by $\mathbb{Z}_{q}$ with $q=2^{32}$ or $q=2^{64}$. Thus we straightforwardly use $\mathbb{Z}_{q}$ instead of $\mathbb{T}$.

Remark 2. In FHEW-like cryptosystem, the gadget (R)LWE is mainly used as the evaluation key and appears as an auxiliary input in algorithms such as key switching. To simplify the presentation and facilitate the understanding of the key switching algorithm, which is the main focus of this paper, we provide a formal definition and notation of gadget (R)LWE.

### 2.4 Bootstrapping

The error rate of the LWE/RLWE ciphertext will significantly affect the decryption failure probability, which can be calculated by $1-\operatorname{erf}\left(\frac{q}{8 \sqrt{2} \sigma}\right)$, where $\sigma$ is the standard deviation of the error. We then introduce the bootstrapping algorithm to reduce the error rate. FHEW-like bootstrapping can evaluate a 1 -in/1-out LUT function while refreshing ciphertext noise. It typically contains the following operations: blind rotation (BR), sample extraction (SE), key switching, and modulus switching (MS).

As the goal of bootstrapping is to refresh the noise in the ciphertext, it is necessary to pay extra attention and precisely control the noise generated in each step of the bootstrapping algorithm itself. The basic strategy is to execute the BR step, which mainly generates the new noise, under a large modulus. Then recovering the LWE ciphertext form through SE, reducing the modulus while eliminating the blind rotation noise size through MS, and recovering the original key through key switching algorithm. We introduce two typical bootstrapping work flows as follows:

GINX Bootstrapping [5,6,7]. $\operatorname{LWE}_{571,2^{11}} \xrightarrow{\mathrm{BR}} \operatorname{RLWE}_{1024,2^{25}} \xrightarrow{\mathrm{SE}} \mathrm{LWE}_{1024,2^{25}}$ $\xrightarrow{\mathrm{MS}} \mathrm{LWE}_{1024,2^{14}} \xrightarrow{\mathrm{LtL}} \mathrm{LWE}_{571,2^{14}} \xrightarrow{\mathrm{MS}} \mathrm{LWE}_{571,2^{11}}$.

Remark 3. The above parameters are taken from Lee's recent article [21], with a security level of 128 -bit. Due to the update of attack methods, the security level of the parameters in TFHE articles $[5,6,7]$ has been reduced to 115 -bit.

LFHE Bootstrapping [18]. LWE $_{571,2^{11}} \xrightarrow{\text { BR }}$ RLWE $_{2048,2^{54}} \xrightarrow{\text { MS }}$ RLWE $_{2048,2^{27}}$ $\xrightarrow{\text { RtR }}$ RLWE $_{1024,2^{27}} \xrightarrow{\text { SE }} \operatorname{LWE}_{1024,2^{27}} \xrightarrow{\text { MS }}$ LWE $_{1024,2^{14}} \xrightarrow{\text { LtL }} \operatorname{LWE}_{571,2^{14}} \xrightarrow{\text { MS }}$ $\mathrm{LWE}_{571,2^{11}}$.

To display the switching of the keys and modulus, we use the form ( R$) \mathrm{LWE}_{n, q}$ to represent ciphertexts, where $n$ is the dimension of the secret key vector (or polynomial) and $q$ represents the modulus of the ciphertext. LtL stands for LWE to LWE key switching, and both of the above bootstrapping algorithms use its storage version (for a summary and comparison between different versions, see sec. 4). RtR stands for RLWE to RLWE key switching.

## 3 Gadget Products

The gadget product is used to calculate the scalar multiplication in FHEW-like cryptosystem. It works by gadget decomposing the plaintext scalar and then multiplying the corresponding gadget (R)LWE ciphertexts. This algorithm can reduce the noise growth of scalar multiplication and is widely used in core algorithms such as external product [7] and key switching. This section summarizes three types of gadget products, and analyze their differences in terms of noise growth, auxiliary input size and computational complexity. The first one is the canonical gadget product primarily used for external product. It was first abstracted as a separate algorithm by Micheli et al. in 2023 [9].

Gadget Product: The canonical gadget product $\odot: \mathbb{Z} \times(\mathrm{R}) \mathrm{LWE}^{\prime} \rightarrow(\mathrm{R}) \mathrm{LWE}$ is defined as:

$$
\begin{aligned}
t \odot(\mathrm{R}) \mathrm{LWE}_{\mathbf{s k}}^{\prime}(\mathbf{m}) & :=\sum_{i=0}^{l-1} t_{i} \cdot(\mathrm{R}) \mathrm{LWE}_{\mathbf{s k}}\left(v_{i} \cdot \mathbf{m}\right) \\
& =(\mathrm{R}) \mathrm{LWE}_{\mathbf{s k}}\left(\sum_{i=0}^{l-1} v_{i} \cdot t_{i} \cdot \mathbf{m}\right) \\
& =(\mathrm{R}) \operatorname{LWE}_{\mathbf{s k}}\left(t \cdot \mathbf{m}+\varepsilon_{\text {gadget }}(t) \cdot \mathbf{m}\right)
\end{aligned}
$$

Lemma 1. [18] Let $B$ and $l$ denote the base and the length of the gadget decomposition, respectively, then the error variance of the result of the gadget product is bounded by

$$
\sigma_{\odot, \text { input }}^{2} \leq \frac{1}{12} l B^{2} \sigma_{\text {input }}^{2}+\frac{1}{3} \operatorname{Var}(\mathbf{m}) \epsilon^{2}
$$

where $\sigma_{\text {input }}^{2}$ is the error variance of the input $\mathrm{LWE}^{\prime}$ ciphertext, and $\operatorname{Var}(\mathbf{m})$ is the variance of the message $\mathbf{m}$.

Lemma. 1 is derived from [18] proposition. 1 with the fact $\epsilon \leq \frac{1}{2}\left\lceil\frac{q}{B^{l}}\right\rceil$. This method use the modular multiplication to compute the gadget product. However, for a fixed input (R)LWE $\mathbf{s k}_{\prime}^{\prime}(\mathbf{m})$, there is an time-space trade-off that reduces the computational complexity by using additional storage. Specifically, since the range of $t_{i}$ is bounded by the gadget base $B$, one can pre-compute and store all possible values of $(\mathrm{R}) \mathrm{LWE}_{\mathbf{s k}}^{\prime}\left(v_{i} \cdot t_{i} \cdot \mathbf{m}\right)$, then use modular addition instead of modular multiplication. This method was first used in the FHEW bootstrapping algorithm proposed by Ducas et al. in 2015 [12], which inspired us to summarize a store version of the gadget product. We denote this method using operator $\oplus$ :

Gadget Product (Store Version): The store version $\oplus: \mathbb{Z} \times(\mathrm{R}) \mathrm{LWE}^{\prime} \rightarrow$ (R)LWE is defined as:

$$
\begin{aligned}
& t \oplus\left(\mathrm{R}_{\mathrm{LWE}}^{\mathbf{s k}}\right. \\
& \prime(\mathbf{m}):=\sum_{i=0}^{l-1}\left(\mathrm{R}_{\mathrm{L}} \mathrm{LWE}_{\mathbf{s k}}^{\prime}\left(v_{i} \cdot t_{i} \cdot \mathbf{m}\right)\right. \\
&=(\mathrm{R}) \mathrm{LWE}_{\mathbf{s k}}\left(\sum_{i=0}^{l-1} v_{i} \cdot t_{i} \cdot \mathbf{m}\right) \\
&=(\mathrm{R}) \mathrm{LWE}_{\mathbf{s k}}\left(t \cdot \mathbf{m}+\varepsilon_{\text {gadget }}(t) \cdot \mathbf{m}\right)
\end{aligned}
$$

Corollary 1. Let $B$ and $l$ denote the base and the length of the gadget decomposition, respectively, then the error variance of the result of the gadget product (store version) is bounded by

$$
\sigma_{\oplus, \text { input }}^{2} \leq l \sigma_{\text {input }}^{2}+\frac{1}{3} \operatorname{Var}(\mathbf{m}) \epsilon^{2}
$$

where $\sigma_{\text {input }}^{2}$ is the error variance of the input $\mathrm{LWE}^{\prime}$ ciphertext, and $\operatorname{Var}(\mathbf{m})$ is the variance of the message $\mathbf{m}$.

The store version of gadget product use $l$ times modular addition instead of modular multiplication. Thus corollary. 1 can be directly derived from lemma. 1 by replacing multiplication error growth with addition error growth.

Lastly, we introduce the ring version of the gadget product, denoted by $\odot_{R}$ :
Gadget Product (Ring Version): The Ring gadget product $\odot_{R}: \mathcal{R} \times$ RLWE $^{\prime} \rightarrow$ RLWE is defined as:

$$
\begin{aligned}
\mathbf{t} \odot_{R} \operatorname{RLWE}_{\mathbf{s k}}^{\prime}(\mathbf{m}) & :=\sum_{i=0}^{l-1} \mathbf{t}_{i} \cdot \operatorname{RLWE}_{\mathbf{s k}}\left(v_{i} \cdot \mathbf{m}\right) \\
& =\operatorname{RLWE}_{\mathbf{s k}}\left(\sum_{i=0}^{l-1} v_{i} \cdot \mathbf{t}_{i} \cdot \mathbf{m}\right) \\
& =\operatorname{RLWE}_{\mathbf{s k}}\left(\mathbf{t} \cdot \mathbf{m}+\varepsilon_{\text {gadget }}(\mathbf{t}) \cdot \mathbf{m}\right),
\end{aligned}
$$

Corollary 2. Let $n$ denote the dimension of the ring polynomial of RLWE ciphertexts, $B$ and $l$ denote the base and the length of the gadget decomposition, respectively, then the error variance of the result of the gadget product is bounded by

$$
\sigma_{\oplus_{R}, \text { input }}^{2} \leq \frac{1}{12} n l B^{2} \sigma_{\text {input }}^{2}+\frac{1}{3} n \operatorname{Var}(\mathbf{m}) \epsilon^{2}
$$

 the variance of $\mathbf{m}$.

This algorithm is a ring version of the gadget product. Notice that since polynomial dimension $n$ causes an exponential increase in polynomial gadget decomposition results, it is impractical to accelerate computation by pre-storing all possible $\mathrm{RLWE}_{\mathbf{s k}}^{\prime}\left(v_{i} \cdot \mathbf{t}_{i} \cdot \mathbf{m}\right)$. In other words, the store version of the ring gadget product is not practical and we do not consider it. The error growth of the ring gadget product needs to take into account the expansion factor of the ring. In this paper, we use a power-of-two cyclotomic ring, with an expansion factor of $\sqrt{n}$ for the two-norm, resulting in a factor of $n$ when evaluating the noise variance. Then corollary. 2 can be derived from lemma.1.

Comparison. The computational complexity and auxiliary input size of the three gadget products are listed in tab.2. From lemma. 1 and the corollaries in this section, we can conclude that in terms of error growth, $\odot_{R}=\odot>\oplus$. From tab.2, it is evident that in terms of computational complexity, $\odot_{R}>\odot>\oplus$, in terms of the size of auxiliary inputs, $\oplus>\odot_{R}=\odot$.

As the key switching algorithm is always a combination of scalar multiplication and addition, these algorithms can be re-written using the three types of gadget products. This novel perspective makes it easier to examine key switching algorithms, provides insights into their comparison in terms of correctness, error growth, computational complexity, and key size. Our analysis can serve as a guideline for time-space trade-offs in the implementation of the key switching.

| Calculation | Computation Complexity | Auxiliary Input (in (R)LWE') |
| :---: | :---: | :---: |
| $\odot$ | $\ln \mathrm{MM}$ | 1 |
| $\oplus$ | $l \mathrm{MA}$ | $B$ |
| $\odot_{R}$ | $l \mathrm{NTT}+\ln \mathrm{MM}$ | 1 |

Table 2. Comparison between different version of the gadget product, where $l$ and $B$ are the gadget length and base, MA and MM denote the modular addition and modular multiplication operations. NTT is the number theoretic transform algorithm (with $O(\ln \log n)$ MM computational complexity) used in polynomial multiplication.

We then revisit LWE-to-LWE key switching algorithms in sec.4, and RLWE-toRLWE key switching algorithms in sec.5.

## 4 LWE-to-LWE Key Switching

Chillotti et al. proposed in the TFHE series $[5,6,7]$ that their key switching algorithm can calculate a R-Lipschitz linear morphism while switching keys. This section generalizes all existing LWE-to-LWE key switching algorithms into functional versions, and classifies key switching algorithms into public functional key switching and private functional key switching (following Chilloti et al. [7]) based on whether the Lipschitz morphism needs to be kept confidential.


Fig. 2. Six LWE-to-LWE key switching algorithms revisited in this section.

### 4.1 Public Functional Key Switching

## LWE-to-LWE Using Canonical Gadget Product.

- Input: $\operatorname{LWE}_{\mathbf{s k}}(m)=(\mathbf{a}, b)$, and a public R-Lipschitz linear morphism $f$ : $\mathbb{Z} \rightarrow \mathbb{Z}$
- Switching key:LtLK $=\operatorname{LWE}_{\mathbf{s k}^{\prime}}^{\prime}\left(s k_{i}\right)_{i \in[1, n]}$
- Output: $\operatorname{LWE}_{\mathbf{s k}^{\prime}}(f(m))$
- Algorithm:

$$
\mathrm{Lt}_{\mathbf{s k} \rightarrow \mathbf{s k}} \mathrm{LW}^{\prime}\left(\mathrm{LWE}_{\mathbf{s k}}(m)\right):=\sum_{i=1}^{n} f\left(a_{i}\right) \odot \mathrm{LWE}_{\mathbf{s k}^{\prime}}^{\prime}\left(s k_{i}\right)+(0, f(b))
$$

This algorithm was first proposed by Chillotti et al. [7], and we formalize it using gadget product. We then re-analyze the error growth of this algorithm, and update the theorem 4.1 in [7] for two reasons.

Firstly, TFHE used binary gadget decomposition for scalars in the key switching algorithm. But currently FHEW-like cryptosystems generally use the standard approximate gadget decomposition (power-of- $B$ ), as what we considered. Secondly, TFHE utilized the torus algebraic structure in its theoretical analysis, rather than the power of 2 cyclotomic ring used in the implementation. Thus it did not consider the coefficient $1 / 12$ when calculating the variance of the uniform distribution, resulting in a less compact error bound in theorem 4.1 [7].

## Correctness and error analysis:

Theorem 1. Let $n$ denote the dimension of the LWE ciphertexts, $B$ and $l$ denote the base and the length of the gadget decomposition, respectively, then the error variance of the result of the LWE to LWE public functional key switching algorithm is bounded by:

$$
\sigma_{\mathrm{LtL}}^{2} \leq \frac{1}{12} n l B^{2} \sigma_{\mathrm{LtLK}}^{2}+\frac{1}{6} n \epsilon^{2}+R^{2} \sigma_{\text {input }}^{2}
$$

where $\epsilon$ is the gadget decomposition error, $\sigma_{\text {input }}^{2}$ is the error variance of the input LWE ciphertext, and $\sigma_{\text {LtLK }}^{2}$ is the error variance of the switching key.

Proof. Basing the correctness of the gadget product, we have,

$$
\begin{aligned}
& \sum_{i=1}^{n} f\left(a_{i}\right) \odot \operatorname{LWE}_{\mathbf{s k}^{\prime}}^{\prime}\left(s k_{i}\right)+(0, f(b)) \\
= & \operatorname{LWE}_{\mathbf{s k}^{\prime}}\left(\sum_{i=1}^{n} f\left(a_{i} \cdot s k_{i}\right)+f(b)\right) \\
= & \operatorname{LWE}_{\mathbf{s k}^{\prime}}(f(m)+f(e)),
\end{aligned}
$$

then we measure the error variance based on lemma.1:

$$
\sigma_{\mathrm{LtL}}^{2}=n \sigma_{\odot, \mathrm{LtLK}}^{2}+\operatorname{Var}(f(e)) \leq \frac{1}{12} n l B^{2} \sigma_{\mathrm{LtLK}}^{2}+\frac{1}{6} n \epsilon^{2}+R^{2} \sigma_{\text {input }}^{2}
$$

Store Version. As we analyzed in sec.3, the LtL algorithm, which uses the canonical gadget product, also has a corresponding store version. It only requires modifications to the auxiliary input and calculation method:

- Switching key:LtLK $=\operatorname{LWE}_{\mathbf{s k}^{\prime}}^{\prime}\left(j \cdot s k_{i}\right)_{i \in[1, n], j \in[0, B-1]}$
- Algorithm:

$$
\mathrm{LtL}_{\mathbf{s k} \rightarrow \mathbf{s k}}{ }^{f}\left(\mathrm{LWE}_{\mathbf{s k}}(m)\right):=\sum_{i=1}^{n} f\left(a_{i}\right) \oplus \mathrm{LWE}_{\mathbf{s k}^{\prime}}^{\prime}\left(s k_{i}\right)+(0, f(b)) .
$$

It can be derived from tab. 2 that, the store version is faster and has smaller noise growth compared to the canonical LtL algorithm. However, the trade-off is an increase in key size by a factor of $B$, where $B$ is the base for gadget decomposition.

LWE-to-LWE Using Ring Gadget Product.

- Input: $\operatorname{LWE}_{s \vec{k}}(m)=(\vec{a}, b)$, and a public R-Lipschitz linear morphism $f$ : $\mathbb{Z} \rightarrow \mathbb{Z}$
- Switching key: $\operatorname{LtL}_{2} \mathrm{~K}=\operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(\mathbf{s k})$, where $\mathbf{s k}=\sum_{i=0}^{l-1} s k_{i} X^{-i}, \mathbf{s k}^{\prime}=$ $\sum_{i=0}^{l-1} s k_{i}^{\prime} X^{-i}$
- Output: $\operatorname{LWE}_{\overrightarrow{s k^{\prime}}}(f(m))=\left(\vec{a}^{\prime}, b^{\prime}\right)$
- Algorithm:

$$
\begin{gathered}
\left(\mathbf{a}^{\prime}, \mathbf{b}^{\prime}\right):=\sum_{i=1}^{n} f\left(a_{i}\right) X^{i} \odot_{R} \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(\mathbf{s k})+(0, f(b)), \\
\operatorname{LtL}_{2}{ }_{s \overrightarrow{s k} \rightarrow s \vec{k}^{\prime}}^{f}\left(\operatorname{LWE}_{\overrightarrow{s k}}(m)\right):=\left(a_{0}^{\prime}, a_{1}^{\prime}, \ldots, a_{n-1}^{\prime}, b_{0}^{\prime}\right) .
\end{gathered}
$$

Remark 4. This algorithm involves the conversion between vectors and polynomials. Thus to avoid confusion, we use $\vec{a}$ to represent vectors in this algorithm, while a to represent polynomials. The notation $a_{i}$ is the $i$-th term of the vector $\vec{a}$, and $[\mathbf{a}]_{i}$ is the $i$-th coefficient of the polynomial a.

This switching method was proposed by Chen et al. [3]. We formalize it using ring gadget product, and first extend it to the functional version. Therefore, the error growth of this algorithm must take into account the Lipschitz morphism. In addition, Chen et al. only considered the exact gadget decomposition, which is a special case ( $q \leq B^{l}$ ) of the canonical approximate gadget decomposition we use. This also prompts us to re-analyze the error.

## Correctness and error analysis:

Theorem 2. Let n denote the dimension of the LWE ciphertexts, $B$ and $l$ denote the base and the length of the gadget decomposition, respectively, then the error variance of the result of the LWE to LWE using $\operatorname{RtR}$ algorithm is bounded by:

$$
\sigma_{\mathrm{Ltt}_{2}}^{2} \leq \frac{1}{12} n l B^{2} \sigma_{\mathrm{Ltt}_{2} \mathrm{~K}}^{2}+\frac{1}{6} n \epsilon^{2}+R^{2} \sigma_{\text {input }}^{2},
$$

where $\sigma_{\text {input }}^{2}$ is the error variance of the input LWE ciphertext, and $\sigma_{\mathrm{Ltt}_{2} \mathrm{~K}}^{2}$ is the error variance of the switching key.

| Method | Computation complexity | Key size (in bits) |
| :---: | :---: | :---: |
| $\mathrm{LtL}, \odot$ | $O\left(\ln ^{2}\right) \mathrm{MM}$ | $\ln (n+1) \log q$ |
| $\mathrm{LtL}, \oplus$ | $(\ln +n+1) \mathrm{MA}$ | $B \ln (n+1) \log q$ |
| $\mathrm{LtL}_{2}, \odot_{R}$ | $O(\ln \log n) \mathrm{MM}$ | $2 \ln \log q$ |

Table 3. Comparison between different version of the public LWE-to-LWE key switching, where $q$ is the ciphertext modulus, $l$ and $B$ are the gadget length and base, MA and MM denote the modular addition and modular multiplication operations.

Proof. Basing the correctness of the ring gadget product, we have,

$$
\begin{aligned}
b_{0}^{\prime}+\sum_{i=1}^{n} a_{i}^{\prime} s k_{i}^{\prime} & =\left[\mathbf{b}^{\prime}+\mathbf{a}^{\prime} \cdot \mathbf{s k}^{\prime}\right]_{0} \\
& =\sum_{i=1}^{n} f\left(a_{i} \cdot s k_{i}\right)+f(b) \\
& =f(m)+f(e)
\end{aligned}
$$

thus $\left(a_{0}^{\prime}, a_{1}^{\prime}, \ldots, a_{n-1}^{\prime}, b_{0}^{\prime}\right)$ is the LWE ciphertext of $f(m)$ under secret key $\overrightarrow{s k}^{\prime}$, then we measure the error variance based on corollary.2:

$$
\sigma_{\mathrm{LtL}_{2}}^{2}=\sigma_{\odot_{R}, \mathrm{LtL}_{2} \mathrm{~K}}^{2}+\operatorname{Var}(f(e)) \leq \frac{1}{12} n l B^{2} \sigma_{\mathrm{LtLK}}^{2}+\frac{1}{6} n \epsilon^{2}+R^{2} \sigma_{\text {input }}^{2}
$$

Comparison. The computational complexity and key size of LWE-to-LWE public functional key switching algorithms are listed in tab.3. From the theorems in this section, we can conclude that in terms of error growth, we have $(L t L, \odot)=$ $\left(\mathrm{LtL}_{2}, \odot_{R}\right)>(\mathrm{LtL}, \oplus)$. From tab.2, it is evident that $(\mathrm{LtL}, \odot)>\left(\mathrm{LtL}_{2}, \odot_{R}\right)>$ $(\mathrm{LtL}, \oplus)$ in computational complexity. In terms of the key size, we have $(\mathrm{LtL}, \oplus)>$ $(\mathrm{LtL}, \odot)>\left(\mathrm{LtL}_{2}, \odot_{R}\right)$.

Comparison results indicate that $(\mathrm{LtL}, \odot)$ is inferior to $\left(\mathrm{LtL}_{2}, \odot_{R}\right)$ in all aspects. Thus when we care more about the computational efficiency and error control of the algorithm, $(\mathrm{LtL}, \oplus)$ is the best choice. On the other hand, if key size (which affects transfer size and storage space) is of greater concern, we should use $\left(L_{t} L_{2}, \odot_{R}\right)$ as the substitute.

### 4.2 Private Functional Key Switching

In the public functional key switching algorithm, the Lipschitz morphism $f$ is used as a public input. However, $f$ should be kept confidential in some cases. For example, it is related to the secret key or derived from a protected model. Chillotti et al. proposed private functional key switching algorithm for this situation [7], where the morphism $f$ is secretly encoded within the algorithm's switching key. In this section, we first revisit this canonical algorithm. Then we
introduce two novel algorithms. The first is the store version of private functional key switching, which we extended based on the method of Ducas et al. [12]. The second is the ring version, extended based on Chen et al.'s methods [3].

## Private LWE-to-LWE Using Canonical Gadget Product.

- Input: $\operatorname{LWE}_{\mathbf{s k}}(m)=(\mathbf{a}, b)$
- Switching key: PLtLK $=\left(\operatorname{LWE}_{\mathbf{s k}^{\prime}}^{\prime}\left(f\left(s k_{i}\right)\right)_{i \in[1, n]}, \operatorname{LWE}_{\mathbf{s k}^{\prime}}^{\prime}(f(1))\right)$, where $f$ : $\mathbb{Z} \rightarrow \mathbb{Z}$ is a private R -Lipschitz linear morphism
- Output: $\operatorname{LWE}_{\mathbf{s k}^{\prime}}(f(m))=\left(\mathbf{a}^{\prime}, b^{\prime}\right)$
- Algorithm:

$$
\operatorname{PLtL}_{\mathbf{s k} \rightarrow \mathbf{s k}} \mathbf{k}^{\prime}\left(\operatorname{LWE}_{\mathbf{s k}}(m)\right):=\sum_{i=1}^{n} a_{i} \odot \operatorname{LWE}_{\mathbf{s k}} \mathbf{k}^{\prime}\left(f\left(s k_{i}\right)\right)+b \odot \operatorname{LWE}_{\mathbf{s k}^{\prime}}^{\prime}(f(1))
$$

Store Version. This version only requires modifications to the switching key and calculation method:

- Switching key:PLtLK $=\left(\operatorname{LWE}_{\mathbf{s k}^{\prime}}^{\prime}\left(j \cdot f\left(s k_{i}\right)\right), \operatorname{LWE}_{\mathbf{s k}^{\prime}}^{\prime}(j \cdot f(1))\right)$, where $i \in$ $[1, n], j \in[0, B-1], f: \mathbb{Z} \rightarrow \mathbb{Z}$ is a private R-Lipschitz linear morphism
- Algorithm:

$$
\operatorname{PLtL}_{\mathbf{s k} \rightarrow \mathbf{s k}} \mathbf{k}^{\prime}\left(\operatorname{LWE}_{\mathbf{s k}}(m)\right):=\sum_{i=1}^{n} a_{i} \oplus \operatorname{LWE}_{\mathbf{s \mathbf { k } ^ { \prime }}}^{\prime}\left(f\left(s k_{i}\right)\right)+b \oplus \operatorname{LWE}_{\mathbf{s k}^{\prime}}^{\prime}(f(1))
$$

## Private LWE-to-LWE Using Ring Gadget Product.

- Input: $\operatorname{LWE}_{s \vec{k}}(m)=(\vec{a}, b)$
- Switching key: $\operatorname{LtL}_{2} \mathrm{~K}=\left(\operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(\mathbf{s k}), \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(f(1))\right)$, where $\mathbf{s k}=\sum_{i=0}^{l-1}$ $f\left(s k_{i}\right) X^{-i}, \mathbf{s k}^{\prime}=\sum_{i=0}^{l-1} s k_{i}^{\prime} X^{-i}, f: \mathbb{Z} \rightarrow \mathbb{Z}$ is a private R-Lipschitz linear morphism
- Output: $\operatorname{LWE}_{s \vec{k}^{\prime}}(f(m))=\left(\vec{a}^{\prime}, b^{\prime}\right)$
- Algorithm:

$$
\begin{gathered}
\left(\mathbf{a}^{\prime}, \mathbf{b}^{\prime}\right):=\sum_{i=1}^{n} a_{i} X^{i} \odot_{R} \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(\mathbf{s k})+b \odot_{R} \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(f(1)), \\
\operatorname{LtL}_{2}{ }_{s \vec{k} \rightarrow s \vec{k}^{\prime}}^{f}\left(\operatorname{LWE}_{\overrightarrow{s k}}(m)\right):=\left(a_{0}^{\prime}, a_{1}^{\prime}, \ldots, a_{n-1}^{\prime}, b_{0}^{\prime}\right) .
\end{gathered}
$$

| Method | Computation complexity | Key size (in bits) |
| :---: | :---: | :---: |
| PLtL, $\odot$ | $O\left(l n^{2}\right) \mathrm{MM}$ | $l(n+1)^{2} \log q$ |
| $\mathrm{PLtL}, \oplus$ | $(l n+n+l+1) \mathrm{MA}$ | $B l(n+1)^{2} \log q$ |
| $\mathrm{PLtL}_{2}, \odot_{R}$ | $O(l n \log n) \mathrm{MM}$ | $2 l(n+1) \log q$ |

Table 4. Comparison between different versions of the private LWE-to-LWE key switching, where $q$ is the ciphertext modulus, $l$ and $B$ are the gadget length and base, MA and MM denote the modular addition and modular multiplication operations.

Correctness, Error Growth and Comparison. The correctness and error analysis of these algorithms are similar to those in section 4.1. For self completeness, we include them in A.1. The computational complexity and key size of LWE-to-LWE private functional key switching algorithms are listed in tab.4.

A comparison of tab. 3 and tab. 4 reveals that the computational complexity and key size of private algorithms are both larger than the corresponding public algorithms. The comparison results between these three methods are similar to those in sec.4.1: $(\mathrm{PLtL}, \oplus)$ is more suitable for computation-priority scenarios, while $\left(\mathrm{PLtL}_{2}, \odot_{R}\right)$ is more suitable for storage-priority scenarios.

## 5 RLWE-to-RLWE Key Switching

Besides LWE-to-LWE key switching, LWE-to-RLWE and RLWE-to-RLWE key switching are also largely described in the literature. However, LWE-to-RLWE algorithms are highly similar to LWE-to-LWE algorithms. Therefore, we put the whole section in the Appendix A. 3 for readers to refer to the algorithms and theorems. RLWE-to-RLWE key switching is different. To the best of our knowledge, it can only be calculated through ring gadget product.

In this section, we extend this method into functional versions, which support calculation of both public and private Lipschitz functions. We also prove that the widely-used scheme switching algorithm [9] (or EvalSquareMult algorithm [18]) is a special case of our extended private functional key switching algorithm.

### 5.1 Public Functional Key Switching

## RLWE-to-RLWE Using Ring Gadget Product.

$-\operatorname{Input:} \operatorname{RLWE}_{\mathbf{s k}}(\mathbf{m})=(\mathbf{a}, \mathbf{b})$, and a public R-Lipschitz linear morphism $f$ : $\mathcal{R} \rightarrow \mathcal{R}$

- Switching key: $\mathrm{RtRK}=\operatorname{RLWE}_{\mathrm{sk}^{\prime}}^{\prime}(\mathbf{s k})$
- Output: $\operatorname{RLWE}_{\mathbf{s k}^{\prime}}(f(\mathbf{m}))=\left(\mathbf{a}^{\prime}, \mathbf{b}^{\prime}\right)$
- Algorithm:

$$
\operatorname{RtR}_{\mathbf{s k} \rightarrow \mathbf{s k}}{ }^{\prime}\left(\operatorname{RLWE}_{\mathbf{s k}}(\mathbf{m})\right):=f(\mathbf{a}) \odot_{R} \operatorname{RLWE}_{\mathbf{s k}} \mathbf{k}^{\prime}(\mathbf{s k})+(0, f(\mathbf{b}))
$$

## Correctness and error analysis:

Theorem 3. Let $n$ denote the dimension of the ring polynomial of RLWE ciphertexts, $B$ and $l$ denote the base and the length of the gadget decomposition, respectively, then the error variance of the result of the LWE to LWE public functional key switching algorithm is bounded by:

$$
\sigma_{\mathrm{RtR}}^{2} \leq \frac{1}{12} n l B^{2} \sigma_{\mathrm{RtRK}}^{2}+\frac{1}{6} n \epsilon^{2}+\sigma_{\text {input }}^{2}
$$

where $\sigma_{\text {input }}^{2}$ is the error variance of the input $R L W E$ ciphertext, and $\sigma_{\mathrm{RtLR}}^{2}$ is the error variance of the switching key.

Proof. Basing the correctness of the Ring gadget product, we have,

$$
\begin{aligned}
& f(\mathbf{a}) \odot_{R} \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(\mathbf{s k})+(0, f(\mathbf{b})) \\
= & \operatorname{RLWE}_{\mathbf{s k}^{\prime}}(f(\mathbf{a} \cdot \mathbf{s k})+f(\mathbf{b})) \\
= & \operatorname{RLWE}_{\mathbf{s k}^{\prime}}(f(\mathbf{m})+f(\mathbf{e})) .
\end{aligned}
$$

then we measure the error variance based on lemma.2:

$$
\sigma_{\mathrm{RtR}}^{2}=\sigma_{\oplus_{R}, \text { RtRK }}^{2}+\operatorname{Var}(\mathbf{e}) \leq \frac{1}{12} n l B^{2} \sigma_{\mathrm{RtRK}}^{2}+\frac{1}{6} n \epsilon^{2}+R^{2} \sigma_{\text {input }}^{2} .
$$

### 5.2 Private Functional Key Switching

## Private RLWE-to-RLWE Using Ring Gadget Product.

- Input: $^{\operatorname{RLWE}_{\mathbf{s k}}}(\mathbf{m})=(\mathbf{a}, \mathbf{b})$
- Switching key: $\operatorname{RtRK}=\left(\operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(f(\mathbf{s k})), \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(f(1))\right.$, where $f: \mathcal{R} \rightarrow$ $\mathcal{R}$ is a private R -Lipschitz linear morphism
- Output: $\operatorname{RLWE}_{\mathbf{s k}^{\prime}}(f(\mathbf{m}))=\left(\mathbf{a}^{\prime}, \mathbf{b}^{\prime}\right)$
- Algorithm:

$$
\operatorname{RtR}_{\mathbf{s k} \rightarrow \mathbf{s k}}{ }^{\prime}\left(\operatorname{RLWE}_{\mathbf{s k}}(\mathbf{m})\right):=\mathbf{a} \odot_{R} \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(f(\mathbf{s k}))+\mathbf{b} \odot_{R} \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(f(1))
$$

The correctness and error analysis of this algorithm is similar to theorem.3. We put it in A. 2 for self completeness. When the private Lipschitz morphism is $f(x)=$ sk $\cdot x$, our algorithm becomes: $\left.\mathbf{a} \odot_{R} \mathrm{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}\left(\mathbf{s k}^{2}\right)\right)+\mathbf{b} \odot_{R} \mathrm{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(\mathbf{s k})=$ $\left.\mathbf{a} \odot_{R} \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}\left(\mathbf{s k}^{2}\right)\right)+(\mathbf{b}, 0)$, where the right side is the well-known scheme switching algorithm [9] (or EvalSquareMult algorithm [18]).

## 6 Light-Key Bootstrapping

To illustrate the effectiveness of our result, we apply the above analysis to construct light-key bootstrapping algorithms. First, we modify the classical GINX bootstrapping [7], for which our method provides a time-space trade-off. We then improve the LFHE (light-key FHE) bootstrapping proposed by Kim et al. [18], which is specifically designed to reduce the key size. We optimize their result and yield the bootstrapping algorithm with the smallest key.

| Parameters | $Q$ | $Q_{\mathrm{RtR}}$ | $Q_{\mathrm{LtL}}$ | $q$ | $N$ | $N_{\mathrm{RtR}}$ | $n$ | $l_{\mathrm{br}}$ | $l_{\mathrm{ak}}$ | $l_{\mathrm{sqk}}$ | $l_{\mathrm{RtR}}$ | $l_{\mathrm{LtL}}$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| GINX [21] | 25 | - | 14 | 11 | 1024 | - | 571 | 4 | - | - | - | 2 |
| GINX_our | 25 | - | 15 | 11 | 1024 | - | 571 | 4 | - | - | - | 13 |
| LFHE [18] | 54 | 27 | 14 | 11 | 2048 | 1024 | 571 | 3 | 5 | 2 | 2 | 3 |
| LFHE_our | 54 | 27 | 15 | 11 | 2048 | 1024 | 571 | 3 | 5 | 2 | 2 | 13 |

Table 5. Security and Parameters.

### 6.1 Security and Parameters

GINX bootstrapping algorithm use (LtL, $\oplus$ ) for key switching due to its higher efficiency and lower noise growth. However, the large key size of (LtL, $\oplus$ ) results in the key switching key occupying $91.6 \%$ of the GINX bootstrapping key (see tab.1). LFHE replace part of the key switching from $(\mathrm{LtL}, \oplus)$ to $\left(\mathrm{RtR}, \odot_{R}\right)$, which has a smaller key size. However, it still retains an $(L t L, \oplus)$ step, so that the key switching key still occupies $48.3 \%$ of the LFHE bootstrapping key.

In order to construct light-key bootstrapping algorithms, our idea is to use $\left(\mathrm{LtL}_{2}, \odot_{R}\right)$ to replace $(\mathrm{LtL}, \oplus)$ in GINX and LFHE bootstrapping. However, a direct adoption would not work since the noise growth of $\left(\operatorname{LtL}_{2}, \odot_{R}\right)$ is much higher than that of $(L t L, \oplus)$ under the same parameters. Therefore, to ensure algorithm security and control noise introduced by bootstrapping itself, we made necessary adjustments to the bootstrapping parameters, see tab.6.1. This set of parameters ensures that the security level of algorithms exceeds 128 -bit ${ }^{5}$, and the decryption failure rate due to noise accumulation is less than $2^{-32}{ }^{6}$.
$q$ and $n$ denotes the modulus and dimension of the ciphertext before bootstrapping. $Q, N, l_{\mathrm{br}}, l_{\mathrm{ak}}$, and $l_{\text {sqk }}$ are the parameters used for blind rotation, representing the modulus and ring dimension of the blind rotation key, and the gadget length in blind rotation, automorphism, and SquareKeyMult (the latter two are used in LFHE), respectively. $Q_{\mathrm{RtR}}, N_{\mathrm{RtR}}$, and $l_{\mathrm{RtR}}$ denote the modulus, ring dimension, and gadget decomposition length of the RtR key switching key. $Q_{\mathrm{LtL}}$ and $l_{\mathrm{LtL}}$ represent the modulus and gadget decomposition length of the $\mathrm{Lt}_{2}$ key switching key.

### 6.2 Work Flow

The improved GINX and LFHE bootstrapping are shown as follows (all abbreviations are defined in previous section, or check sec.2.4 for explanations):
GINX $_{\text {our }}: \operatorname{LWE}_{571,2^{11}} \xrightarrow{\text { BR }}$ RLWE $_{1024,2^{25}} \xrightarrow{\text { SE }}$ LWE $_{1024,2^{25}} \xrightarrow{\text { MS }}$ LWE $_{1024,2^{15}}$ $\xrightarrow{\mathrm{LtL}_{2}} \mathrm{LWE}_{571,2^{15}} \xrightarrow{\mathrm{MS}^{2}} \mathrm{LWE}_{571,2^{11}}$

LFHE $_{\text {our }}: \mathrm{LWE}_{571,2^{11}} \xrightarrow{\text { BR }}$ RLWE $_{2048,2^{54}} \xrightarrow{\text { MS }}$ RLWE $_{2048,2^{27}} \xrightarrow{\text { RtR }}$ RLWE $_{1024,2^{27}}$ $\xrightarrow{\text { SE }}$ LWE $_{1024,2^{27}} \xrightarrow{\text { MS }}$ LWE $_{1024,2^{15}} \xrightarrow{\text { LtL }_{2}}$ LWE $_{571,2^{15}} \xrightarrow{\text { MS }}$ LWE $_{571,2^{11}}$

[^35]

Fig. 3. The transfer model of TFHE bootstrapping.

### 6.3 Key Size

This section analyze the bootstrapping key size. Since the modulus switching and sample extraction algorithms do not require evaluation keys, the bootstrapping key includes two parts: the blind rotation key and the key switching key.

GINX $_{\text {our }}$ : The blind rotation key contains $n$ RGSW ciphertexts, with each having $4 l_{\mathrm{br}} N \log Q$ bits. This results in a blind rotation key size of 27.88 MB . We use $L t L_{2}$ instead of $L t L_{1}$, the key contains 1 RLWE' ciphertext with a size of $2 n l_{\mathrm{LtL}} \log Q_{\mathrm{LtL}}$ bits. Thus the key switching key size is 27.2 KB . The total key size is 27.91 MB .

LFHE $_{\text {our }}$ : The blind rotation key also contains $n$ RGSW ciphertexts, resulting in a blind rotation key size of 90.33 MB . The RtR key switching key contains 1 RLWE ${ }^{\prime}$ ciphertext with a size of $2 N l_{\text {RtR }} \log Q_{\text {RtR }}$ bits, resulting in a key size of 27 KB . The $\mathrm{LtL}_{2}$ key switching key size is 27.2 KB . The total is 90.38 MB .

### 6.4 Transfer Model and Transfer Key Size

LFHE [18] proposed a transfer model, see fig.3. The client transmits a transfer key (a seed) to the server. Then the server runs the reconstruction algorithm to obtain the complete bootstrapping key, and performs the bootstrapping algorithm. In this model, client and server utilize a common reference string (CRS) to generate the a-components of each transferred LWE and RLWE ciphertext. Thus only the $b$ (or $\mathbf{b}$ for RLWE)-components of the ciphertext needs to be transmitted. LFHE's blind rotation algorithm is specifically designed for the transfer model and uses a pached blind rotation key to reduce the bootstrapping transfer key size to within 1 MB . We also calculate the transfer key size of our improved algorithms under the transfer model.

GINX $_{\text {our }}$ : The blind rotation key contains $n$ RGSW ciphertexts, with each needing to transfer $2 l_{\mathrm{br}} N \log Q$ bits. This results in a blind rotation transfer key

| Methods | Transfer key size | Bootstrapping key size |
| :---: | :---: | :---: |
| GINX | 16.48 MB | 250 MB |
| GINX $_{\text {our }}$ | 13.96 MB | 27.91 MB |
| LFHE | 881 KB | 175 MB |
| LFHE $_{\text {our }}$ | 810.1 KB | 90.38 MB |

Table 6. Transfer key size and the bootstrapping key size in different methods.
size of 13.94 MB . We use $\mathrm{Lt}_{2}$ instead of $\mathrm{Lt}_{1}$, the key contains $1 \mathrm{RLWE}^{\prime}$ ciphertext, with a transfer key size of $n l_{\mathrm{LtL}} \log Q_{\mathrm{LtL}}$ bits. This results in a key switching transfer key size of 13.6 KB . The total is 13.96 MB .

LFHE $_{\text {our }}$ : The packed blind rotation key contains 1 RLWE ciphertext and $(\log N+1)$ RLWE $^{\prime}$ ciphertexts. Each RLWE ciphertext needs to transfer $N \log Q$ bits, each RLWE' ciphertext needs to transfer $N l_{\text {ak }}\left(l_{\text {sqk }}\right) \log Q$ bits. This results in a blind rotation transfer key size of 783 KB . The RtR key switching key contains 1 RLWE' ciphertext and has a transfer size of $N l_{\text {RtR }} \log Q_{\text {RtR }}$ bits, resulting in a transfer key size of 13.5 KB . The $\mathrm{LtL}_{2}$ transfer key size is 13.6 KB . The total key size is 810.1 KB .

For GINX bootstrapping, our method reduces the bootstrapping key size by $88.8 \%$ and the transfer key size by $15.3 \%$. We do not want to oversell this result, but take it as a trade-off method towards practical TFHE applications. For LFHE bootstrapping, our method outperforms Kim's method [18] by reducing $48.4 \%$ bootstrapping key size and $8 \%$ transfer key size.

## 7 Conclusion

The key switching algorithm is crucial in real-world fully homomorphic encryption (FHE) applications due to its significant impact on the key size and efficiency of the FHE system. This paper revisits currently known key switching algorithms, expands their functionality, carefully recalculates the error growth, and provide a comparison of different algorithms under the same benchmark. Our analysis is applied to the bootstrapping algorithm, resulting in optimal light-key FHE. This paper can be served as an reference for the time-space trade-off of key switching algorithms and assists to build FHE applications with different computational and storage requirements.

## Acknowledgments

We are grateful for the helpful comments from the anonymous reviewers of ICISC 2023. This work was supported by CAS Project for Young Scientists in Basic Research (Grant No. YSBR-035).

## References

1. Amuthan, A., Sendhil, R.: Hybrid gsw and dm based fully homomorphic encryption scheme for handling false data injection attacks under privacy preserving data aggregation in fog computing. Journal of Ambient Intelligence and Humanized Computing 11, 5217-5231 (2020)
2. Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., Sirdey, R.: Stream ciphers: A practical solution for efficient homomorphicciphertext compression. Journal of Cryptology 31(3), 885-916 (2018)
3. Chen, H., Dai, W., Kim, M., Song, Y.: Efficient homomorphic conversion between (ring) lwe ciphertexts. In: International Conference on Applied Cryptography and Network Security. pp. 460-479. Springer (2021)
4. Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: Bootstrapping for approximate homomorphic encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 360-384. Springer (2018)
5. Chillotti, I., Gama, N., Georgieva, M., Izabachene, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. ADVANCES IN CRYPTOLOGY-ASIACRYPT 2016, PT I 10031, 3-33 (2016)
6. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster packed homomorphic operations and efficient circuit bootstrapping for tfhe. In: Advances in CryptologyASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I. pp. 377-408. Springer (2017)
7. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Tfhe: fast fully homomorphic encryption over the torus. Journal of Cryptology 33(1), 34-91 (2020)
8. Cosseron, O., Hoffmann, C., Méaux, P., Standaert, F.X.: Towards case-optimized hybrid homomorphic encryption: Featuring the elisabeth stream cipher. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 32-67. Springer (2022)
9. De Micheli, G., Kim, D., Micciancio, D., Suhl, A.: Faster amortized fhew bootstrapping using ring automorphisms. Cryptology ePrint Archive (2023)
10. Deviani, R.: The application of fully homomorphic encryption on xgboost based multiclass classification. JIEET (Journal of Information Engineering and Educational Technology) 7(1), 49-58 (2023)
11. Dobraunig, C., Eichlseder, M., Grassi, L., Lallemand, V., Leander, G., List, E., Mendel, F., Rechberger, C.: Rasta: a cipher with low anddepth and few ands per bit. In: Advances in Cryptology-CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I 38. pp. 662-692. Springer (2018)
12. Ducas, L., Micciancio, D.: Fhew: bootstrapping homomorphic encryption in less than a second. In: Advances in Cryptology-EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I 34. pp. 617-640. Springer (2015)
13. Gomes, F.A., de Matos, F., Rego, P., Trinta, F.: Analysis of the impact of homomorphic algorithm on offloading of mobile application tasks. In: 2023 IEEE 20th Consumer Communications \& Networking Conference (CCNC). pp. 961-962. IEEE (2023)
14. Halevi, S., Shoup, V.: Bootstrapping for helib. Journal of Cryptology 34(1), 7 (2021)
15. Jiang, L., Lou, Q., Joshi, N.: Matcha: A fast and energy-efficient accelerator for fully homomorphic encryption over the torus. In: Proceedings of the 59th ACM/IEEE Design Automation Conference. pp. 235-240 (2022)
16. Jutla, C.S., Manohar, N.: Modular lagrange interpolation of the mod function for bootstrapping of approximate he. Cryptology ePrint Archive (2020)
17. Jutla, C.S., Manohar, N.: Sine series approximation of the mod function for bootstrapping of approximate he. Springer-Verlag (2022)
18. Kim, A., Lee, Y., Deryabin, M., Eom, J., Choi, R.: Lfhe: Fully homomorphic encryption with bootstrapping key size less than a megabyte. Cryptology ePrint Archive (2023)
19. Kocabas, O., Soyata, T.: Towards privacy-preserving medical cloud computing using homomorphic encryption. In: Virtual and Mobile Healthcare: Breakthroughs in Research and Practice, pp. 93-125. IGI Global (2020)
20. Lee, J.W., Kang, H., Lee, Y., Choi, W., Eom, J., Deryabin, M., Lee, E., Lee, J., Yoo, D., Kim, Y.S., et al.: Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. IEEE Access 10, 30039-30054 (2022)
21. Lee, Y., Micciancio, D., Kim, A., Choi, R., Deryabin, M., Eom, J., Yoo, D.: Efficient fhew bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 227-256. Springer (2023)
22. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 1-23. Springer, Heidelberg (2010)
23. Micciancio, D., Polyakov, Y.: Bootstrapping in fhew-like cryptosystems. In: Proceedings of the 9th on Workshop on Encrypted Computing \& Applied Homomorphic Cryptography. pp. 17-28 (2021)
24. Naehrig, M., Lauter, K., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Proceedings of the 3rd ACM workshop on Cloud computing security workshop. pp. 113-124 (2011)
25. Nam, K., Oh, H., Moon, H., Paek, Y.: Accelerating n-bit operations over tfhe on commodity cpu-fpga. In: Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design. pp. 1-9 (2022)
26. Peralta, G., Cid-Fuentes, R.G., Bilbao, J., Crespo, P.M.: Homomorphic encryption and network coding in iot architectures: Advantages and future challenges. Electronics 8(8), 827 (2019)
27. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM 56(6), 34:1-34:40 (2009)
28. Ren, W., Tong, X., Du, J., Wang, N., Li, S.C., Min, G., Zhao, Z., Bashir, A.K.: Privacy-preserving using homomorphic encryption in mobile iot systems. Computer Communications 165, 105-111 (2021)
29. Shrestha, R., Kim, S.: Integration of iot with blockchain and homomorphic encryption: Challenging issues and opportunities. In: Advances in computers, vol. 115, pp. 293-331. Elsevier (2019)
30. Ye, T., Kannan, R., Prasanna, V.K.: Fpga acceleration of fully homomorphic encryption over the torus. In: 2022 IEEE High Performance Extreme Computing Conference (HPEC). pp. 1-7. IEEE (2022)

## A Appendix

## A. 1 LWE-to-LWE Key Switching

## Private LWE-to-LWE Using Canonical Gadget Product

Theorem 4. Let $n$ denote the dimension of the LWE ciphertexts, $B$ and $l$ denote the base and the length of the gadget decomposition, respectively, then the error variance of the result of the LWE to LWE public functional key switching algorithm is bounded by:

$$
\sigma_{\text {PLtL }}^{2} \leq \frac{1}{12}(n+1) l B^{2} \sigma_{\text {LtLK }}^{2}+\frac{1}{6} R^{2} n \epsilon^{2}++\frac{1}{3} n \operatorname{Var}(f(1)) \epsilon^{2}+R^{2} \sigma_{\text {input }}^{2}
$$

where $\sigma_{\text {input }}^{2}$ is the error variance of the input LWE ciphertext, and $\sigma_{\text {LtLK }}^{2}$ is the error variance of the switching key.

Proof. Basing the correctness of the gadget product, we have,

$$
\begin{aligned}
& \sum_{i=1}^{n} a_{i} \odot \operatorname{LWE}_{\mathbf{s k}^{\prime}}^{\prime}\left(f\left(s k_{i}\right)\right)+b \odot \mathrm{LWE}_{\mathbf{s k}^{\prime}}^{\prime}(f(1)) \\
= & \operatorname{LWE}_{\mathbf{s k}^{\prime}}\left(\sum_{i=1}^{n} f\left(a_{i} \cdot s k_{i}\right)+f(b)\right) \\
= & \operatorname{LWE}_{\mathbf{s k}^{\prime}}(f(m)+f(e))
\end{aligned}
$$

then we measure the error variance based on lemma.1:

$$
\begin{aligned}
\sigma_{\mathrm{PLtL}}^{2} & =\sum_{i=1}^{n} \sigma_{\odot, \mathrm{LWE}_{\mathrm{sk}^{\prime}}^{\prime}\left(f\left(s k_{i}\right)\right)}^{2}+\sigma_{\odot, \mathrm{LWE}_{\mathrm{sk}^{\prime}}^{\prime}(f(1))}^{2}+\operatorname{Var}(f(e)) \\
& \leq \frac{1}{12}(n+1) l B^{2} \sigma_{\mathrm{LtLK}}^{2}+\frac{1}{3} n \operatorname{Var}\left(f\left(s k_{i}\right) \epsilon^{2}+\frac{1}{3} n \operatorname{Var}(f(1)) \epsilon^{2}+R^{2} \sigma_{\text {input }}^{2} .\right. \\
& \leq \frac{1}{12}(n+1) l B^{2} \sigma_{\mathrm{LtLK}}^{2}+\frac{1}{6} R^{2} n \epsilon^{2}++\frac{1}{3} n \operatorname{Var}(f(1)) \epsilon^{2}+R^{2} \sigma_{\text {input }}^{2} .
\end{aligned}
$$

## Private LWE-to-LWE Using Ring Gadget Product

Theorem 5. Let $n$ denote the dimension of the LWE ciphertexts, $B$ and $l$ denote the base and the length of the gadget decomposition, respectively, then the error variance of the result of the LWE to LWE using RtR algorithm is bounded by:

$$
\sigma_{\mathrm{PLtL}_{2}}^{2} \leq \frac{1}{6} N l B^{2} \sigma_{\mathrm{LtL}_{2} \mathrm{~K}}^{2}+\frac{1}{6} R^{2} n \epsilon^{2}+\frac{1}{3} n \operatorname{Var}(f(1)) \epsilon^{2}+R^{2} \sigma_{\text {input }}^{2}
$$

where $\sigma_{\text {input }}^{2}$ is the error variance of the input LWE ciphertext, and $\sigma_{\mathrm{PLtL} 2_{2} \mathrm{~K}}^{2}$ is the error variance of the switching key.

Proof. Basing the correctness of the ring gadget product, we have,

$$
b_{0}^{\prime}+\sum_{i=1}^{n} a_{i}^{\prime} s k_{i}^{\prime}=\left[\mathbf{b}^{\prime}+\mathbf{a}^{\prime} \cdot \mathbf{s k}\right]_{0}=\sum_{i=1}^{n} f\left(a_{i} \cdot s k_{i}\right)+f(b)=f(m)+f(e),
$$

thus $\left(a_{0}^{\prime}, a_{1}^{\prime}, \ldots, a_{n-1}^{\prime}, b_{0}^{\prime}\right)$ is the LWE ciphertext of $f(m)$ under secret key $\overrightarrow{s k}^{\prime}$, then we measure the error variance based on corollary.2:

$$
\begin{aligned}
\sigma_{\mathrm{LtL}_{2}}^{2} & =\sigma_{\odot_{R}, \mathrm{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(\mathbf{s k})}^{2}+\sigma_{\odot_{R}, \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(f(\mathbf{1}))}^{2}+\operatorname{Var}(f(\mathbf{e})) \\
& \leq \frac{1}{6} N l B^{2} \sigma_{\mathrm{LtL}_{2} \mathrm{~K}}^{2}+\frac{1}{3} n \operatorname{Var}\left(f\left(s k_{i}\right) \epsilon^{2}+\frac{1}{3} n \operatorname{Var}(f(1)) \epsilon^{2}+R^{2} \sigma_{\text {input }}^{2}\right. \\
& \leq \frac{1}{6} N l B^{2} \sigma_{\mathrm{LtL}_{2} \mathrm{~K}}^{2}+\frac{1}{6} R^{2} n \epsilon^{2}+\frac{1}{3} n \operatorname{Var}(f(1)) \epsilon^{2}+R^{2} \sigma_{\text {input }}^{2}
\end{aligned}
$$

## A. 2 RLWE-to-RLWE Key Switching

## Private RLWE-to-RLWE

Theorem 6. Let $n$ denote the dimension of the ring polynomial of RLWE ciphertexts, $B$ and $l$ denote the base and the length of the gadget decomposition, respectively, then the error variance of the result of the LWE to LWE public functional key switching algorithm is bounded by:

$$
\sigma_{\mathrm{RtR}}^{2} \leq \frac{1}{12} n l B^{2} \sigma_{\mathrm{RtRK}}^{2}+\frac{1}{6} n \epsilon^{2}+\sigma_{\text {input }}^{2}
$$

where $\sigma_{\text {input }}^{2}$ is the error variance of the input RLWE ciphertext, and $\sigma_{\mathrm{RtLR}}^{2}$ is the error variance of the switching key.

Proof. Basing the correctness of the Ring gadget product, we have,

$$
\begin{aligned}
& \mathbf{a} \odot_{R} \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(f(\mathbf{s k}))+\mathbf{b} \odot_{R} \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(f(1)) \\
= & \operatorname{RLWE}_{\mathbf{s k}^{\prime}}(f(\mathbf{a} \cdot \mathbf{s k})+f(\mathbf{b})) \\
= & \operatorname{RLWE}_{\mathbf{s k}^{\prime}}(f(\mathbf{m})+f(\mathbf{e})) .
\end{aligned}
$$

then we measure the error variance based on lemma.2:

$$
\begin{aligned}
\sigma_{\mathrm{RtR}}^{2} & =\sigma_{\odot_{R}, \operatorname{RLWE}_{\mathbf{s k}}}^{\prime}(f(\mathbf{s k}))+\sigma_{\odot_{R}, \mathrm{RLWE}_{\mathbf{s k}}}^{2}(f(\mathbf{1})) \\
& \leq \frac{1}{6} n l B^{2} \sigma_{\mathrm{RtRK}}^{2}+\frac{1}{3} n \operatorname{Var}(f(\mathbf{e})) \\
& \leq \frac{1}{6} n l B^{2} \sigma_{\mathrm{RtRK}}^{2}+\frac{1}{6} R^{2} n \epsilon^{2}+\frac{1}{3} n \operatorname{Var}(f(1)) \epsilon^{2}+\frac{1}{3} n \operatorname{Var}(f(1)) \epsilon^{2}+R^{2} \sigma_{\text {input }}^{2} \sigma_{\text {input }}^{2}
\end{aligned}
$$

## A. 3 LWE-to-RLWE Key Switching

## Public LWE-to-RLWE Functional Key Switching

Input: $\operatorname{LWE}_{\text {sk }}(m)=(\mathbf{a}, b)$, and a public R-Lipschitz morphism $f: \mathbb{Z} \rightarrow \mathbb{Z}$
Switching key: $\operatorname{LtRK}=\operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}\left(s k_{i}\right)_{i \in[1, n]}$
Output: $\operatorname{RLWE}_{\mathbf{s k}^{\prime}}(f(m))=\left(\mathbf{a}^{\prime}, \mathbf{b}^{\prime}\right)$
Algorithm:

$$
\operatorname{LtR}_{\mathbf{s k} \rightarrow \mathbf{s k}^{\prime}}^{f}\left(\operatorname{LWE}_{\mathbf{s k}}(m)\right):=\sum_{i=1}^{n} f\left(a_{i}\right) \odot \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}\left(s k_{i}\right)+(0, f(b))
$$

## Correctness and error analysis:

Theorem 7. Let $n$ denote the dimension of the LWE ciphertexts, $B$ and $l$ denote the base and the length of the gadget decomposition, respectively, then the error variance of the result of the LWE to LWE public functional key switching algorithm is bounded by:

$$
\sigma_{\mathrm{LtR}}^{2} \leq \frac{1}{12} n l B^{2} \sigma_{\mathrm{LtRK}}^{2}+\frac{1}{6} n \epsilon^{2}+R^{2} \sigma_{\text {input }}^{2}
$$

where $\sigma_{\text {input }}^{2}$ is the error variance of the input LWE ciphertext, and $\sigma_{\text {LtLK }}^{2}$ is the error variance of the switching key.

Proof. Basing the correctness of the gadget product, we have,

$$
\begin{aligned}
& \sum_{i=1}^{n} f\left(a_{i}\right) \odot \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}\left(s k_{i}\right)+(0, f(b)) \\
= & \operatorname{RLWE}_{\mathbf{s k}^{\prime}}\left(\sum_{i=1}^{n} f\left(a_{i} \cdot s k_{i}\right)+f(b)\right) \\
= & \operatorname{RLWE}_{\mathbf{s k}^{\prime}}(f(m)+f(e)),
\end{aligned}
$$

then we measure the error variance based on lemma.1:

$$
\sigma_{\mathrm{LtR}}^{2}=n \sigma_{\odot, \mathrm{LtRK}}^{2}+\operatorname{Var}(f(e)) \quad \leq \frac{1}{12} n l B^{2} \sigma_{\mathrm{LtLK}}^{2}+\frac{1}{6} n \epsilon^{2}+R^{2} \sigma_{\text {input }}^{2}
$$

## Private LWE-to-RLWE Functional Key Switching

Input: $\operatorname{LWE}_{\mathbf{s k}}(m)=(\mathbf{a}, b)$
Switching key: $\operatorname{PLtRK}=\left(\operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}\left(f\left(s k_{i}\right)\right)_{i \in[1, n]}, \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(f(1))\right)$, where $f$ : $\mathbb{Z} \rightarrow \mathbb{Z}$ is a private R-Lipschitz linear morphism
Output: $\operatorname{RLWE}_{\mathbf{s k}^{\prime}}(f(m))=\left(\mathbf{a}^{\prime}, \mathbf{b}^{\prime}\right)$
Algorithm:

$$
\operatorname{PLtR}_{\mathbf{s k} \rightarrow \mathbf{s k}^{\prime}}^{f}\left(\operatorname{LWE}_{\mathbf{s k}}(m)\right):=\sum_{i=1}^{n} a_{i} \odot \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}\left(f\left(s k_{i}\right)\right)+b \odot \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(f(1))
$$

Correctness and error analysis:

24 Ruida Wang et al.
Theorem 8. Let $n$ denote the dimension of the LWE ciphertexts, $B$ and $l$ denote the base and the length of the gadget decomposition, respectively, then the error variance of the result of the LWE to LWE public functional key switching algorithm is bounded by:

$$
\sigma_{\mathrm{PLtR}}^{2} \leq \frac{1}{12}(n+1) l B^{2} \sigma_{\mathrm{LtLK}}^{2}+\frac{1}{6} R^{2}(n+1) \epsilon^{2}+R^{2} \sigma_{\text {input }}^{2}
$$

where $\sigma_{\text {input }}^{2}$ is the error variance of the input LWE ciphertext, and $\sigma_{\text {LtLK }}^{2}$ is the error variance of the switching key.

Proof. Basing the correctness of the gadget product, we have,

$$
\begin{aligned}
& \sum_{i=1}^{n} a_{i} \odot \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}\left(f\left(s k_{i}\right)\right)+b \odot \operatorname{RLWE}_{\mathbf{s k}^{\prime}}^{\prime}(f(1)) \\
= & \operatorname{RLWE}_{\mathbf{s k}^{\prime}}\left(\sum_{i=1}^{n} f\left(a_{i} \cdot s k_{i}\right)+f(b)\right) \\
= & \operatorname{RLWE}_{\mathbf{s k}^{\prime}}(f(m)+f(e)),
\end{aligned}
$$

then we measure the error variance based on lemma.1:

$$
\begin{aligned}
\sigma_{\mathrm{PLtR}}^{2} & =(n+1) \sigma_{\odot, \mathrm{PLtRK}}^{2}+\operatorname{Var}(f(e)) \\
& \leq \frac{1}{12}(n+1) l B^{2} \sigma_{\mathrm{LtLK}}^{2}+\frac{1}{6} R^{2}(n+1) \epsilon^{2}+R^{2} \sigma_{\text {input }}^{2}
\end{aligned}
$$

## ICISC 2023


[^0]:    * This work was supported by Institute for Information \& communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2017-0-00520, Development of SCR-Friendly Symmetric Key Cryptosystem and Its Application Modes).

[^1]:    ${ }^{1}$ The full depth is naturally reduced thanks to the reduction in the Toffoli depth.

[^2]:    ${ }^{1}$ https://www.ibm.com/quantum/roadmap

[^3]:    ${ }^{2}$ Detailed estimation of Grover's search while varying the parameters of the NV sieve remains for our future work.
    ${ }^{3}$ https://en.wikipedia.org/wiki/Unimodular_matrix

[^4]:    ${ }^{4}$ https://qiskit.org/

[^5]:    ${ }^{1}$ We ran the experiment using the BKZ implementation from fpylll in Sage9.2. See https://github.com/fplll/fpylll

[^6]:    * This work was supported by the Sungshin Women's University Research Grant of H20210012.
    ** Corresponding author: Department of Future Convergence Technology Engineering, Sungshin Women's University, Seoul, Korea, Tel: +82-2-920-7449

[^7]:    ${ }^{1}$ Note that ECALLs are used for to enter the enclave and OCALLs are used to switch an execution flow to untrusted region, respectively.

[^8]:    ${ }^{4}$ The exact location of the implementation is 'AOSP $\backslash$ system $\backslash \mathrm{bt} \backslash \mathrm{hci}$ \src $\backslash$ hci_layer_ android.cc:hci_initialize().'

[^9]:    * Corresponding Author
    ${ }^{1}$ All of the formal models and lemmas are open to the public through the following url https://github.com/HackProof/mdTLS

[^10]:    * Equal contribution.
    ** Corresponding author.

[^11]:    * Corresponding author: chentianyu@iie.ac.cn.

[^12]:    ${ }^{5}$ Strictly speaking, the Equation (1) is built on composite-order groups. A general approach to transforming schemes over composite-order groups into ones over primeorder groups has been proposed in [5]. Thus, in this section, we decide to abuse constructions over composite-order groups as ones over prime-order groups for simplicity.

[^13]:    ${ }^{4}$ In our protocol, players commit the secrets at the beginning of the protocol by using a cryptographic hash function. Thus, more precisely, we need to extend the bit lengths of secrets to an appropriate length by adding multiples of $v_{a}+v_{b}$.

[^14]:    ${ }^{5}$ Only the Win transaction corresponding to the winner of the final match uses the template for the root node. See Fig. 3, and $\operatorname{Win}\left(\pi_{r}, a\right)$ is the corresponding template.

[^15]:    ${ }^{5}$ The reason utilizing eight cards in [27] comes from this point.

[^16]:    * This work was supported by JST CREST Grant Number JPMJCR2113 and JSPS KAKENHI Grant Number JP23K16841.

[^17]:    ${ }^{1}$ With the following modification, our scheme also supports the case where $n>1$ is not a power of 2 . Let $k$ be an integer such that $2^{k-1}<n<2^{k}$. We change a list of message $M=\left(m_{0}, \ldots m_{n-1}\right)$ which is given to $\mathrm{OS} . \mathrm{U}_{1}$ and $\mathrm{OS} . \mathrm{S}_{2}$ as a part of an input to an augmented message list $M^{\prime}=\left(m_{0}^{\prime}, \ldots m_{2^{k}-1}^{\prime}\right)$ where $m_{i}^{\prime}=m_{i}$ for $i \in\{0, \ldots, n-1\}, m_{n-1+i}^{\prime}=\phi \| i$ for $i \in\left\{1, \ldots 2^{k}-n\right\}$, and $\phi$ is a special symbol representing that a message is empty.

[^18]:    ${ }^{4}$ To be precise, S has to change the way to retrieve ciphertexts depending on $\mathrm{SP}_{q}^{(t)}$; S first retrieves ciphertexts re-added at the last search for $q$, i.e., at $t^{\prime}=\max \mathrm{SP}_{q}^{(t)}$, and then retrieves ciphertexts simulated from $t^{\prime}$ to $t$.

[^19]:    ${ }^{5}$ We did not implement sOurs since we want to compare dynamic SSE schemes with the same security level. Note that $s$-Laura is secure even if deleted entries are readded.

[^20]:    * This work was supported by Institute of Mathematics for Industry, Joint Usage/Research Center in Kyushu University. (FY2022 Workshop(II) "CRISMATH2022" (2022c006).).

[^21]:    ${ }^{3}$ Very recently, Yan et al. proposed a new quantum factoring algorithm which requires a fewer number of qubits [21] and gave a new estimation for factoring 2048-bit integers. However, the validity of the algorithm and the correctness of the estimation are under the analysis.

[^22]:    * This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the Convergence Security Core Talent Training Business (Pusan National University), support program (IITP-2023-2022-0-01201) supervised by the IITP (Institute for Information \& Communications Technology Planning \& Evaluation, $50 \%$ ) and by the Institute for Information \& Communications Technology Planning \& Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00033, Study on Quantum Security Evaluation of Cryptography based on Computational Quantum Complexity, 50\%)

[^23]:    ${ }^{1}$ Number Theorists 'R' Us, or Number Theory Research Unit, or N-th degree TRuncated polynomial Ring.

[^24]:    ${ }^{2}$ This corresponds to NIST-I and NIST-V requirements.

[^25]:    ${ }^{3}$ The same identity can also be used to check the validity of signatures only with a hash of the public key $h$, requiring this time send both $\mathbf{s}_{1}$ and $\mathbf{s}_{2}$, but we will not consider this setting here.

[^26]:    * corresponding author

[^27]:    Author list in alphabetical order; see https://www.ams.org/profession/leaders/ culture/CultureStatement04.pdf. This work was funded in part by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy-EXC 2092 CASA-390781972"Cyber Security in the Age of Large-Scale Adversaries" and by the Netherlands Organisation for Scientific Research (NWO) under grants OCENW.KLEIN. 539 and VI.Vidi.203.045. Date: 2023.11.14. For the full version see [LPR23].

[^28]:    ${ }^{1}$ https://www.kpqc.or.kr/images/zip/REDOG.zip

[^29]:    ${ }^{4}$ A modern, compilable re-host of the Espresso heuristic logic minimizer can be found at https://github.com/classabbyamp/espresso-logic.

[^30]:    ${ }^{3}$ For readability we do not make input and key dependence explicit in the leakage $L$.
    ${ }^{4}$ Side-channel attacks are also possible by exploiting the output with $f_{k^{*}}^{-1}$.

[^31]:    ${ }^{5}$ It is worth noting that there exists no known optimal multivariate implementation for the above mentioned side-channel distinguishers [ $\left.\mathrm{BGP}^{+} 11, \mathrm{WOM} 11\right]$, because the

[^32]:    ${ }^{6}$ We refrain to include more details at this point in order to maintain the anonymity of the submission.
    ${ }^{7}$ Spearman and DoM are excluded from Figure 4c as they failed against the masked implementation.

[^33]:    ${ }^{3}$ Many reasons can cause the mismatch in access order. The filter effect itself is a potential cause as soon shown in Fig. 3b. The imperfect pseudo-LRU used in hardware and the RRIP derivative policies used in L2 and LLC [31] also cause mismatching replacement order and access order. Finally, the L2 in this case (also in modern Intel processor) is exclusive, whose replacement order is also affected by the block swapping between L2 and L1 when a block hits in L2.

[^34]:    ${ }^{3}$ The gadget product is the computational units for scalar multiplication in FHEWlike cryptosystems
    ${ }^{4}$ The gadget decomposition is a technique used to decompose large numbers into smaller digits. This helps control error growth in FHE algorithms.

[^35]:    ${ }^{5}$ test by LWE estimator, https://bitbucket.org/malb/lwe-estimator/src/master/
    ${ }^{6}$ calculate by $1-\operatorname{erf}\left(\frac{q}{8 \sqrt{2} \sigma}\right)$, where erf represents the Gaussian error function.

