Delegating Supersingular Isogenies over $\mathbb{F}_{p^2}$ with Cryptographic Applications

Robi Pedersen[1][0000−0001−5120−5709] and Osmanbey Uzunkol[2][0000−0002−5151−3848]

[1] imec-COSIC KU Leuven, Kasteelpark Arenberg 10 Bus 2452, 3001 Leuven, Belgium
robi.pedersen@esat.kuleuven.be (corresponding author)
[2] Information und Kommunikation, Flensburg University of Applied Sciences, Flensburg, Germany
osmanbey.uzunkol@gmail.com

Abstract. Although isogeny-based cryptographic schemes enjoy the smallest key sizes amongst current post-quantum cryptographic candidates, they come at a high computational cost, making their deployment on the ever-growing number of resource-constrained devices difficult. Speeding up the expensive post-quantum cryptographic operations by delegating these computations from a weaker client to untrusted powerful external servers is a promising approach. Following this, we present in this work mechanisms allowing computationally restricted devices to securely and verifiably delegate isogeny computations to potentially untrusted third parties. In particular, we propose two algorithms that can be integrated into existing isogeny-based protocols and which lead to a much lower cost for the delegator than the full, local computation. For example, compared to the local computation cost, we reduce the public-key computation step of SIDH/SIKE by a factor 5 and zero-knowledge proofs of identity by a factor 16 for the prover, while it becomes almost free for the verifier, respectively, at the NIST security level 1.

Keywords: Isogeny-based cryptography · Post-quantum cryptography · Secure computation outsourcing · Lightweight cryptography

1 Introduction

Delegation of Cryptographic Primitives. In recent years, the number of interconnected devices using new computational paradigms such as cloud, edge and mobile computing, and their interactions with the industrial internet of things, big data and artificial intelligence, are steadily increasing in numbers. As a result, delegating expensive computations from clients such as RFID-cards and low power sensors with constrained resources or capabilities to powerful external resources has become a highly active and an indispensable research and development area for researchers and industry alike. Delegation of sensitive computation to *potentially malicious* external devices and services, however, comes with some additional challenges, such as requiring security of the clients' inputs/outputs

as well as verifiability of the outputs coming from these external devices and services. A particular case of interest is the delegation of cryptographic algorithms and protocols. The security and verifiability properties of cryptographic delegations were first formalized in a security model introduced by Hohenberger and Lysyanskaya [25], introduced in the context of modular exponentiations. In this model, a weak, trusted client $\mathcal{T}$ makes queries to a set of untrusted external servers $\mathcal{U}$ in such a way that their interaction $\mathcal{T}^{\mathcal{U}}$ realizes a computational task Alg in a joint manner. The goal is to reduce the computational cost of $\mathcal{T}$ while guaranteeing the security of its inputs and outputs, and the possibility of verifying the correctness of the outputs of $\mathcal{U}$.

Isogenies and Cryptography. Many currently deployed public-key cryptographic primitives are based on the infeasibility of either the factorization or discrete logarithm problems. Possible efficient implementations of Shor's algorithm [38] on large scale quantum computers could render these schemes insecure against such quantum adversaries. This threat resulted in the United States' National Institute of Standards and Technology (NIST) launching a post-quantum cryptography standardization process at the end of 2017. Of the 69 initially proposed key-establishment and signature protocols, a list of 15 main and alternate candidates (9 encryption and KEMs, 6 digital signature schemes) have progressed to the third round of scrutiny, announced in July 2020 [33].

One of these alternate candidates is the key encapsulation scheme SIKE [39] which is based on the Supersingular Isogeny Diffie-Hellman (SIDH) key exchange protocol, originally proposed by Jao and De Feo [27]. SIDH is a quantum resistant key agreement scheme, which uses isogenies between supersingular elliptic curves over finite fields $\mathbb{F}_{p^2}$. Besides the key agreement scheme in [27] and SIKE [39], several other cryptographic schemes based on the supersingular elliptic curves have been recently proposed in the literature ranging from group key agreement schemes [3,22], zero-knowledge proofs of identity [27,17], identification and signature schemes [23] and hash functions [10,21] to verifiable delay functions [20].

Motivation. A significant advantage of isogeny-based cryptographic schemes are the small key sizes when compared to their lattice- or code-based post-quantum counterparts. However, the main drawback is performance: SIKE is about an order of magnitude slower than its NIST competitors [1,7]. Furthermore, as pointed out in [32], post-quantum cryptographic schemes are especially required to also work efficiently on resource-constrained devices with highly limited processing storage, power and battery life to be able to utilize them in lightweight environments, which is highly desired for various applications requiring certain interoperability properties. We address this problem and study the secure and verifiable delegation of isogeny computations between supersingular elliptic curves over $\mathbb{F}_{p^2}$ in order to reduce the computational cost of resource-constrained clients requiring to utilize different isogeny-based cryptographic schemes.

Previous Work. In [34], two isogeny delegation algorithms in the *honest-but-curious* (HBC) and *one-malicious version of a two-untrusted program* (OMTUP)

assumptions were proposed using the security model of Hohenberger and Lysyanskaya [25]. The first, ScIso, allowed to delegate the computation of any isogeny with revealed kernel, while allowing to push through hidden elliptic curve points or multiply unprotected points with hidden scalars. Random torsion point generation was done using lookup-tables of the form $\{(i, \ell^i P)\}_{i\in\{1,\ldots,e-1\}}$, $\{(i, \ell^i Q)\}_{i\in\{1,\ldots,e-1\}}$ for generators $\langle P, Q\rangle \in E[\ell^e]$. The second algorithm, HIso, used ScIso as a subroutine and allowed to hide the kernel and the codomain of the delegated isogeny. The work of [34] did not propose a protocol to delegate public-key computations.

Our Contributions. The main contribution of this paper is to propose two new delegation algorithms for isogeny computations using the security model of Hohenberger and Lysyanskaya [25] in the HBC and OMTUP models, and to show how to apply these to different isogeny-based cryptographic protocols and computing the respective gains for the delegator. In particular,

1. We show how to break the HIso subroutine of [34] using pairings, and discuss some new approaches to hide the codomain curve in delegation algorithms.
2. We introduce the delegation algorithm Iso, which allows to delegate isogeny computations with unprotected kernel and to push through public and hidden points. Iso does not require lookup-tables, eliminating the large local memory requirement of the ScIso-algorithm from [34] on the delegator's side, while also speeding up the delegation algorithms.
3. The second algorithm, IsoDetour, uses Iso as a subroutine and allows to delegate the computation of an isogeny without revealing the kernel. This allows the computation of public keys, a question left open in [34]. The security of IsoDetour is based on a difficulty assumption implicitly used in the identification protocol of [27], which we introduce as the *decisional point preimage problem* (DPP). We show that this problem reduces to the decisional supersingular product problem (DSSP) introduced in [27].
4. We discuss applications of algorithms to the protocols introduced in [3,10,17,20,21,22,23,27] and benchmark our delegation algorithms for various standardized SIKE primes ($p434, p503, p610, p751$) corresponding to NIST's security levels 1, 2, 3 and 5. We also indicate the necessary communication costs between the delegator and the servers. Iso allows to reduce the delegator's cost in the identification protocols of [17,27] to about 6% of the local computation cost in the OMTUP and 11% in the HBC assumption for $p503$. On the other hand, IsoDetour allows to reduce the cost of SIDH-type public-key generation to about 20% and 35% for OMTUP and HBC, respectively.

2 Background

2.1 Elliptic Curves and Isogenies

We work with supersingular elliptic curves over the field $\mathbb{F}_{p^2}$ with p prime and with Frobenius trace $t_\pi = \mp 2p$. The group of points on elliptic curves of this

type is given as $E(\mathbb{F}_{p^2}) \simeq (\mathbb{Z}/(p \pm 1)\mathbb{Z})^2$ [40], so that the choice of p allows full control of the subgroup structure. Like most isogeny-based schemes, e.g. [27,39], we use $t_\pi = -2p$. The elliptic curves with $t_\pi = 2p$ correspond to the quadratic twists of these curves, i.e. curves having the same j-invariant which become first isomorphic over $\mathbb{F}_{p^4}$. We slightly abuse notation and write e.g. $P \in E$ for $P \in E(\mathbb{F}_{p^2})$. We indicate by $E[\tau]$ the τ-torsion group on $E(\mathbb{F}_{p^2})$ for $\tau \in \mathbb{Z}$ non-zero. Torsion groups of specific points and the generators of these groups are written with the specific point as index, e.g. we write $A \in E[\tau_A]$ and $\langle P_A, Q_A \rangle = E[\tau_A]$, where we assume A to have full order, i.e. $|\langle A \rangle| = \tau_A$. We further use the shorthand $\mathbb{Z}_\tau = \mathbb{Z}/\tau\mathbb{Z}$. We assume that different torsion groups are always coprime.

Isogenies. Isogenies are homomorphisms between two elliptic curves, that are also algebraic maps [16,40]. Separable isogenies are uniquely defined by their kernel. In the cryptographic schemes treated in this work, these kernels are subgroups of torsion groups, generated by a primitive point. For example, the group generated by $A \in E[\tau_A]$, i.e. $\langle A \rangle = \{\lambda A | \lambda \in \mathbb{Z}_{\tau_A}\} \subset E[\tau_A]$, defines the isogeny $\alpha : E \to E/\langle A \rangle$ with $\ker \alpha = \langle A \rangle$. Any other primitive point within $\langle A \rangle$ generates the same isogeny, so we can define the equivalence class $[A]$ of points generating $\langle A \rangle$. One can efficiently verify if two points in $E[\tau_A]$ belong to the same equivalence class by checking if they define the same isogeny or by using pairings. In order to allow efficient isogeny computations between elliptic curves, torsion groups $E[\tau]$ need τ to be smooth [27]. For most cryptographic applications, we require several smooth torsion groups of approximately the same size. This can be guaranteed by choosing $p + 1 = \prod_{i=1}^{n} \tau_i$, where $\tau_i \approx \tau_j$ for all i, j and all smooth. By this choice, supersingular elliptic curves consist of the smooth torsion groups $E[\tau_i]$ for $i = 1, \ldots, n$. Each of these torsion groups is generated by two elements, $\langle P_i, Q_i \rangle = E[\tau_i]$, so any point can be written as a linear combination of these two generators.

Notation. We write isogeny codomains in index notation, e.g. $E_A = E/\langle A \rangle$, $E_{AB} = E/\langle A, B \rangle$, where the index represents (the equivalence class of) the isogeny kernel generator. We represent points on elliptic curves with a superscript corresponding to the index of the elliptic curve they are defined on, e.g. if $P \in E$, then $P^A \in E_A$ and $P^{AB} \in E_{AB}$, where we assume the used map to be clear from context. The same holds for point sets, e.g. $\{P, Q\}^A = \{P^A, Q^A\} \subset E_A$.

2.2 Elliptic Curve Arithmetic

Computational costs. We denote by A and D the theoretical cost estimates of point addition and point doubling on E, respectively, by $\mathsf{S}(\tau)$ the cost estimate of a (large) scalar multiplication of a point by a scalar in $\mathbb{Z}_\tau$ and by $\mathsf{I}(\tau, \mu)$ the cost estimate of computing a (large) τ-isogeny, and pushing μ points through this isogeny. Each of these operations can be expressed in terms of the cost of multiplications m of elements over $\mathbb{F}_{p^2}$. To this end, we assume that squaring on $\mathbb{F}_{p^2}$ costs $0.8\mathsf{m}$, while addition on $\mathbb{F}_{p^2}$ and comparisons are negligible. Expensive

inversions are circumvented by using projective coordinates. Large scalar multiplications are typically done using a double-and-add approach, so that we can express the cost of scalar multiplication by an element τ as [34]

$$\mathsf{S}(\tau) = \mathsf{M}\lceil \log_2 \tau \rceil - \mathsf{A}\,, \quad \text{where} \quad \mathsf{M} = \mathsf{A} + \mathsf{D}\,. \tag{1}$$

Scalar multiplications by a small prime ℓ_i are written as S_{ℓ_i}. We further define C_{ℓ_i} and P_{ℓ_i} as the cost of a computing the codomain of an ℓ_i-isogeny and evaluating an ℓ_i-isogeny respectively. In the full version of the paper, we establish the following cost of a τ-isogeny with $\tau = \prod_{i=1}^{n} \ell_i^{e_i}$:

$$\mathsf{I}(\tau, \mu) = \sum_{i=1}^{n} \left[(\mathsf{P}_{\ell_i} + \mathsf{S}_{\ell_i}) \frac{e_i}{2} \log_2 e_i + (\mathsf{C}_i + \mu \mathsf{P}_i) e_i \right] + \sum_{i=1}^{n-1} \mathsf{P}_{\ell_i} e_i (n-i)\,. \tag{2}$$

Elliptic curve models. We will work with elliptic curves in Montgomery [31] and in twisted Edwards form with extended coordinates [4,24]. For lack of space, we refrain from fully defining them here, and refer the reader to the original sources for more information. Montgomery curves are used in most deployed isogeny-based protocol, as they are particularly efficient if they are reduced to the Kummer line. However, points on the Kummer line form no longer a group, and addition operations have to be substituted by differential additions. Optimized arithmetic on twisted Edwards curves is a bit slower, but points still form a group, which will prove necessary for some of our applications. Note that there is a one-to-one correspondence between Montgomery and twisted Edwards curves, and switching between equivalent curves can be done very efficiently [4,8,30]. As estimates for the cost of point addition and doubling on Montgomery curves over $\mathbb{F}_{p^2}$, we use [5,15]

$$\mathsf{A} = 5.6\mathsf{m}, \quad \mathsf{D} = 3.6\mathsf{m} \quad \text{and} \quad \mathsf{M} = 9.2\mathsf{m}\,,$$

where M represents a step in the Montgomery ladder algorithm [31]. For twisted Edwards curves with extended coordinates, we find [24]

$$\mathsf{A} = 9\mathsf{m}, \quad \mathsf{D} = 7.2\mathsf{m} \quad \text{and} \quad \mathsf{M} = 16.2\mathsf{m}\,,$$

where M represents a step in the typical double-and-add scheme.

Isogeny computations will always be performed on Montgomery curves, for which we can use the optimized results from [14]:

$$\begin{array}{lll} \mathsf{C}_3 = 4.4\mathsf{m}, & \mathsf{S}_3 = 9.2\mathsf{m}, & \mathsf{P}_3 = 5.6\mathsf{m}, \\ \mathsf{C}_4 = 3.2\mathsf{m}, & \mathsf{S}_4 = 7.2\mathsf{m}, & \mathsf{P}_4 = 7.6\mathsf{m}, \end{array}$$

2.3 Security Model

The security model for delegating cryptographic computations used throughout this paper was originally proposed by Hohenberger and Lysyanskaya [25]. In this model, delegation algorithms are split into a trusted component $\mathcal{T}$ and a set of

untrusted servers $\mathcal{U}$. The delegator makes oracle queries to the servers such that their interaction $\mathcal{T}^{\mathcal{U}}$ results in the correct execution of an algorithm Alg with the goal of reducing the computational cost of $\mathcal{T}$ when compared to the local execution of Alg. Since $\mathcal{U}$ might potentially be malicious, the delegator needs to both ensure that $\mathcal{U}$ is not able to extract any sensitive data from the interaction, and be able to verify that the results returned by $\mathcal{U}$ are computed correctly. The full adversary in this model $\mathcal{A} = (\mathcal{E}, \mathcal{U})$ further includes the environment $\mathcal{E}$, representing any third party, that should also not be able to extract sensitive data, while having a different view of the inputs and output of Alg as $\mathcal{U}$ does.

The *outsource input/output specification* (or *outsource-IO*) distinguishes *secret* (only $\mathcal{T}$ has access), *protected* ($\mathcal{T}$ and $\mathcal{E}$ have access) and *unprotected* (everyone has access) inputs and outputs, while non-secret inputs are further subdivided into *honest* and *adversarial*, depending on whether they originate from a trusted source or not. An important assumption of this model is that, while the servers in $\mathcal{U}$ and the environment $\mathcal{E}$ might initially devise a joint strategy, there is no direct communication channel between the different servers within $\mathcal{U}$ or between $\mathcal{U}$ and the environment $\mathcal{E}$ after $\mathcal{T}$ starts using them ($\mathcal{U}$ can be seen to be installed behind $\mathcal{T}$'s firewall). However, they could try to establish an indirect communication channel via the unprotected inputs and un/protected outputs of Alg. To mitigate this threat, $\mathcal{T}$ should ensure that the adversarial, unprotected input stays empty (see also Remark 2.4 in [25]), while the non-secret outputs do not contain any sensitive data. The security of delegation schemes is formalized in the following definition, which also formalizes $\mathcal{T}$'s efficiency gain due to the delegation, as well as its ability to verify correctness of $\mathcal{U}$'s outputs.

Definition 1 ((α, β)-outsource-security [25]). *Let* Alg *be an algorithm with outsource-IO. The pair* $(T, \mathcal{U})$ *constitutes an* outsource-secure *implementation of* Alg *if:*

- ***Correctness****:* $\mathcal{T}^{\mathcal{U}}$ *is a correct implementation of Alg.*
- ***Security****: For all PPT adversaries* $\mathcal{A} = (\mathcal{E}, \mathcal{U})$*, there exist PPT simulators* $(\mathcal{S}_1, \mathcal{S}_2)$ *that can simulate the views of* $\mathcal{E}$ *and* $\mathcal{U}$ *indistinguishable from the real process. We write* $\mathcal{E}VIEW_{real} \sim \mathcal{E}VIEW_{ideal}$ *(*$\mathcal{E}$ *learns nothing) and* $\mathcal{U}VIEW_{real} \sim \mathcal{U}VIEW_{ideal}$ *(*$\mathcal{U}$ *learns nothing). The details of these experiments can be found in Definition 2.2 of [25]. If* $\mathcal{U}$ *consists of multiple servers, then there is a PPT-simulator* $\mathcal{S}_{2,i}$ *for each of their views.*
- ***Cost reduction****: for all inputs* x*, the running time of* $\mathcal{T}$ *is at most an* α*-multiplicative factor of the running time of* $\mathsf{Alg}(x)$*,*
- ***Verifiability****: for all inputs* x*, if* $\mathcal{U}$ *deviates from its advertised functionality during the execution* $\mathcal{T}^{\mathcal{U}}(x)$*, then* $\mathcal{T}$ *will detect the error with probability* $\geq \beta$*.*

Adversarial models differ along the number and intent of servers. The models we will analyze in this work are the following.

Definition 2 (Honest-but-curious [11]). *The* one honest-but-curious program model *defines the adversary as* $\mathcal{A} = (\mathcal{E}, \mathcal{U})$*, where* $\mathcal{U}$ *consists of a single server that always returns correct results, but may try to extract sensitive data.*

Definition 3 (OMTUP [25]). *The* one-malicious version of a two untrusted program model *defines the adversary as* $\mathcal{A} = (\mathcal{E}, (\mathcal{U}_1, \mathcal{U}_2))$ *and assumes that at most one of the two servers* $\mathcal{U}_1$ *or* $\mathcal{U}_2$ *deviates from its advertised functionality (for a non-negligible fraction of the inputs), while* $\mathcal{T}$ *does not know which one.*

We refer to the paper of Hohenberger and Lysyanskaya [25] for other security models without any honest party, namely the *two untrusted program model* (TUP) and the *one untrusted program model* (OUP). Models without honest entity are further discussed in the full version of this paper [35].

2.4 Cryptographic Protocols and Difficulty Assumptions

Let $E/\mathbb{F}_{p^2}$ be a publicly known supersingular elliptic curve with at least two coprime torsion groups $\langle P_A, Q_A \rangle = E[\tau_A]$ and $\langle P_B, Q_B \rangle = E[\tau_B]$, whose generators are also publicly known. Cryptographic protocols in the SIDH setting are generally based on the following commutative diagram:

$$\begin{array}{ccc} E & \xrightarrow{\quad\alpha\quad} & E_A \\ {\scriptstyle\beta}\downarrow & & \downarrow{\scriptstyle\beta'} \\ E_B & \xrightarrow{\quad\alpha'\quad} & E_{AB} \end{array}$$

Let $\langle A \rangle = \ker \alpha$ and $\langle B \rangle = \ker \beta$, then the commutativity of the upper diagram is given by choosing $\ker \alpha' = \langle A^B \rangle$ and $\ker \beta' = \langle B^A \rangle$.

We revisit some of the security assumptions upon which isogeny-based cryptographic protocols are based. Note that we only show the ones that are explicitly used in this work. For other hard problems, we refer for example to [27].

Problem 1 (Computational Supersingular Isogeny Problem (CSSI) [27]). Given the triplet (E_B, P_A^B, Q_A^B), find an element in $[B] \subset E[\tau_B]$.

Problem 2 (Decisional Supersingular Product Problem (DSSP) [27]). Let $\alpha : E \to E_A$. Given a tuple $(E, E_A, E_1, E_2, \alpha, \alpha')$, determine from which of the following distributions it is sampled

- E_1 is random with $|E_1| = |E|$ and $\alpha' : E_1 \to E_2$ is a random τ_A-isogeny,
- $E_1 \times E_2$ is chosen at random among those isogenous to $E \times E_A$ and where $\alpha' : E_1 \to E_2$ is a τ_A-isogeny.

We further define the following difficulty assumption and show that it is at least as hard as DSSP.

Problem 3 (Decisional Point Preimage Problem (DPP)). Given (E, E_B, A, A'^B), where $A \in E[\tau_A]$, and $A'^B \in E_B[\tau_A]$, decide whether $[A] = [A']$.

Let $\mathcal{A}_{\mathrm{DPP}}$ be an adversary to the DPP problem which, upon receiving the tuple (E, E_B, A, A'^B), returns $b = 1$ if $[A^B] = [A'^B]$, otherwise $b = 0$. Then, we can construct an adversary $\mathcal{B}^{\mathcal{A}_{\mathrm{DPP}}}_{\mathrm{DSSP}}$ against DSSP, which returns $b = 0$ in the first and $b = 1$ in the second case of Problem 2. Upon receiving $(E, E_A, E_B, E_C, \alpha, \alpha')$, $\mathcal{B}^{\mathcal{A}_{\mathrm{DPP}}}_{\mathrm{DSSP}}$ extracts kernel generators $\langle S \rangle = \ker \alpha$ and $\langle S'^B \rangle = \ker \alpha'$, then sends the query (E, E_B, S, S'^B) to $\mathcal{A}_{\mathrm{DPP}}$. $\mathcal{B}^{\mathcal{A}_{\mathrm{DPP}}}_{\mathrm{DSSP}}$ returns what $\mathcal{A}_{\mathrm{DPP}}$ returns: if $[S] = [S']$, then $E_B \times E_C$ is isogenous to $E \times E_A$ and we have $b = 1$, otherwise $b = 0$.

3 Delegating Isogenies

Throughout this section, we assume that the delegator $\mathcal{T}$ is able to generate elements in $\mathbb{Z}$ uniformly at random in an efficient manner. We further assume that $\mathcal{T}$ knows a representation of any of its secret and protected points in terms of the public torsion group generators.

3.1 Advertised Server Functionality

Let $E/\mathbb{F}_{p^2}$ be an elliptic curve, $\mathcal{K} \subset \mathbb{Z}_\tau \times E[\tau]$, $\mathcal{M} \subset \mathbb{Z} \times E$ two distinct sets of scalar-point pairs and $b \in \{0, 1\}$ a bit. We assume that the delegator gives inputs of the form $(E, \mathcal{K}; \mathcal{M}; b)$ to the server, who proceeds as follows.

- $\mathcal{K}$ encodes the kernel of the isogeny to compute, thus the server computes $K = \sum_{(a,P)\in\mathcal{K}} aP$, which it uses to compute the isogeny $\phi : E \to E_K$. Throughout this work, we are only interested in sets of the form $\mathcal{K} = \{(1, P), (k, Q)\}$ for generators $\langle P, Q \rangle = E[\tau]$.
- $\mathcal{M}$ contains points to push through and multiply with the associated scalar, i.e. the server computes $\mathcal{M}^K := \{aX^K \mid (a, X) \in \mathcal{M}\}$, where $X^K = \phi(X)$.
- If $b = 1$, the server generates a deterministic *return basis* $\mathcal{B}^K = \{R^K, S^K\} \subset E_K[\tau]$, such that $R^K + kS^K = P^K$.[3] If $b = 0$, then $\mathcal{B}^K = \emptyset$.

The server then returns $(E_K; \mathcal{M}^K; \mathcal{B}^K)$. We write the delegation step as follows

$$(E_K; \mathcal{M}^K; \mathcal{B}^K) \leftarrow \mathcal{U}(E, \mathcal{K}; \mathcal{M}; b).$$

The points in $\mathcal{M}$ are always submitted in a random order in order to avoid distinguishability. Further, to reduce the communication cost we assume that servers return all points scaled with $Z = 1$.

Notation. For a scalar-point pair (a, P) in $\mathcal{K}$ or $\mathcal{M}$, we simply write P if $a = 1$. If a set contains multiple pairs of the same point, e.g. $\{(a_1, P), (a_2, P), (a_3, P)\}$, we condense them as $\{(\{a_1, a_2, a_3\}, P)\}$.

[3] This can simply be achieved by first generating $S^K \in E_K[\tau]$ deterministically (e.g. by hashing into the elliptic curve using a procedure such as the one described in [26], and map out the unwanted torsion), then computing $R^K = P^K - kS^K$.

3.2 The Iso-Algorithm

Definition 4 (The Iso-algorithm). *The* isogeny delegation algorithm Iso *takes as inputs a supersingular elliptic curve $E/\mathbb{F}_{p^2}$, a kernel set $\mathcal{K} \subset \mathbb{Z} \times E(\mathbb{F}_{p^2})$, two scalar-point pair sets $\mathcal{H}_0, \mathcal{H} \subset \mathbb{Z} \times E(\mathbb{F}_{p^2})$ and a bit $b \in \{0,1\}$, then computes the isogeny $\phi : E \to E_K$ and produces the output $(E_K; \mathcal{H}_0^K, \mathcal{H}^K; \mathcal{B}^K)$, where $K = \sum_{(a,P)\in\mathcal{K}} aP$, $\mathcal{H}_{(0)}^K = \{aP^K \mid (a,P) \in \mathcal{H}_{(0)}\}$ and $\mathcal{B}^K$ is a return basis as described in Section 3.1, if $b = 1$ and $\emptyset$ otherwise. The inputs $E, \mathcal{K}, \mathcal{H}_0, b$ are all honest, unprotected parameters, while $\mathcal{H}$ contains secret or (honest/adversarial) protected scalars and honest, unprotected points. The outputs E_K, $\mathcal{H}_0^K$ and $\mathcal{B}^K$ are unprotected while $\mathcal{H}^K$ is secret or protected. We write*

$$(E_K; \mathcal{H}_0^K, \mathcal{H}^K; \mathcal{B}^K) \leftarrow \mathsf{Iso}(E, \mathcal{K}; \mathcal{H}_0, \mathcal{H}; b).$$

If $b = 0$ and thus $\mathcal{B}^K = \emptyset$, we shorten this as $(E_K; \mathcal{H}_0^K, \mathcal{H}^K) \leftarrow \mathsf{Iso}(E, \mathcal{K}; \mathcal{H}_0, \mathcal{H})$.

In Figures 1 and 2, we show how a delegator $\mathcal{T}$ can use the advertised server functionality from Section 3.1 in order to implement Iso in an outsource-secure way under the HBC and OMTUP assumptions. The delegation subroutines are organized according to 5 main steps: First, auxiliary elements are generated (Gen), which are used to shroud protected elements (Shr), before being delegated to the server (Del). After the delegation, the server outputs are verified (Ver) and finally the results are recovered and output (Out).

Note that the HBC case does not need a verification step by assumption. The idea behind Figure 1 is relatively trivial but effective: the delegator hides the secret/protected scalars simply by not disclosing them to the server and computing the scalar multiplication on the codomain point itself. The OMTUP case of Figure 2 is a bit more complex, but will result in a lower cost for the delegator when compared to the HBC case. The underlying idea (for $N = 1$) is that the delegator shrouds the secret/protected scalars as a linear combination of small and large random scalars. The large scalars are distributed between the two servers in order to prevent reconstruction of the secrets, while the small scalars are kept secret by the delegator and used to ultimately verify correctness of the returned points. The size of the small scalars influences the cost for the delegator and the verifiability of the protocol. To further increase verifiability, the delegator can add more random scalars to the mix by increasing N, which leads to multiple, interconnected verification conditions, and results in an even higher verifiability, albeit at a higher cost for the delegator. There is an optimal trade-off between these two parameters, depending on the desired verifiability. We will discuss this trade-off further in Section 3.2. In the full version of this paper [35], we establish the protocol execution costs for the delegator

$$T_{\mathrm{HBC}}(\mu, \tau_A) = \mu \mathsf{S}(\tau_A)\,, \tag{3}$$

$$T_{\mathrm{OMTUP}}(\mu, t) = \mu\left[(4N+3)\mathsf{m} + 2\mathsf{M}t + (2^{N+1} - N - 3)\mathsf{A}\right]\,, \tag{4}$$

of Figures 1 and 2 and further prove the following theorems.

Theorem 1. *Under the honest-but-curious assumption, the outsourcing algorithm* $(\mathcal{T},\mathcal{U})$ *given in Figure 1 is a* $\left(O\left(\frac{1}{\log\log\tau}\right),1\right)$*-outsource secure implementation of* Iso*, where* τ *is the smooth degree of the delegated isogeny.*

Theorem 2. *Under the OMTUP assumption, the outsourcing algorithm* $(\mathcal{T},(\mathcal{U}_1,\mathcal{U}_2))$ *given in Figure 2 is an* $\left(O\left(\frac{t}{\log\tau\log\log\tau}\right),1-\frac{1}{(N+1)2^{Nt}}\right)$*-outsource secure implementation of* Iso*, where* τ *is the smooth degree of the delegated isogeny. If* $\mathcal{H}=\emptyset$*, then it is fully verifiable.*

Hiding a point. If the delegator wants to push through a secret or (honest/adversarial) protected elliptic curve point $A = P+aQ \in E[\tau_A]$, then $\mathcal{T}$ simply has to delegate

$$(E_K;\mathcal{H}_0^K\cup\{P\},\mathcal{H}^K\cup\{aQ^K\};\mathcal{B}^K)\leftarrow\mathsf{Iso}(E,\mathcal{K};\mathcal{H}_0\cup\{P\},\mathcal{H}\cup\{(a,Q)\};b)\,,$$

and compute $A^K = P^K + aQ^K$. We assume that a representation of A in the normal form is always known, as will always be the case in the cryptographic protocols that we discuss in this paper.

Honest-but-curious approach.

Gen: No auxiliary elements are needed.
Shr: Set $\mathcal{H}' = \{Q \mid (a,Q)\in\mathcal{H}\}$.
Del: Delegate $(E_K;\mathcal{H}_0^K\cup\mathcal{H}'^K;\mathcal{B}^K)\leftarrow\mathcal{U}(E,\mathcal{K};\mathcal{H}_0\cup\mathcal{H}';b)$.
Out: Compute $\mathcal{H}^K=\{aQ^K\mid(a,Q)\in\mathcal{H},Q^K\in\mathcal{H}'^K\}$, then return $(E_K;\mathcal{H}_0^K,\mathcal{H}^K;\mathcal{B}^K)$.

Fig. 1. Implementation of Iso in the HBC assumption

OMTUP approach.

Gen: For each $(a,Q)\in\mathcal{H}$, choose $N\in\mathbb{N}$, then (assuming $Q\in E[\tau]$) generate
- small non-zero scalars $c_1,\dots,c_N,d_1,\dots,d_N\in\{-2^{t-1},\dots,2^{t-1}\}$, and
- random scalars $r_0,s_0,s_1,\dots,s_{N-1}\in\mathbb{Z}_\tau$.

Shr: For each $(a,Q)\in\mathcal{H}$, compute $r_i=-s_i+c_is_0+d_ir_0$ for $i=1,\dots,N-1$. Define $\sigma=\sum_{i=1}^{N-1}(s_i+r_i)$ and let γ be the smallest integer >1 coprime to τ, then compute $s_N=\gamma^{-1}(d_Nr_0+c_Ns_0+\sigma-a)$ and $r_N=-s_N+c_Ns_0+d_Nr_0$. Set

$$\mathcal{H}_1'=\{(\{s_0,\dots,s_N\},Q)\mid(a,Q)\in\mathcal{H}\}\,,\quad \mathcal{H}_2'=\{(\{r_0,\dots,r_N\},Q)\mid(a,Q)\in\mathcal{H}\}\,.$$

Del: Delegate $(E_K;\mathcal{H}_0^K\cup\mathcal{H}_1'^K;\mathcal{B}^K)\leftarrow\mathcal{U}_1(E,\mathcal{K};\mathcal{H}_0\cup\mathcal{H}_1';b)$ and $(E_K';\mathcal{H}_0'^K\cup\mathcal{H}_2'^K;\mathcal{B}'^K)\leftarrow\mathcal{U}_2(E,\mathcal{K};\mathcal{H}_0\cup\mathcal{H}_2';b)$.
Ver: Verify, if $E_K\stackrel{?}{=}E_K'$, $\mathcal{H}_0^K\stackrel{?}{=}\mathcal{H}_0'^K$, $\mathcal{B}^K\stackrel{?}{=}\mathcal{B}'^K$, and if $(s_iQ)^K+(r_iQ)^K\stackrel{?}{=}c_i(s_0Q)^K+d_i(r_0Q)^K$, for $i=1,\dots,N$.
Out: If any of the verifications fail, return $\perp$, otherwise return $(E_K;\mathcal{H}_0^K,\mathcal{H}^K;\mathcal{B}^K)$, where

$$\mathcal{H}^K=\left\{r_NQ^K-(\gamma-1)s_NQ^K+\sum_{i=1}^{N-1}(s_iQ^K+r_iQ^K)\Big|(a,Q)\in\mathcal{H}\right\}.$$

Fig. 2. Implementation of Iso in the OMTUP assumption

The parameter t. In some cases, the parameter t does not only influence the verifiability and cost of the underlying system, but also its security. Related attacks become unfeasible, if the size of t reflects the security of the underlying cryptosystem against both classical and quantum attackers, i.e. in general we need to ensure that guessing all $c_1, \ldots, c_N$ correctly is at least as hard as some targeted security level 2^λ, i.e. $(N+1)2^{Nt} \approx 2^\lambda$ or $t \approx \frac{\lambda}{N}$. In this case, using equation (4), the protocol cost per hidden point becomes

$$\mu^{-1} T_{\mathrm{OMTUP}}(\mu, \lambda/N) = (4N+3)\mathsf{m} + \frac{2\lambda}{N}(\mathsf{D}+\mathsf{A}) + \left(2^{N+1} - N - 3\right)\mathsf{A}\,.$$

In Section 5, we minimize this cost with respect to N for specific choices of λ. Note that choosing $tN = \lambda$ further implies a verifiability of $1 - O(2^{-\lambda})$, which is very close to 1 for a cryptographically sized λ.

Difference to delegation of mudular exponentiation. We want to point out a few key differences of isogeny delegation schemes to those of modular exponentiation as in [11,25,29]. First of all, in contrast to modular exponentiations, the domain and codomain of isogenies are different (except in the trivial case where $\mathcal{K} = \emptyset$), and more importantly, these are a priori unknown to the delegator. This means that the delegator not only has to verify if the codomain is correct, but also can not generate points on the codomain before the delegation step is completed. This also means that lookup-tables with points in the domain and codomain curves are not possible, hence the delegator can compute the final result only from linear combinations of elements the server(s) returned. Another circumstance of isogenies is that elliptic curves can not be combined in an easy way without computing isogenies, which means that combinations, such as $(A, E_A) \circ (B, E_B) = ((A, B), E_{AB})$ are not available to the delegator.

Now we turn our attention to what the delegator actually can do. One of the most important properties of isogenies in this context is that they are group homomorphisms. This means that linear combinations of points on the domain curve still hold on the codomain curve and can therefore be used to shroud and verify points, as Iso does. In order to verify the codomain curve, there seems to be no efficient way except for including at least one honest server, which will consistently return the correct curve and verify the malicious servers' results against it. The honest server is also necessary to verify if mapped points are correct. If none of the servers were honest, all points could be scaled by some previously determined factors, returning wrong results, which would still satisfy the verification conditions.

4 Shrouding Isogenies

We aim to hide the kernel generator $A \in E[\tau_A]$ via the isogenies generated by a coprime torsion group $E[\tau_I]$ with $\tau_I \approx \tau_A$. The idea is to go from E to E_A via the path $E \xrightarrow{\kappa} E_K \xrightarrow{\alpha} E_{AK} \xrightarrow{\hat{\kappa}'} E_A$, where $\hat{\kappa}'$ is the dual of κ pushed through α. The point A (or the isogeny α) is hidden

via the isogeny $A^K = \kappa(A)$, since the knowledge of $[A^K]$ does not give any information about $[A]$ by the DPP-assumption (Problem 3). Note that our approach necessarily has to take at least three steps, since any linear combination of A with elements from $E[\tau_I]$ (i.e. any "shortcut") would always reveal information about A by mapping out the τ_I-torsion elements. Similarly, any shorter isogeny, smaller than the length of $\tau_A \approx \tau_I$, would reduce the security of the system.

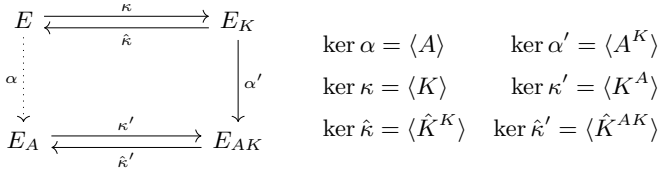

Fig. 3. Detour from $E \to E_A$ via E_K and E_{AK} and the associated kernel generators. The point $\hat{K}$ is any point of full order in $E[\tau_I]\backslash\langle K\rangle$.

Another important aspect is that any server that has computed the delegation in Step 2 should not see any information of the delegation performed in Steps 1 or 3 (and vice versa), since the knowledge of K (or $\hat{K}^{AK}$) and A^K can be used to recover A. We therefore in general need to work with multiple sets of servers, each being composed of one or more servers according to the underlying server assumptions (e.g. HBC, OMTUP). We denote these sets as $\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3$, for delegation steps 1, 2 and 3. Under certain conditions, we can choose $\mathbf{U}_1 = \mathbf{U}_3$, which we will discuss further below. We also note, that in the OMTUP case, the malicious servers within these sets could exchange their knowledge about the kernel generators indirectly, which also needs to be addressed in our algorithm.

Definition 5 (The IsoDetour-algorithm). *The* isogeny detour delegation algorithm IsoDetour *takes as inputs a supersingular elliptic curve $E/\mathbb{F}_{p^2}$, a kernel generator $A = P_A + aQ_A$ where $\langle P_A, Q_A\rangle = E[\tau_A]$, two scalar-point pair sets $\mathcal{H}_0, \mathcal{H} \subset \mathbb{Z} \times E\backslash(E[\tau_A] \cup E[\tau_I])$, and a torsion-group indicator I. It then computes the isogeny $\phi : E \to E_A$ as $\phi = \kappa' \circ \alpha' \circ \kappa$ via the kernels $\ker\kappa = \langle K\rangle$, $\ker\alpha' = \langle A^K\rangle$ and $\ker\kappa' = \langle \hat{K}^{AK}\rangle$, where $K, \hat{K} \in E[\tau_I]$, both of full order and such that $\langle\hat{K}\rangle \neq \langle K\rangle$.* IsoDetour *then produces the output $(E_A; \mathcal{H}_0^A, \mathcal{H}^A)$. The inputs $E, \mathcal{H}_0$ are honest, unprotected parameters. $\mathcal{A}$ is secret, or (honest/adversarial) protected and $\mathcal{H}$ contains honest, unprotected points and secret or (honest/adversarial) protected scalars. The outputs E_A and $\mathcal{H}_0^A$ are unprotected while $\mathcal{H}^A$ is secret or protected. We write*

$$(E_A; \mathcal{H}_0^A, \mathcal{H}^A) \leftarrow \mathsf{IsoDetour}(E, A, I; \mathcal{H}_0, \mathcal{H}).$$

In Figure 4, we present the IsoDetour-Algorithm, that uses the commutative diagram from Figure 3 in order to delegate α via a detour over the curves E_K and E_{AK}. We assume that the generators $\langle P_I, Q_I\rangle = E[\tau_I]$ are known.

IsoDetour proceeds as follows: First, the isogeny κ is delegated to $\mathbf{U}_1$ and the point A is pushed through, hidden from the servers. The servers are also

prompted to return a basis $R^K, S^K \in E_K$, such that $R^K + kS^K = P^K \in \ker \hat{\kappa}$. These points will later be used to compute the "return" isogeny $\hat{\kappa}'$. The point A^K is then used as the kernel generator for α', computed by $\mathbf{U}_2$, with R^K, S^K are pushed through. Finally, the delegator constructs the kernel generator $R^{AK} + kS^{AK}$ of $\hat{\kappa}'$ for the third delegation by $\mathbf{U}_3$. For any other scalar-point pair, that we want to push through, the general idea is to extract the (unprotected) points in $\mathcal{H}_0$ and $\mathcal{H}$ and simply push them through the first two rounds of delegation; the desired multiplication with hidden scalars needs to be done in the third round only. Note that since these points are pushed through κ and later through $\hat{\kappa}'$, the result will be multiplied by a factor $\deg \kappa = \deg \hat{\kappa}' = \tau_I$. Thus, we need to multiply the related scalars with τ_I^{-1}, in order to compensate for this.

$\mathsf{IsoDetour}(E, A, I; \mathcal{H}_0, \mathcal{H})$

1. Generate random $k \in \mathbb{Z}^*_{\tau_I}$ and let $\mathcal{H}'_0 = \{Q \mid (a, Q) \in \mathcal{H}_0 \cup \mathcal{H}\}$.
2. Delegate to server (group) $\mathbf{U}_1$ (in the OMTUP case, choose $tN \geq \lambda$)

$$(E_K; \mathcal{H}'^K_0 \cup \{P^K_A\}, \{aQ^K_A\}; \{R^K, S^K\}) \leftarrow \mathsf{Iso}(E, \{P_I, (k, Q_I)\}; \mathcal{H}'_0 \cup \{P_A\}, \{(a, Q_A)\}; 1).$$

   and compute $A^K = P^K_A + aQ^K_A$.
3. Delegate to server (group) $\mathbf{U}_2$ (in the OMTUP case, choose $tN \geq \lambda$)

$$(E_{AK}; \{R^{AK}\} \cup \mathcal{H}'^{AK}_0, \{kS^{AK}\}; \emptyset) \leftarrow \mathsf{Iso}(E_K, \{A^K\}; \{R^K\} \cup \mathcal{H}'^K_0, \{(k, S^K)\}; 0).$$

4. From $\mathcal{H}'^{AK}_0$, build $\mathcal{H}''^{AK}_{(0)} = \{(a\tau_I^{-1}, Q^{AK}) \mid (a, Q) \in \mathcal{H}_{(0)}\}$. Then, compute the kernel $\hat{K}^{AK} = R^{AK} + kS^{AK}$.
5. Delegate $(E_A; \mathcal{H}^A_0, \mathcal{H}^A; \emptyset) \leftarrow \mathsf{Iso}(E_{AK}, \{\hat{K}^{AK}\}; \mathcal{H}''^{AK}_0, \mathcal{H}''^{AK}; 0)$ to server (group) $\mathbf{U}_1$.
6. Return $(E_A; \mathcal{H}^A_0, \mathcal{H}^A)$.

Fig. 4. Implementation of the $\mathsf{IsoDetour}$ algorithm given in Definition 5 using the Iso algorithm from Definition 4 as a subroutine.

Mapping points. Note that since $\hat{\kappa}'$ is represents the dual isogeny of κ pushed through α', any points mapped via the detour path will necessarily by multiplied by τ_I. This is corrected in step 4 by multiplying these points with the inverse of τ_I. Note that this multiplication is only defined for points in torsion groups of order coprime to τ_I,[4] thus not for points in $E[\tau_I]$. An important aspect of SIDH and related protocols (such as SIKE [2,39] and the PKE from [27]) is that there are two large torsion groups $E[\tau_A]$, $E[\tau_B]$ with generators P_A, Q_A and P_B, Q_B, respectively. Each party chooses a torsion group, in which it computes its isogeny. Then it transports the generators of the other torsion group via its isogeny to the codomain curve in order to create their public key, e.g. the public key of Alice is (E_A, P^A_B, Q^A_B). These point maps turn out to be a problem for the $\mathsf{IsoDetour}$-algorithm, since any point in $E[\tau_B]$ will map to $\mathcal{O}$ on E_A, and we are not able to map P_B, Q_B along this path. We present two ways to circumvent this

[4] We assume τ_I^{-1} to be known with respect any other torsion group.

problem below. We also note that due to the security constraints of IsoDetour, we also cannot map points in $E[\tau_A]$ to E_A. Fortunately, this is not necessary for the cryptographic protocols analyzed in this work.

More torsion groups. Assuming the protocol has more torsion groups than two, we can easily transport Bob's kernel generators $P_B, Q_B \in E[\tau_B]$ by doing a detour via isogenies defined over a third torsion group $I \neq A, B$. More generally, let $p = \prod_{i=1}^{n} \tau_i \mp 1$ with $n > 2$, then Alice can delegate the computation of her public key (E_A, P_B^A, Q_B^A) as

$$(E_A; \{P_B, Q_B\}^A, \emptyset) \leftarrow \mathsf{IsoDetour}(E, A, I; \{P_B, Q_B\}, \emptyset)\,.$$

Working with twists. If we are working with a prime of the form $p \pm 1 = f\tau_A\tau_B$, i.e. we only have two torsion groups at our disposal on E, we can use twists to generate "new" torsion groups [13] on E^t. Assuming the prime decomposition $p \mp 1 = D\tau_S$, with $\tau_S \approx \tau_A$ smooth and D a co-factor, we have another torsion group on the "backside" of our elliptic curve, $E^t[\tau_S]$. We can simply delegate the public key computation via

$$(E_A; \{P_B, Q_B\}^A, \emptyset) \leftarrow \mathsf{IsoDetour}(E, A, S; \{P_B, Q_B\}, \emptyset),$$

by running over the twists $E \simeq E^t \to E_K^t \to E_{AK}^t \to E_A^t \simeq E_A$. For efficiency reasons, τ_S has to be smooth. There are not many primes p such that $p \pm 1$ and $\tau_S \mid p \mp 1$ are smooth. We call primes of this type *delegation-friendly primes* and generalize them in the following definition. We present an approach to generate such primes in the full version of the paper [35].

Definition 6 (Delegation-friendly primes). *An n-*delegation-friendly prime *(DFP) is a prime p with n smooth factors $\prod_{i=1}^{n} \tau_i \mid p \pm 1$ and at least one smooth factor $\tau_S \mid p \mp 1$, such that $\tau_i \approx \tau_S$ for all i.*

We discuss under which conditions we can choose $\mathbf{U}_1 = \mathbf{U}_3$. An important consequence of using multiple torsion groups or delegation-friendly primes are the susceptibility to torsion-attacks as described in [36,37]. The security of such a delegation depends strongly on the points revealed on E_K and E_{AK}, which in turn reveal the action of α' on these subgroups. As an example, consider standard SIDH with a DFP, i.e. where we have $p \pm 1 = f\tau_A\tau_B$ and $p \mp 1 = \tau_S D$. Using IsoDetour in order to compute a public key reveals the action of α' on $E[\tau_B]$ and $E[\tau_S]$, which would allow a quadratic speedup of the isogeny recovery attack by [37, Prop. 25 and Prop.27]. In this case, we would need three sets of servers in order to not allow this attack. Taking the non-DFP $p \pm 1 = f\tau_A\tau_B\tau_I$ instead, results in a slightly less than quadratic speedup, but in more expensive arithmetic. While small speedups might in some situations not pose a problem, we will discuss under which conditions these occur in Section 5 as well as in the proofs of Theorems 3 and 4, found in the full version of this paper [35]. Note that this does not make our schemes insecure, as we simply point out, under which conditions two server groups can be used instead of three. In the case of

three different server sets, these attacks do not apply.

Choosing t. We point out the issues outlined in Remark 2.4 of [25], which in short states that "*the adversarial, unprotected input must be empty*". In Figure 4, the kernel generators A^K and $\hat{K}^{AK}$ actually do constitute adversarial unprotected inputs, and might allow the malicious server in $\mathbf{U}_1$ to communicate information about K to $\mathbf{U}_2$, revealing information about A. To mitigate this threat, $\mathcal{T}$ can increase the parameter t so far to make this attack at least as hard as breaking the underlying cryptosystem. As discussed in Section 3.2, choosing $tN \geq \lambda$ guarantees that the unprotected inputs are actually honest up to a negligible probability. Note that if such points do not constitute adversarial unprotected inputs, t and N will only influence the cost and verifiability of the protocol. There is no advantage in choosing N different from 1 in this case.

Outsource-security of $\mathsf{IsoDetour}$. In the full version of the paper [35]´, we derive the following costs

$$T^{\mathrm{HBC}}_{\mathsf{IsoDet}}(\mu, \tau_A) = (\mu + 2)\mathsf{S}(\tau_A) + 2\mathsf{A}\,,$$
$$T^{\mathrm{OMTUP}}_{\mathsf{IsoDet}}(\mu, t) = (8N + 6 + 5\mu)\mathsf{m} + \left(\frac{4\lambda}{N} + 2t\mu\right)\mathsf{M} + \left(2^{N+2} - 2N - 3 + \mu\right)\mathsf{A}\,.$$

for the delegator and prove the following theorems.

Theorem 3. *Under the honest-but-curious assumption, the outsourcing algorithm $(\mathcal{T},\mathcal{U})$ given in Figure 4 is an $\left(O\left(\frac{1}{\log\log\tau}\right), 1\right)$-outsource secure implementation of $\mathsf{IsoDetour}$, where τ is the smooth degree of the delegated isogeny.*

Theorem 4. *Under the OMTUP assumption, the outsourcing algorithm $(\mathcal{T}, (\mathcal{U}_1,\mathcal{U}_2))$ given in Figure 4 is an $\left(O\left(\frac{\lambda}{\log\tau\log\log\tau}\right), 1 - \frac{1}{2^{t+1}}\right)$-outsource secure implementation of $\mathsf{IsoDetour}$, where τ is the smooth degree of the delegated isogeny and λ a security parameter. If $\mathcal{H} = \emptyset$, then $\mathsf{IsoDetour}$ is fully verifiable.*

Hiding the kernel generator. A first attempt of hiding the kernel generator of a delegated isogeny was presented with the HIso algorithm of [34]. In the full version of this paper, we show that this scheme is not secure and that the secret can be recovered using pairings. We then discuss how this would be possible using the approach presented in this section. Unfortunately, it turns out to be too expensive for realistic scenarios. In protocols that need a hidden codomain, we therefore assume that the delegator computes them locally.

5 Delegation of Isogeny-based Protocols

We apply our proposed delegation subroutines to some of the cryptographic protocols based on supersingular isogenies over $\mathbb{F}_{p^2}$. In order to assess the computational and communication costs, we will use the 2^{e_2}-torsion groups of the

standardized SIKE primes from [28].[5] To maximize efficiency, we implement the HBC case on Montgomery curves on the Kummer line, while we need a group structure to implement point hiding under the OMTUP-assumption, hence we will use twisted Edwards curves in this case. The efficient transformations between these curves allow seamless integration of our delegation schemes into typically Montgomery-curve based protocols. We assume local computations to always be performed in optimized Montgomery arithmetic.

In the following subsections, we compare the delegated runtimes to the local (non-delegated) cost of some cryptographic protocols. We express our results in terms of the cost reduction function α introduced in Definition 1. To avoid adversarial inputs in the OMTUP-assumption, we use $\lambda = e_2/2$, which reflects the classical security of the underlying protocols. The optimal value of N for all SIKE primes is $N = 4$ (also considering communication costs).

We present our results using the theoretical runtimes established throughout this work and compare them to benchmarks illustrating the runtimes of the delegator under both the HBC- and OMTUP-assumptions.[6] The benchmarks were implemented using Magma v2.25-6 on an Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz with 128 GB memory. Our implementation uses parts of the Microsoft(R) `vOW4SIKE` implementation from [12].[7]

Communication costs between delegator and server are expressed in bits. Let $b(p) = \lceil \log_2 p \rceil$, then elements in $\mathbb{F}_{p^2}$ then contain $2b(p)$ bits of information. We note that Montgomery curves and points on their Kummer line can be expressed by a single $\mathbb{F}_{p^2}$-element, while twisted Edwards curves and their points are expressed using two such elements. Note that we assume $Z = 1$, which can always be achieved by an inversion and that the T-coordinate in the latter case can be recovered by a simple multiplication. In the case $p \approx \prod_{i=1}^{n} \tau_i$ with $\forall i, j : \tau_i \approx \tau_j$, elements in $\mathbb{Z}_{\tau_i}$ can be expressed using approximately $b(p)/n$ bits.

For the sake of conciseness, we assume that the protocols in this section are known. While we briefly review the protocol steps in order to assess the local computation cost, we refer the reader to the original sources for more details.

Remark 1 (Free Delegation). Note that we can freely delegate any protocol that does not need hiding, i.e. where the kernel is unprotected and $\mu = 0$. Verification of the server outputs then reduce to simple comparison operations under the OMTUP-assumption. Some examples of such schemes are isogeny-based hash functions [10,21] with unprotected messages or verifiable delay functions [20].

5.1 Key-agreement protocols

We consider the key agreement protocols from [3,22], which are n-party extensions to SIDH [18]. In this scenario, we have $p + 1 = \prod_{i=1}^{n} \ell_i^{e_i}$ for n parties.

[5] $p434 = 2^{216}3^{137} - 1$, $p503 = 2^{250}3^{159} - 1$, $p610 = 2^{305}3^{192} - 1$, $p751 = 2^{372}3^{239} - 1$.

[6] Our implementation can be found at `https://github.com/gemeis/SIDHdelegation` and includes representative benchmarks for the delegator's operations as well as a proof-of-concept implementation for the correctness of our algorithms.

[7] `https://github.com/microsoft/vOW4SIKE`

Each party $\mathcal{P}_i$ is assigned a subgroup $\langle P_i, Q_i \rangle = E[\ell_i^{e_i}]$ and has a secret key $a_i \in \mathbb{Z}_{\ell_i^{e_i}}$, defining $A_i = P_i + a_i Q_i$ as the kernel of $\phi_i : E \to E_i = E/\langle A_i \rangle$, while the corresponding public key is $(E_i, P_1^i, Q_1^i, \ldots, P_n^i, Q_n^i)$ for party i. While we consider the n-party case in order to stay general, we point out that n-party key agreement protocols have to be used with caution, as torsion point attacks can be quite effective in these settings. In particular, [37] presents improved attacks for $n > 2$ and a polynomial-time break for $n \geq 6$.

Public key generation step. Let Alice be $\mathcal{P}_1$. If $n > 2$, Alice can delegate her public key computation using $\mathsf{IsoDetour}$ twice, along two paths $I_1 \neq I_2$:

$$(E_{A_1}; \mathcal{N}_1^{A_1}, \emptyset) \leftarrow \mathsf{IsoDetour}(E, A_1, I_1; \mathcal{N}_1, \emptyset),$$
$$(E_{A_1}; \mathcal{N}_2^{A_1}, \emptyset) \leftarrow \mathsf{IsoDetour}(E, A_1, I_2; \mathcal{N}_2, \emptyset),$$

where $\mathcal{N}_1 \cup \mathcal{N}_2 = \{(P_i, Q_i)\}_{i \in \{2,\ldots,n\}}$, the set of all other torsion group generators on E, such that $\mathcal{N}_1 \cap \mathcal{N}_2 = \emptyset$ and $(P_{I_1}, Q_{I_1}) \in \mathcal{N}_2$ and $(P_{I_2}, Q_{I_2}) \in \mathcal{N}_1$. By using alternating server groups $\mathbf{U}_1$ and $\mathbf{U}_2$ as indicated in Figure 5, and by carefully choosing $\mathcal{N}_1$ and $\mathcal{N}_2$, we can assure that the servers get as little information as possible about the action of the isogenies α_1' and α_2' on the torsion groups, so that we only need two server groups for delegation.[8]

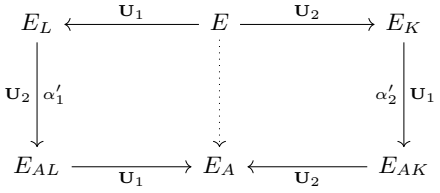

Fig. 5. Alice's concept of delegating the computation of her public key via two detours using two server groups $\mathbf{U}_1$ and $\mathbf{U}_2$. L and K are from different torsion groups.

With an n-DFP, this step can be delegated with a single instance of $\mathsf{IsoDetour}$ using the smooth torsion group on the twist side. This case needs three server groups. Let $d \in \{0, 1\}$ distinguish, if we have an n-DFP ($d = 1$) or not ($d = 0$) at our disposal. The cost reduction for public-key delegation can then be expressed as

$$\alpha_{\mathrm{PubKey},n}(d, \tau_A) = \frac{(2-d)T_{\mathsf{IsoDet}}(0, \tau_A)}{\mathsf{I}(\tau_A, 3(n-1)) + \mathsf{S}(\tau_A) + \mathsf{A}}\,.$$

Figure 6 compares our theoretical estimates with the benchmarked results for $n = 2$, used in most cryptographic protocols. In this case, a delegation-friendly prime is necessary. The communication costs are summarized in Table 1.

[8] For example, we could simply split up generators P_i, Q_i into both sets for all i.

Intermediate steps. If $n > 2$, Alice performs $n-2$ intermediate steps $k \in \{2, \ldots, n-1\}$, in which she has to compute $(E_{k'}, \mathcal{N}^{k'})$ from $(E_k, \mathcal{N}^k \cup \{(P_A^k, Q_A^k)\})$, where $E_{k'} = E_k/\langle P_A^k + a_1 Q_A^k \rangle$ and $\mathcal{N}^{k^{(\prime)}} = \{(P_i^{k^{(\prime)}}, Q_i^{k^{(\prime)}})\}_{i \in \{k+1,\ldots,n\}}$. Note that in this scenario, it is cheaper to compute A_1^k locally and delegate

$$(E_{k'}; \mathcal{N}^{k'}, \emptyset) \leftarrow \mathsf{Iso}(E_k, \{A_1^k\}; \mathcal{N}^k, \emptyset)\,,$$

than using $\mathsf{IsoDetour}$. Note again that A_1^k does not reveal any information about A_1 because of the difficulty of solving the Decisional Point Preimage Problem 3.

Final step. Alice's final step is the computation of the shared secret. As discussed in Section 4, this step needs to be computed locally. It involves the computation of the kernel generator and then of the final isogeny.

Cost. We establish the total cost of an n-party key agreement protocol. Let $d \in \{0, 1\}$ again distinguish if we have a delegation-friendly prime ($d = 1$) or not ($d = 0$) at our disposal. The public-key is computed using $2 - d$ invocations of $\mathsf{IsoDetour}$ with $\mu = 0$. The $n-2$ intermediate computations can then each be delegated using Iso with $\mu = 0$. The final step is then computed locally at the cost of $\mathsf{S}(\tau_A) + \mathsf{A} + \mathsf{I}(\tau_A, 0)$. Since after the public-key computation, Alice does not need to hide any points in either of the steps, she can simply perform all of these computations on Montgomery curves, reducing her computational and communication cost. We find the total cost of

$$T_{n\mathrm{PDH}}(d, \tau_A) = (2-d)T_{\mathsf{IsoDet}}(0) + (n-1)(\mathsf{S}(\tau_A) + \mathsf{A}) + \mathsf{I}(\tau_A, 0)\,,$$

under both the HBC and OMTUP assumptions.[9] In the local version of the protocol, Alice has to transport $2(n-k)$ points in round k, and compute the map of A given her generators on each curve except the first. We find

$$\alpha_{n\mathrm{PDH}}(d, \tau_A) = \frac{(2-d)T_{\mathsf{IsoDetour}}(0) + (n-1)(\mathsf{S}(\tau_A) + \mathsf{A}) + \mathsf{I}(\tau_A, 0)}{n(\mathsf{I}(\tau_A, n-1) + \mathsf{S}(\tau_A) + \mathsf{A})}.$$

Figure 6 shows the evolution of the cost reduction for $p434$ in terms of n for the cases with and without delegation-friendly primes and compares our theoretical estimates and benchmarks for the 2-party case ($d = 1$). Table 1 summarizes the communication costs for different n.

Remark 2. Note that the computational and communication cost established throughout this section also apply to the delegation of isogeny-based public-key encryption [18] and key encapsulation [39] as the steps of these protocols are the same (up to some negligible computations) as (2-party) SIDH.

[9] $T_{\mathsf{IsoDet}}(0)$ denotes a placeholder for either $T^{\mathrm{HBC}}_{\mathsf{IsoDet}}(\mu = 0, \tau_A)$ or $T^{\mathrm{OMTUP}}_{\mathsf{IsoDet}}(\mu = 0, t)$ of Section 4 depending on the underlying assumption.

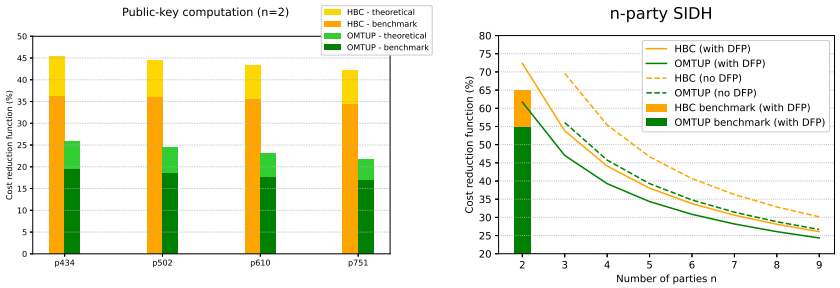

Fig. 6. Cost reduction functions in the HBC and OMTUP assumptions. The left figure compares theoretical costs and benchmarks for delegating 2-party public-key computations for different security levels. The discrepancy between these costs is mainly due to the computational overhead of local isogeny computations, which becomes less important for higher degree isogenies, since the cost of isogeny computation itself increases. The figure on the right shows the theoretical cost of n-party key agreement protocols for different n with and without a delegation-friendly prime in the case of $p434$. The case $n = 2$ further includes benchmarks. We see that the gain for the delegator increases with the security level and with the number of parties n.

5.2 Identification protocols and signatures

In this section, we establish the costs of identification protocols and signature schemes. We assume the public key (E_A, P_B^A, Q_B^A) to be precomputed as it is directly related to the identity of the prover.

Zero-knowledge proof of identity. We show how the ZKPI-protocol from [18] can be delegated. In every round of the protocol, the prover needs to compute the isogenies $\beta : E \to E_B$, $\beta' : E_A \to E_{AB}$ and the map A^B of the prover's secret. This can be done by delegating

$$(E_B; P_A^B, aQ_A^B) \leftarrow \mathsf{Iso}(E, \{P_B, (b, Q_B)\}; \{P_A\}, \{(a, Q_A)\}),$$
$$(E_{AB}; \emptyset, \emptyset) \leftarrow \mathsf{Iso}(E_A, \{P_B^A, (b, Q_B^A)\}; \emptyset, \emptyset).$$

Depending on the challenge, the response is either b or $A^B = P_A^B + aQ_A^B$ for $c = 0, 1$, respectively. If $c = 0$, the verifier delegates

$$(E_B; \emptyset, \emptyset) \leftarrow \mathsf{Iso}(E; \{P_B, (b, Q_B)\}; \emptyset, \emptyset), \quad (E_{AB}; \emptyset, \emptyset) \leftarrow \mathsf{Iso}(E_A; \{P_B^A, (b, Q_B^A)\}, \emptyset, \emptyset),$$

otherwise $(E_{AB}; \emptyset, \emptyset) \leftarrow \mathsf{Iso}(E_B, \{A^B\}; \emptyset, \emptyset)$.

Signature schemes. The delegation procedure of the signature schemes in [23] based on this identification scheme is completely analogous, i.e. for each of the commitments, the prover and/or verifier proceed exactly as in the identification protocol. The delegator further needs to compute hash-functions, but we assume that these have negligible cost (or are delegated with other schemes).

Table 1. Upload and Download costs (in kB per server) of delegating the n-party key agreement protocols in the HBC and OMTUP assumptions. We distinguish the cases with and without a delegation-friendly prime. The cost is given by the inputs and outputs within the three rounds of IsoDetour, assuming the initial E and its torsion group generators are known by the servers. We note that the kernel generator $\hat{K}^{AK}$ in Figure 4 is computed locally and we thus have $Z \neq 1$, which increases the upload cost. In the intermediate steps, Alice has to transport $2(n-k)$ unprotected points. Since the final step is computed locally, no communication costs apply. Therefore, the communication for $n = 2$ is the same as the communication needed to delegate the public key computation.

		no DFP				DFP			
		$p434$		$p751$		$p434$		$p751$	
		HBC	OMT	HBC	OMT	HBC	OMT	HBC	OMT
$n=2$	Upload	–	–	–	–	1.30	2.83	2.25	4.90
	Download	–	–	–	–	1.80	4.86	3.12	8.43
$n=3$	Upload	3.95	7.68	6.84	13.32	2.24	4.11	3.88	7.12
	Download	5.18	12.58	8.98	21.81	2.75	6.45	4.77	11.18
$n=4$	Upload	5.53	10.02	9.58	17.37	3.40	5.64	5.89	9.78
	Download	6.98	15.65	12.1	27.13	3.91	8.25	6.78	14.3

Remark 3. We note that an alternative ID protocol to [18] has recently been proposed in [17]. This scheme is quite similar, except that an $E_B[\tau_A]$ basis needs to be deterministically generated using an algorithm called CanonicalBasis. We can delegate this newer scheme in exactly the same fashion as the one presented here, except that we have to add the execution of CanonicalBasis to the advertised server functionality. Since the algorithm is deterministic, we only have to compare the output of both servers in the OMTUP assumption, in order to verify that the output is correct. Note that the download communication cost is increased by these extra points.

Cost. Following the discussion from Section 3.2, since A^B might be used as an unprotected input by the verifier, we have to choose $tN \geq \lambda$, so the cost for the prover becomes $T_{\text{OMTUP}}(1, N/\lambda)$ in the OMTUP and $T_{\text{HBC}}(1, \tau_A)$ in the HBC assumption. For both cases, we get the cost reduction functions

$$\alpha_{\mathsf{ZKPI.P}}(\tau_B) = \frac{T(1)}{2(\mathsf{S}(\tau_B) + \mathsf{A}) + \mathsf{I}(\tau_B, 1) + \mathsf{I}(\tau_B, 0)}\,, \quad \alpha_{\mathsf{ZKPI.V}} = O(1)\,.$$

Figure 7 shows theoretical estimates and benchmarked results for ZKPI-delegation by the prover. We summarize the communication costs in Table 2.

6 Conclusion and future work

In this work, we presented two outsource-secure delegation schemes, Iso and IsoDetour, under the *one honest-but-curious* (HBC) and *one-malicious version*

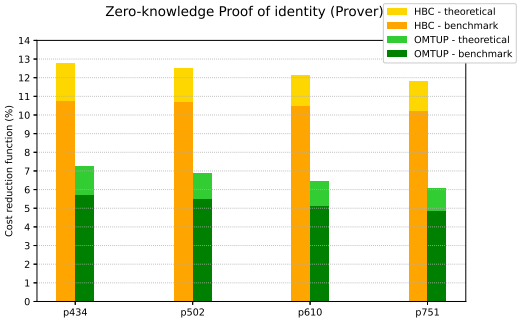

Fig. 7. Theoretical and benchmarked cost reduction function of the prover delegating zero-knowledge proofs of identity in the HBC and OMTUP assumptions. The theoretical predictions again underestimate the cost reduction via delegation, due to the overhead in isogeny computations. The discrepancy is higher this time higher than in Figure 6 due to the much lower cost for the delegator. Again, the gain increases with higher security.

of a two untrusted program (OMTUP) models of [25]. Our delegation algorithms can be used as a toolbox to delegate common isogeny-based cryptographic protocols in a secure and verifiable manner. Our approach reduces the cost of the zero-knowledge proof of identity from [27] as well as the related signature schemes from [23] to about 11% of the original cost in the HBC case and 6% in the OMTUP case. While the cost of n-party key-exchange delegation strongly decreases with increasing n, the case $n = 2$ only reaches a reduction to about 65% of the original cost. It is of substantial interest to further reduce this number in order to make e.g. the standardization candidate SIKE efficiently delegatable. While we were able to reduce the public-key generation step in the SIDH setting to about 35% and 20% of the original cost in the HBC and OMTUP cases, respectively, the main open question in these protocols remains how to efficiently delegate the computation of an isogeny where both the kernel and codomain curve are hidden from the servers. We leave it open to apply the proposed delegation algorithms to other interesting isogeny-based schemes over $\mathbb{F}_{p^2}$. We further note that any protocol that does not need hiding of data is virtually free to delegate. Examples include hashing functions with unprotected messages [10,21] and the verifiable delay function proposed in [20].

We generally find, that while HBC has a much cheaper communication cost and is fully verifiable, our OMTUP implementations result in lower computational cost for the delegator. Further, in all the schemes of Section 5, OMTUP has a very high verifiability, close to 1. It would be interesting to see, if other server assumptions are possible in the isogeny framework, especially using only malicious servers, such as the *two-untrusted program* (TUP) or *one-untrusted program* (OUP) models introduced in [25].

Table 2. Upload and Download costs (in B per server) of delegating the zero-knowledge proof of identity in the HBC and OMTUP assumptions, as well as for the verifier. The cost for the verifier is averaged over both challenge scenarios. We assume that the starting curve E and the associated generators are known by the servers. In the case of the prover, we further assume that its public key E_A and associated generators are also known to the servers. We also assume that the ephemeral parameter b has to be transmitted only once. Since the OMTUP case reduces to simple comparison operations for the verifier, these can also be done on Montgomery curves, saving some of the communication.

	$p434$			$p751$		
	HBC	OMTUP	Ver.	HBC	OMTUP	Ver.
Upload	54	189	298	94	328	516
Download	433	1516	162	751	2628	282

For future work, it is also of interest to construct delegation algorithms for other isogeny-based schemes, such as CSIDH [9] and CSI-FiSh [6] over $\mathbb{F}_p$, or the endomorphism ring based signature protocol of [23] as well as SQI-Sign [19].

Acknowledgments. The authors would like to thank Frederik Vercauteren for discussions and valuable feedback during this work. We would also like Jason LeGrow for valuable discussions concerning the isogeny cost function. This work was supported in part by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101020788 - Adv-ERC-ISOCRYPT), the Research Council KU Leuven grant C14/18/067, and by CyberSecurity Research Flanders with reference number VR20192203.

References

1. Alagic, G., Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q., Kelsey, J., Liu, Y.K., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., Smith-Tone, D.: Status report on the second round of the NIST post-quantum cryptography standardization process. NISTIR 8309 (07/2020 2020). https://doi.org/https://doi.org/10.6028/NIST.IR.8309
2. Azarderakhsh, R., Campagna, M., Costello, C., De Feo, L., Hess, B., Jalali, A., Jao, D., Koziel, B., LaMacchia, B., Longa, P., et al.: Supersingular isogeny key encapsulation. Submission to the NIST Post-Quantum Standardization project (2017)
3. Azarderakhsh, R., Jalali, A., Jao, D., Soukharev, V.: Practical supersingular isogeny group key agreement. IACR Cryptol. ePrint Arch. **2019**, 330 (2019)
4. Bernstein, D.J., Birkner, P., Joye, M., Lange, T., Peters, C.: Twisted Edwards curves. In: International Conference on Cryptology in Africa. pp. 389–405. Springer (2008)

5. Bernstein, D., Lange, T.: Explicit-formulas database. `https://www.hyperelliptic.org/EFD` (accessed 5th May 2021)
6. Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-FiSh: Efficient isogeny based signatures through class group computations. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 227–247. Springer (2019)
7. Bouvier, C., Imbert, L.: An alternative approach for SIDH arithmetic. IACR Cryptol. ePrint Arch. **2020** (2020)
8. Castryck, W., Galbraith, S.D., Farashahi, R.R.: Efficient arithmetic on elliptic curves using a mixed Edwards-Montgomery representation. IACR Cryptol. ePrint Arch. **2008**, 218 (2008)
9. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: an efficient post-quantum commutative group action. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 395–427. Springer (2018)
10. Charles, D.X., Lauter, K.E., Goren, E.Z.: Cryptographic hash functions from expander graphs. Journal of Cryptology **22**(1), 93–113 (2009)
11. Chevalier, C., Laguillaumie, F., Vergnaud, D.: Privately outsourcing exponentiation to a single server: cryptanalysis and optimal constructions. In: European Symposium on Research in Computer Security. pp. 261–278. Springer (2016)
12. Costello, C., Longa, P., Naehrig, M., Renes, J., Virdia, F.: Improved classical cryptanalysis of sike in practice. In: Kiayias, A.(ed.), Public-Key Cryptography–PKC 2020: 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4–7, 2020, Proceedings, Part II. pp. 505–534. Cham: Springer International Publishing (2020)
13. Costello, C.: B-SIDH: Supersingular isogeny Diffie-Hellman using twisted torsion. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 440–463. Springer (2020)
14. Costello, C., Hisil, H.: A simple and compact algorithm for SIDH with arbitrary degree isogenies. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 303–329. Springer (2017)
15. Costello, C., Smith, B.: Montgomery curves and their arithmetic. Journal of Cryptographic Engineering **8**(3), 227–240 (2018)
16. De Feo, L.: Mathematics of isogeny based cryptography. arXiv preprint arXiv:1711.04062 (2017)
17. De Feo, L., Dobson, S., Galbraith, S., Zobernig, L.: SIDH proof of knowledge. IACR Cryptol. ePrint Arch. **2021**, 1023 (2021)
18. De Feo, L., Jao, D., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Journal of Mathematical Cryptology **8**(3), 209–247 (2014)
19. 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)
20. De Feo, L., Masson, S., Petit, C., Sanso, A.: Verifiable delay functions from supersingular isogenies and pairings. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 248–277. Springer (2019)
21. Doliskani, J., Pereira, G.C., Barreto, P.S.: Faster cryptographic hash function from supersingular isogeny graphs. IACR Cryptol. ePrint Arch. **2017**, 1202 (2017)

22. Furukawa, S., Kunihiro, N., Takashima, K.: Multi-party key exchange protocols from supersingular isogenies. In: 2018 International Symposium on Information Theory and Its Applications (ISITA). pp. 208–212. IEEE (2018)
23. Galbraith, S.D., Petit, C., Silva, J.: Identification protocols and signature schemes based on supersingular isogeny problems. Journal of Cryptology **33**(1), 130–175 (2020)
24. Hisil, H., Wong, K.K.H., Carter, G., Dawson, E.: Twisted Edwards curves revisited. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 326–343. Springer (2008)
25. Hohenberger, S., Lysyanskaya, A.: How to securely outsource cryptographic computations. In: Theory of Cryptography Conference. pp. 264–282. Springer (2005)
26. Icart, T.: How to hash into elliptic curves. In: Annual International Cryptology Conference. pp. 303–316. Springer (2009)
27. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: International Workshop on Post-Quantum Cryptography. pp. 19–34. Springer (2011)
28. Jaques, S., Schanck, J.M.: Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE. In: Annual International Cryptology Conference. pp. 32–61. Springer (2019)
29. Kiraz, M.S., Uzunkol, O.: Efficient and verifiable algorithms for secure outsourcing of cryptographic computations. International Journal of Information Security **15**(5), 519–537 (2016)
30. Meyer, M., Reith, S., Campos, F.: On hybrid SIDH schemes using Edwards and Montgomery curve arithmetic. IACR Cryptol. ePrint Arch. **2017**, 1213 (2017)
31. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Mathematics of computation **48**(177), 243–264 (1987)
32. NIST: NIST reveals 26 algorithms advancing to the post-quantum crypto 'semifinals' (2019), https://www.nist.gov/news-events/news/2019/01/nist-reveals-26-algorithms-advancing-post-quantum-crypto-semifinals
33. NIST: NIST post-quantum cryptography PQC (2020), https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions
34. Pedersen, R., Uzunkol, O.: Secure delegation of isogeny computations and cryptographic applications. In: Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop. pp. 29–42 (2019)
35. Pedersen, R., Uzunkol, O.: Delegating supersingular isogenies over $\mathbb{F}_{p^2}$ with cryptographic applications. IACR Cryptol. ePrint Arch. **2021**, 506 (2021)
36. Petit, C.: Faster algorithms for isogeny problems using torsion point images. In: Advances in Cryptology – ASIACRYPT 2017. pp. 330–353 (2017)
37. de Quehen, V., Kutas, P., Leonardi, C., Martindale, C., Panny, L., Petit, C., Stange, K.E.: Improved torsion point attacks on sidh variants. arXiv preprint arXiv:2005.14681 (2020)
38. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science. pp. 124–134 (1994)
39. SIKE: Supersingular Isogeny Key Encapsulation (2018), https://sike.org
40. Silverman, J.H.: The arithmetic of elliptic curves, vol. 106. Springer Science & Business Media (2009)