# Revocable Hierarchical Identity-Based Authenticated Key Exchange

Yuki Okano[1], Junichi Tomida[1], Akira Nagai[1], Kazuki Yoneyama[2], Atsushi Fujioka[3], and Koutarou Suzuki[4]

[1] NTT Social Informatics Laboratories, Tokyo, Japan
{yuki.okano.te, junichi.tomida.vw, akira.nagai.td}@hco.ntt.co.jp
[2] Ibaraki University, Ibaraki, Japan
kazuki.yoneyama.sec@vc.ibaraki.ac.jp
[3] Kanagawa University, Kanagawa, Japan
fujioka@kanagawa-u.ac.jp
[4] Toyohashi University of Technology, Aichi, Japan
suzuki@cs.tut.ac.jp

**Abstract.** Identity-based authenticated key exchange (IB-AKE) would be considered to have an advantage in the sense that it does not require certificate management. However, IB-AKE has not both a key delegation functionality and a key revocation functionality. This leaves the problem of the burden to the private key generator when there are a large number of parties in the system, and the problem of the lack of a clear way to eliminate dishonest parties from the system. In this paper, we propose a new authentication mechanism called revocable hierarchical IB-AKE (RHIB-AKE), which can decentralize key generation and revocation performed by a PKG. We also propose a generic construction of RHIB-AKE from a revocable hierarchical identity-based key encapsulation mechanism (RHIB-KEM). We obtain the first RHIB-AKE schemes from pairings or lattices by our generic construction since RHIB-KEM is known to be constructed from them. For security, we show that our scheme resists against leakage of all combinations of master, static, current, and ephemeral secret keys except ones trivially break the security.

**Keywords:** Revocable hierarchical identity-based authenticated key exchange · rhid-eCK model · Revocable hierarchical identity-based key encapsulation mechanism.

## 1 Introduction

Identity-based authenticated key exchange (IB-AKE) [27, 5, 16, 10, 34] is a cryptographic protocol and enables two parties to share a common session key via unauthenticated networks, where each party is identified with information called an identity (ID). IB-AKE succeeds in removing the management of certificates provided by a certificate authority in the ordinary public key infrastructure (PKI) based setting. Instead of that, it is necessary that an additional single authority called a private key generator (PKG) exists who extracts the secret

key of each party corresponding to the identity of that party from its own secret key called the master secret key. On the contrary, it causes the problem of the burden to the authority if the number of parties belonging to the system is huge.

Hierarchical IB-AKE (HIB-AKE) [12, 36, 17] is one extension of IB-AKE and solves the above problem. HIB-AKE supports a key delegation functionality. Also, HIB-AKE only requires each party to trust the master public key generated by a single PKG when compared to a simple independent multiple PKG setting, in which each party must obtain the master public key of the PKG on another party without PKI. However, since HIB-AKE does not support a secret key revocation mechanism, there remains a problem with how to eliminate dishonest parties from the HIB-AKE system. Only this point is inferior to the PKI setting. Therefore, an efficient revocation mechanism is needed in the HIB-AKE system, assuming the case where the number of parties increases. It is natural to extend HIB-AKE to revocable HIB-AKE (RHIB-AKE).

As for identity-based encryption (IBE) systems, Boneh and Franklin [3] proposed a naive revocation method that requires the PKG to periodically regenerate secret keys of all parties. This method is not practical if the number of parties increases too much. Boldyreva et al. [2] use the complete subtree method to propose the first revocable IBE (RIBE) where the PKG workload is logarithmic of the number of parties in the system. In RIBE systems, ciphertexts cannot be decrypted directly from the secret key corresponding to the party's ID. The party needs a decryption key for each time period. PKG broadcasts key update information over a public channel every time period, and only non-revoked parties can obtain their own decryption key from key update information and their own secret key.

Seo and Emura [29] used Boldyreva's idea of RIBE to propose a new revocable hierarchical identity-based encryption (RHIBE) that is an extension of hierarchical identity-based encryption. RHIBE has both a key delegation functionality and a key revocation functionality. In other words, every non-leaf party in the hierarchical structure can revoke secret keys of low-level parties. Therefore, in HIB-AKE, it is a natural direction to achieve a key revocation with Boldyreva's idea of RIBE. However, it is not trivial to construct a secure RHIB-AKE satisfying desirable security requirements for AKE such as forward secrecy and security against master secret key leakage.

### 1.1 Related Works

Many RHIBE schemes have been proposed [29–31, 24, 19, 35, 8, 23, 33]. Each party has an ID corresponding to a node of a tree. A party of depth-$\ell$ in the tree has a level-$\ell$ ID, $(ID_1, ..., ID_\ell)$ and the parent node with a level-$(\ell - 1)$ ID being $(ID_1, ..., ID_{\ell-1})$. In the beginning, the party in the root node (or level-0) called PKG generates a master public key (MPK) and master secret key (MSK). A party that is not in the leaf nodes can generate static secret keys (SSKs) for lower-level parties from its own SSK (or MSK in case of PKG)[5]. The party can

[5] This means that all parties except those in leaf nodes can take a role as PKG. For convenience, however, we call only the party in the root node PKG in what follows.

also revoke lower-level parties according to time period by distributing (non-secret) key update information (KU) that the party generates from its SSK and a revocation list. Every party can generate a current secret key (CSK) for time period $T$ from its own SSK and KU for $T$ if the party is not revoked at $T$.[6] A valid ciphertext is associated with an ID, $ID$ and a time period $T$, which can be decrypted by only CSK for $ID$ and $T$.

### 1.2 Our Contribution

We put forward a new AKE mechanism named *Revocable Hierarchical Identity-Based AKE* (RHIB-AKE). RHIB-AKE allows for the revocation of secret keys that have been compromised or expired and having multiple PKGs by only trusting a single master public key generated by a single PKG. Formally, our contribution is two-fold: first, we introduce the concept of RHIB-AKE and formally define it; second, we present a generic methodology to construct an RHIB-AKE scheme. By our generic construction, we obtain the first RHIB-AKE schemes from pairings and lattices.

**RHIB-AKE Model.** In RHIB-AKE, the hierarchical ID structure and the key delegation/revocation mechanisms are the same as those in RHIBE. Instead of encryption and decryption in RHIBE, parties run an AKE protocol to create a secure channel. Concretely, in time period $T$, every pair of parties who have their CSK for $T$, that is, who are not revoked at $T$ can run an AKE protocol to compute a shared key called session key (SK). In the protocol, each party generates an ephemeral secret key (ESK) and computes an SK from the ESK and the CSK for $T$.

Note that naively sending a session key via RHIBE does not satisfy desirable security requirements for AKE such as forward secrecy and security against MSK leakage. To capture these threats, we extend the id-eCK model [16], which is one of the strongest security models for IB-AKE, to the RHIB-AKE setting and formally define a security model for RHIB-AKE called the *rhid-eCK model*. Similar to the id-eCK model, we consider maximum exposure of secret information in the rhid-eCK model to capture the following threats. The secret information in RHIB-AKE consists of the MSK, SSKs, CSKs, ESKs, and SKs. The MSK may be exposed when the PKG is corrupted. SSKs may be revealed if a party stores them in an insecure storage while they must be generated in a tamper-proof module. CSKs may be revealed if they remain in memory without being securely erased before they are updated, and are read out by an adversary. ESKs may be guessed to an adversary if a pseudo-random number generator implemented in a system is poor. The SKs may be revealed if they are misused to encrypt messages with a vulnerable scheme.

[6] An SSK and a CSK are the names in the context of RHIB-AKE as defined in this paper, and in the context of RHIBE they are actually called a secret key and a decryption key, respectively.

**RHIB-AKE Construction.** We propose the first RHIB-AKE schemes that are secure in the rhid-eCK security model from pairings and lattices. Our starting point is the work by Fujioka et al. [11], who proposed a generic construction of a $\mathrm{CK}^+$ secure AKE protocol from IND-CCA (indistinguishability against chosen ciphertext attack) secure public-key encryption[7]. Our observation is that while the $\mathrm{CK}^+$ and the eCK model [22] are incomparable, their technique is applicable to the eCK model. Thus, our idea is that if we have an IND-CCA secure RHIBE scheme, we can generically construct a rhid-eCK secure RHIB-AKE scheme analogously to the work by Fujioka et al. Following the idea, we show that such a generic construction is possible and prove that if the underlying RHIBE is IND-CCA secure then the resulting RHIB-AKE is rhid-eCK secure.

A problem here is that no IND-CCA secure RHIBE scheme is known. Fortunately, we can easily achieve one by applying the Fujisaki-Okamoto transformation [13] to an IND-CPA secure RHIBE scheme, which can be constructed from pairings [8, 33] or lattices [19, 35]. Note that the BCHK transformation [18] is not applicable to RHIBE (see [18] for details, where they use the BCHK transformation in a non-black box manner). Thus, starting from these works, we can achieve RHIB-AKE schemes from pairings and lattices through our generic construction, where the latter is considered to be secure against quantum adversaries.

### 1.3 Organization

The rest of this paper is organized as follows. We introduce the definition of RHIB-AKE and its security model in Section 2. We propose our RHIB-AKE scheme in Section 3. We then give some instantiations of our RHIB-AKE scheme in Section 4. The conclusion and future works are finally given in Section 5.

## 2 Revocable Hierarchical Identity-based Authenticated Key Exchange

In this section, we provide the definitions of an RHIB-AKE scheme and the rhid-eCK security model. Before describing the details, we give some notations.

**Notations.** $\mathbb{N}$ denotes the set of natural numbers. If $S$ is a set, $s \in_R S$ denotes that $s$ is chosen uniformly at random from $S$, and $|S|$ the number of elements in $S$. Let $a||b$ be the concatenation of $a$ and $b$. Let $\lceil x \rceil$ denote the smallest integer greater than or equal to $x$. Let $U_i$ be a party and $ID_i = (ID_{i,1}, \ldots, ID_{i,t})$ be the associated identity where $ID_{i,j} \in \{0,1\}^*$. An integer $|ID_i| := t$ denotes the hierarchical level of identity $ID_i = (ID_{i,1}, \ldots, ID_{i,t})$. The level-0 party with the special identity "$\mathsf{pkg}$" is called the PKG. The order $ID'_i \succ ID_i$ between two identities

[7] Precisely, they use IND-CCA secure public-key encapsulation mechanism (PK-KEM) and IND-CPA (indistinguishability against chosen plaintext attack) secure PK-KEM as building blocks. For ease of exposition, we employ IND-CCA secure PKE here since it implies both IND-CCA secure PK-KEM and IND-CPA secure PK-KEM.

holds if $ID'_i = (ID_{i,1}, \ldots, ID_{i,k_1})$ and $ID_i = (ID_{i,1}, \ldots, ID_{i,k_1}, \ldots, ID_{i,k_2})$ for $1 \leq k_1 \leq k_2$. Especially, $ID'_i$ is the parent of the identity $ID_i$, i.e., $ID'_i = pa(ID_i)$ if $k_1 + 1 = k_2$. Let $\mathcal{I}_{ID_i} := \{ID''_i \mid pa(ID''_i) = ID_i\}$ be a set of identities whose parent is $ID_i$ and $\mathcal{I}_{\mathsf{pkg}} := \{ID''_i \mid pa(ID''_i) = \mathsf{pkg}\}$ be a set of identities whose parent is the PKG.

### 2.1 RHIB-AKE scheme

In this subsection, we introduce a syntax of RHIB-AKE. All parties are modeled as a probabilistic polynomial-time (PPT) Turing machine. RHIB-AKE scheme $\Pi$ consists of the following five algorithms and a key exchange protocol.

$\mathsf{ParGen}(1^\kappa, L) \to (MSK, MPK)$ : It takes a security parameter, $1^\kappa$, and the maximum depth of the hierarchy, $L \in \mathbb{N}$, which is a polynomial of $\kappa$ as input, and outputs a master secret key, $MSK$, and a master public key, $MPK$. The PKG runs this algorithm.

$\mathsf{SSKGen}(MPK, ID'_i, ID_i, SSK_{ID'_i}) \to (SSK_{ID_i}, SSK'_{ID'_i})$ : It takes $MPK$, two identities $ID'_i = pa(ID_i)$ and $ID_i = (ID_{i,1}, \ldots, ID_{i,t})$, and a static secret key, $SSK_{ID'_i}$, corresponding to $ID'_i$, and outputs $SSK_{ID_i}$, corresponding to $ID_i$ and the updated $SSK'_{ID'_i}$ corresponding to $ID'_i$. The party associated with $ID'_i$ runs this algorithm. If the party is the PKG, this algorithm takes the $MPK$, identity $ID_i$, and the $MSK$, and outputs $SSK_{ID_i}$, corresponding to $ID_i$ and the updated $MSK'$, i.e., $\mathsf{SSKGen}(MPK, ID_i, MSK) \to (SSK_{ID_i}, MSK')$.

$\mathsf{Rev}(MPK, T, RL_{ID_i,T-1}, Add) \to RL_{ID_i,T}$ : It takes $MPK$, a time period, $T$, a revocation list, $RL_{ID_i,T-1}$, and a set of identities, $Add \subset \mathcal{I}_{ID_i}$, that are added to the revocation list, and outputs the updated revocation list, $RL_{ID_i,T}$. The party associated with $ID_i$ runs this algorithm. Note that $RL_{ID_i,0} = \emptyset$.

$\mathsf{KUGen}(MPK, T, SSK_{ID_i}, RL_{ID_i,T}, ku_{pa(ID_i),T}) \to (ku_{ID_i,T}, SSK'_{ID_i})$ : It takes $MPK$, a time period, $T \in \mathbb{N}$, $SSK_{ID_i}$ corresponding to $ID_i$, a revocation list, $RL_{ID_i,T} \subset \mathcal{I}_{ID_i}$, and parent's key update information, $ku_{pa(ID_i),T}$, and outputs $ku_{ID_i,T}$, corresponding to $ID_i$ and the updated $SSK'_{ID_i}$. The party associated with $ID_i$ runs this algorithm. If the party is the PKG, this algorithm takes $MPK$, a time period, $T$, and $MSK$, and a revocation list, $RL_{\mathsf{pkg},T} \subset \mathcal{I}_{\mathsf{pkg}}$, and outputs $ku_{\mathsf{pkg},T}$, and the updated $MSK'$, i.e., $\mathsf{KUGen}(MPK, T, MSK, RL_{\mathsf{pkg},T}) \to (ku_{\mathsf{pkg},T}, MSK')$.

$\mathsf{CSKGen}(MPK, SSK_{ID_i}, ku_{pa(ID_i),T}) \to CSK_{ID_i,T} \ / \ \perp$ : It takes $MPK$, $SSK_{ID_i}$, corresponding to $ID_i$, and parent's $ku_{pa(ID_i),T}$, outputs a current secret key, $CSK_{ID_i,T}$, or $\perp$ indicating that $ID_i$ or some $ID'_i$ such that $ID'_i \succ ID_i$ have been revoked.

*Key Exchange Protocol.* Parties $U_A$ and $U_B$ can share a session key in time period $T \in \mathbb{N}$ by performing the two-pass protocol by using each of the following four algorithms in turn. $U_A$ has $SSK_A$, and $CSK_{A,T}$, corresponding to $ID_A$, and $U_B$ has $SSK_B$, and $CSK_{B,T}$, corresponding to $ID_B$.

$\mathsf{InitEK}(MPK, T, ID_A, SSK_A, CSK_{A,T}, ID_B) \rightarrow (ESK_A, EPK_A)$ : Party $U_A$ computes ephemeral keys by algorithm $\mathsf{InitEK}$ which takes the MPK, $MPK$, the time period, $T$, $ID_A$, $SSK_A$, $CSK_{A,T}$, $ID_B$, and outputs an ephemeral secret key, $ESK_A$, and an ephemeral public key, $EPK_A$. $U_A$ sends $EPK_A$ to party $U_B$.

$\mathsf{ResEK}(MPK, T, ID_B, SSK_B, CSK_{B,T}, ID_A, EPK_A) \rightarrow (ESK_B, EPK_B)$ : Party $U_B$ computes ephemeral keys by algorithm $\mathsf{ResEK}$ which takes $MPK$, the time period, $T$, $ID_B$, $SSK_B$, $CSK_{B,T}$, $ID_A$, and $EPK_A$, and outputs $ESK_B$, and $EPK_B$. $U_B$ sends $EPK_B$ to $U_A$.

$\mathsf{ResSK}(MPK, T, ID_B, SSK_B, CSK_{B,T}, ID_A, ESK_B, EPK_B, EPK_A) \rightarrow SK$ : Party $U_B$ computes a session key by algorithm $\mathsf{ResSK}$ which takes $MPK$, the time period, $T$, $ID_B$, $SSK_B$, $CSK_{B,T}$, $ID_A$, $ESK_B$, $EPK_B$, and $EPK_A$, and outputs a session key, $SK$.

$\mathsf{InitSK}(MPK, T, ID_A, SSK_A, CSK_{A,T}, ID_B, ESK_A, EPK_A, EPK_B) \rightarrow SK$ : Party $U_A$ computes a session key by algorithm $\mathsf{InitSK}$ which takes $MPK$, the time period, $T$, $ID_A$, $SSK_A$, $CSK_{A,T}$, $ID_B$, $ESK_A$, $EPK_A$, and $EPK_B$, and outputs $SK$.

**Session.** An invocation of a protocol is called a *session*. For a party, $U_A$, a session is activated via an incoming message of the forms, $(\Pi, \mathcal{I}, T, ID_A, ID_B)$ or $(\Pi, \mathcal{R}, T, ID_A, ID_B, EPK_B)$, where $\Pi$ is a protocol identifier. If $U_A$ is activated with $(\Pi, \mathcal{I}, T, ID_A, ID_B)$ (resp. $(\Pi, \mathcal{R}, T, ID_A, ID_B, EPK_B)$), then $U_A$ is the *initiator* (resp. *responder*). After activation, $U_A$ switches the second element (or role) of the incoming message, appends an EPK, $EPK_A$, to it, and sends it as an outgoing response. If $U_A$ is the responder, $U_A$ computes a session key. If $U_A$ is the initiator, $U_A$ that has been successfully activated via $(\Pi, \mathcal{I}, T, ID_A, ID_B)$ can be further activated via $(\Pi, \mathcal{I}, T, ID_A, ID_B, EPK_A, EPK_B)$ to compute a session key.

An initiator $U_A$ identifies its session via $(\Pi, \mathcal{I}, ID_A, ID_B, EPK_A)$ or $(\Pi, \mathcal{I}, ID_A, ID_B, EPK_A, EPK_B)$. If $U_A$ is a responder, the session is identified via $(\Pi, \mathcal{R}, ID_A, ID_B, EPK_B, EPK_A)$. The *matching session* of session $\mathsf{sid} = (\Pi, \mathcal{I}, ID_A, ID_B, EPK_A, EPK_B)$ is session $\mathsf{sid}' = (\Pi, \mathcal{R}, ID_B, ID_A, EPK_A, EPK_B)$ and vice versa. We say that $U_A$ is the *owner* of session $\mathsf{sid}$ if the third element of $\mathsf{sid}$ is $ID_A$, $U_B$ is the *peer* of session $\mathsf{sid}$ if the fourth element of $\mathsf{sid}$ is $ID_B$, and a session is *completed* if its owner computes a session key.

### 2.2 Security Model

In this subsection, we define the rhid-eCK security model. This security model, like the definition of the id-eCK model, consists of the definition of an adversary and its capabilities, the definition of freshness, and the security experiment.

This security model is different from the id-eCK model in the following points: 1) an adversary also has access to $\mathsf{CSKReveal}$ and $\mathsf{Revoke}$ queries, the former capturing the reveal of CSKs of parties and the latter capturing system-wide revocation list updates and time period increments, and 2) the freshness condition is modified based on the additional queries.

**Adversaries.** Adversary $\mathcal{A}$ is modeled as a PPT Turing machine. It controls all communications between parties including session activation by performing the following query, where $T_{cu}$ is the current time period initialized with 1.

- $\mathsf{Send}$(message): The input message has one of the following forms: $(\Pi, \mathcal{I}, T_{cu}, ID_A, ID_B)$, $(\Pi, \mathcal{R}, T_{cu}, ID_B, ID_A, EPK_A)$, or $(\Pi, \mathcal{I}, T_{cu}, ID_A, ID_B, EPK_A, EPK_B)$. Adversary $\mathcal{A}$ obtains the response from the party.

Note that adversary $\mathcal{A}$ does not control the communication for delivery of either SSK or KU between each party and its parent.

A party's private information is not accessible to the adversary. However, leakage of private information is captured via the following adversary queries, where $\mathcal{ID}$ is a set of identities which the SSK is already generated.

- $\mathsf{SKReveal}(\mathsf{sid})$: The adversary obtains the session key for session $\mathsf{sid}$, provided that the session is completed.
- $\mathsf{ESKReveal}(\mathsf{sid})$: The adversary obtains the ESK of the owner of session $\mathsf{sid}$.
- $\mathsf{SSKReveal}(ID_i)$: The adversary learns the SSK corresponding to identity $ID_i$ if $SSK_{ID_i}$ is already generated.
- $\mathsf{CSKReveal}(ID_i, T)$: The adversary learns the CSK corresponding to $ID_i$ and time period $T$ if $T \leq T_{cu}$ and $\mathsf{CSKGen}(MPK, SSK_{ID_i}, ku_{pa(ID_i),T}) = CSK_{ID_i,T} \neq \perp$.
- $\mathsf{MSKReveal}()$: The adversary learns the MSK of the system.
- $\mathsf{EstablishParty}(U_i, ID_i)$: This query allows the adversary to give an SSK corresponding to identity $ID_i$ on behalf of party $U_i$ if the SSK is not yet generated and the SSK corresponding to the parent identity of $ID_i$ is already generated. Otherwise, $\perp$ is returned. When $\perp$ is not returned, the adversary controls party $U_i$. If a party is established by this query, then we call party $U_i$ *dishonest*. If not, we call the party *honest*.
- $\mathsf{Revoke}(RL)$: If the following conditions are not satisfied for the set of identities $RL \subset \{ID \mid |ID| \geq 1\}$, then $\perp$ is returned.
  - For all $ID_i \in \mathcal{ID} \cup \{\mathsf{pkg}\}$, $RL_{ID_i,T_{cu}} \subset RL$.
  - For all $ID_i \in \mathcal{ID}$, $ID_i \in RL$ if $pa(ID_i) \in RL$.

  Otherwise, $T_{cu}$ is incremented as $T_{cu} \leftarrow T_{cu} + 1$. Then, PKG's revocation list is updated as $RL_{\mathsf{pkg},T_{cu}} \leftarrow \mathsf{Rev}(MPK, T_{cu}, RL_{\mathsf{pkg},T_{cu}-1}, RL \cap \mathcal{I}_{\mathsf{pkg}})$ and $(ku_{\mathsf{pkg},T_{cu}}, MSK') \leftarrow \mathsf{KUGen}(MPK, T_{cu}, MSK, RL_{\mathsf{pkg},T_{cu}})$ is computed. For any party $U_i$ with $ID_i \in \mathcal{ID}$, its revocation list is updated as $RL_{ID_i,T_{cu}} \leftarrow \mathsf{Rev}(MPK, T_{cu}, RL_{ID_i,T_{cu}-1}, RL \cap \mathcal{I}_{ID_i})$ and $(ku_{ID_i,T_{cu}}, SSK'_{ID_i}) \leftarrow \mathsf{KUGen}(MPK, T, SSK_{ID_i}, RL_{ID_i,T_{cu}}, ku_{pa(ID'_i),T_{cu}})$ is computed. The adversary obtains all key updated information $\{ku_{ID_i,T_{cu}}\}_{ID_i \in \mathcal{ID} \cup \{\mathsf{pkg}\}}$.

**Freshness.** Our security definition requires the notion of *freshness*. $\mathsf{sid}^* = (\Pi, \mathcal{I}, T^*, ID_A, ID_B, EPK_A, EPK_B)$ or $(\Pi, \mathcal{R}, T^*, ID_A, ID_B, EPK_B, EPK_A)$ be a completed session between honest party $U_A$ with identity $ID_A$ and $U_B$ with identity $ID_B$. Let $\overline{\mathsf{sid}^*}$ be the matching session of $\mathsf{sid}^*$ if the matching session exists. We define session $\mathsf{sid}^*$ to be fresh if none of the following conditions hold:

1. Adversary $\mathcal{A}$ issues $\mathsf{SKReveal}(\mathsf{sid}^*)$, or $\mathsf{SKReveal}(\overline{\mathsf{sid}^*})$ if $\overline{\mathsf{sid}^*}$ exists.
2. $\overline{\mathsf{sid}^*}$ exists and adversary $\mathcal{A}$ makes either of the following queries
   - both $\mathsf{ESKReveal}(\mathsf{sid}^*)$ and $\mathsf{SSKReveal}(ID)$ for identity $ID$ such that $ID \succ ID_A$ and $ID \notin RL_{pa(ID),T^*-1}$,
   - both $\mathsf{ESKReveal}(\overline{\mathsf{sid}^*})$ and $\mathsf{SSKReveal}(ID)$ for identity $ID$ such that $ID \succ ID_B$ and $ID \notin RL_{pa(ID),T^*-1}$,
   - both $\mathsf{CSKReveal}(ID_A, T^*)$ and $\mathsf{ESKReveal}(\overline{\mathsf{sid}^*})$, or
   - both $\mathsf{CSKReveal}(ID_B, T^*)$ and $\mathsf{ESKReveal}(\overline{\mathsf{sid}^*})$.
3. $\overline{\mathsf{sid}^*}$ does not exist and adversary $\mathcal{A}$ makes either of the following queries
   - both $\mathsf{ESKReveal}(\mathsf{sid}^*)$ and $\mathsf{SSKReveal}(ID)$ for identity $ID$ such that $ID \succ ID_A$ and $ID \notin RL_{pa(ID),T^*-1}$,
   - $\mathsf{SSKReveal}(ID)$ for $ID$ such that $ID \succ ID_B$ and $ID \notin RL_{pa(ID),T^*-1}$,
   - both $\mathsf{CSKReveal}(ID_A, T^*)$ and $\mathsf{ESKReveal}(\mathsf{sid}^*)$, or
   - $\mathsf{CSKReveal}(ID_B, T^*)$.

Note that if adversary $\mathcal{A}$ issues $\mathsf{MSKReveal}()$, we regard $\mathcal{A}$ as having issued $\mathsf{SSKReveal}(ID_A)$ and $\mathsf{SSKReveal}(ID_B)$.

**Security Experiment.** We consider the following security experiment. Initially, adversary $\mathcal{A}$ is given a set of honest parties and makes any sequence of the above queries. During the experiment, $\mathcal{A}$ makes the following query.

- $\mathsf{Test}(\mathsf{sid}^*)$: Session $\mathsf{sid}^*$ must be fresh. Select a bit $b \in_R \{0,1\}$ and return the session key held by $\mathsf{sid}^*$ if $b = 0$, and return a random key if $b = 1$.

The experiment continues until adversary $\mathcal{A}$ makes a guess $b'$. The adversary *wins* the game if the test session $\mathsf{sid}^*$ is still fresh and if $\mathcal{A}$'s guess is correct, i.e., $b' = b$. The advantage is defined as $\mathrm{Adv}_{\Pi}^{\text{RHIB-AKE}}(\mathcal{A}) = |\Pr[\mathcal{A} \text{ wins}] - 1/2|$. We define the security as follows.

**Definition 1.** *We say that RHIB-AKE scheme $\Pi$ is secure in the rhid-eCK model if the following conditions hold:*

1. *If two honest parties complete matching sessions, then, except with negligible probability, they both compute the same session key.*
2. *For any PPT adversary $\mathcal{A}$, $\mathrm{Adv}_{\Pi}^{\text{RHIB-AKE}}(\mathcal{A})$ is negligible.*

Moreover, we say that RHIB-AKE scheme $\Pi$ is selective secure in the rhid-eCK model, if $\mathcal{A}$ outputs $(T^*, ID_A, ID_B)$ at the beginning of the security experiment.

## 3 Our RHIB-AKE Scheme

In this section, we give a generic construction of selective rhid-eCK secure RHIB-AKE from key encapsulation mechanism (KEM) and revocable hierarchical identity-based KEM (RHIB-KEM). We define RHIB-KEM required for our construction. We show the instantiation of RHIB-KEM in Section 4.

### 3.1 Preliminaries

We recall the security definition for pseudorandom function (PRF), key derivation function (KDF), twisted PRF, and KEM.

**Basic Cryptographic Functions.** We give definitions of a pseudorandom function and a key derivation function [20, 11].

*Pseudorandom Function (PRF).* Let $\mathcal{F} = \{F : \mathcal{K}_\kappa \times D_\kappa \to R_\kappa\}$ be a function family with respect to key $k \in \mathcal{K}_\kappa$. Let $\mathcal{RF}$ be the set of all functions from $D_\kappa$ to $R_\kappa$. We say that $\mathcal{F}$ is a PRF (family) if for any PPT distinguisher $\mathcal{D}$ it holds that $\mathrm{Adv}^{prf}_{\mathcal{F},\mathcal{D}}(\kappa) = |\Pr[\mathcal{D}^{F_k(\cdot)}(1^\kappa) \to 1 \mid k \in_R \mathcal{K}_\kappa] - \Pr[\mathcal{D}^{f(\cdot)}(1^\kappa) \to 1 \mid f \in_R \mathcal{RF}]|$ is negligible for security parameter $\kappa$.

*Key Derivation Function (KDF).* Let $KDF : Salt_\kappa \times D_\kappa \to R_\kappa$ be a function. Function $KDF$ is a key derivation function if for any PPT distinguisher $\mathcal{D}$, any distribution $X_{D_\kappa}$ over $D_\kappa$ with $H_\infty(X_{D_\kappa}) \geq \kappa$, and any salt $s \in Salt_\kappa$, it holds that $\mathrm{Adv}^{KDF}_{\mathcal{D}}(\kappa) := |\Pr[\mathcal{D}(s, y) \to 1 \mid y \in_R R_\kappa] - \Pr[\mathcal{D}(s, KDF(s, x)) \to 1 \mid x \leftarrow X_{\mathcal{D}}]|$ is negligible for security parameter $\kappa$.

**Twisted PRF.** We give a definition of a twisted PRF (TPRF). This function is used with the input of ESKs and CSKs to this function in order to ensure that our construction is resilient to ephemeral secret key exposure.

**Definition 2 (Twisted Pseudorandom Function [21]).** *Let $F' : LD_\kappa \times RD_\kappa \to R_\kappa$ be a function. Function $F'$ is a TPRF if for any PPT distinguisher $\mathcal{D}$ and $\mathcal{D}'$ it holds that two difference $|\Pr[\mathcal{D}((b, F'(a, b))) \to 1 \mid a \in_R LD_\kappa, b \in_R RD_\kappa] - \Pr[\mathcal{D}((b, R)) \to 1 \mid b \in_R RD_\kappa, R \in_R \mathcal{R}_\kappa]|$ and $|\Pr[\mathcal{D}'((a, F'(a, b))) \to 1 \mid a \in_R LD_\kappa, b \in_R RD_\kappa] - \Pr[\mathcal{D}'((a, R)) \to 1 \mid a \in_R LD_\kappa, \mathcal{R} \in_R \mathcal{R}_\kappa]|$ are both negligible for security parameter $\kappa$.*

**KEM.** We recall the definition of IND-CPA security for KEM and the min-entropy of KEM keys. In our construction, session keys are computed from KEM keys via a KDF and a PRF. To use a KDF to obtain a PRF key computationally indistinguishable from a uniformly random PRF key, KEM keys need to have $\kappa$-min-entropy property.

A KEM scheme consists of three algorithms $(\mathsf{wKeyGen}, \mathsf{wEnCap}, \mathsf{wDeCap})$ with two randomness spaces $\mathcal{R}_{wK}$ and $\mathcal{R}_{wE}$, and a key space $\mathcal{KS}$.

$\mathsf{wKeyGen}(1^\kappa, r_g) \to (ek, dk)$ : It takes a security parameter, $1^\kappa$ and randomness, $r_g \in \mathcal{R}_{wK}$, and outputs a pair of encapsulation and decapsulation keys, $(ek, dk)$.

$\mathsf{wEnCap}(ek, r_e) \to (K, C)$ : It takes $ek$ and randomness, $r_e \in \mathcal{R}_{wE}$, and outputs a pair of a key and a ciphertext, $(K, C)$, where $K \in \mathcal{KS}$.

$\mathsf{wDeCap}(dk, C) \to K$ : It takes $dk$ and $C$, and outputs $K \in \mathcal{KS}$.

We require the correctness condition. That is, we have $K = \mathsf{wDeCap}(dk, C)$ for any $r_g \in \mathcal{R}_{wK}$, $r_e \in \mathcal{R}_{wE}$, $(ek, dk) \leftarrow \mathsf{wKeyGen}(1^\kappa, r_g)$, and $(K, C) \leftarrow \mathsf{wEnCap}(ek, r_e)$.

**Definition 3 (IND-CPA Security for KEM).** *A KEM is IND-CPA secure if for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, it holds that* $\mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{A}}(\kappa) = |2\Pr[(ek, dk) \leftarrow \mathsf{wKeyGen}(1^\kappa), state \leftarrow \mathcal{A}_1(ek), b \in_R \{0,1\}, (K_0^*, C_0^*) \leftarrow \mathsf{wEnCap}(ek), K_1^* \in_R \mathcal{KS}, b' \leftarrow \mathcal{A}_2(K_b^*, C_b^*, state), b' = b] - 1|$ *is negligible for security parameter* $\kappa$*.*

**Definition 4 (Min-Entropy of KEM Key).** *A KEM is* $\kappa$*-min-entropy KEM if for any encapsulation key* $ek$*, randomness* $r_e \in \mathcal{R}_{wE}$*, distribution* $D_{\mathcal{KS}}$ *of variable* $K$ *defined by* $(K, C) \leftarrow \mathsf{wEnCap}(ek, r_e)$*, and distribution* $D_{pub}$ *of public information,* $H_\infty(D_{\mathcal{KS}_{KEM}}|D_{pub}) \geq \kappa$ *holds, where* $H_\infty$ *denotes min-entropy.*

### 3.2 RHIB-KEM

We newly define RHIB-KEM and its two security notions: selective indistinguishability against chosen-ciphertext attacks (selective-IND-CCA), and min-entropy of RHIB-KEM keys. Our definition is based on Katsumata et al's definition of RHIBE and its security [19], and also includes adversary's access to the decapsulation oracle using the decapsulation key corresponding to the challenge time period and the challenge identity in the security game.

An RHIB-KEM scheme consists of six algorithms ($\mathsf{Setup}$, $\mathsf{SKGen}$, $\mathsf{KeyUp}$, $\mathsf{DKGen}$, $\mathsf{EnCap}$, $\mathsf{DeCap}$). The revocation list $RL_{ID,T}$ of a party with identity $ID$ in the time period $T$ is a subset of $\mathcal{I}_{ID}$, which contains the identities of its children. As in the definition of Katsumata et al., we do not explicitly introduce the revoke algorithm as part of our syntax since it is a simple operation of appending revoked party identities to a list. The above six algorithms have the following interfaces.

$\mathsf{Setup}(1^\kappa, L) \rightarrow (mpk, msk)$ : It takes a security parameter, $1^\kappa$ and the maximum depth of the hierarchy, $L \in \mathbb{N}$, and outputs a master public key, $mpk$ and a master secret key, $msk$.

$\mathsf{SKGen}(mpk, sk_{pa(ID)}, ID) \rightarrow (sk_{ID}, sk'_{pa(ID)})$ : It takes $mpk$, a parent's secret key, $sk_{pa(ID)}$, and an identity, $ID$, and outputs a secret key, $sk_{ID}$ for $ID$ and the parent's updated secret key, $sk'_{pa(ID)}$. Note that $sk_{pa(ID)} = msk$ if the PKG is the parent of $ID$.

$\mathsf{KeyUp}(mpk, T, sk_{ID}, RL_{ID,T}, ku_{pa(ID),T}) \rightarrow (ku_{ID,T}, sk'_{ID})$ : It takes $mpk$, a time period, $T \in \mathbb{N}$, $sk_{ID}$ corresponding to $ID$, a revocation list, $RL_{ID,T} \subset \mathcal{I}_{ID}$, and a parent's key update information, $ku_{pa(ID),T}$, and outputs a key update information, $ku_{ID,T}$ and the updated $sk'_{ID}$. Note that $sk_{ID} = msk$, $RL_{ID,T} = RL_{\mathsf{pkg},T}$, and $ku_{pa(ID),T} = \perp$ if $ID = \mathsf{pkg}$.

$\mathsf{DKGen}(mpk, sk_{ID}, ku_{pa(ID),T}) \rightarrow dk_{ID,T}$ or $\perp$ : It takes $mpk$, $sk_{ID}$ corresponding to $ID$, and a parent's $ku_{pa(ID),T}$, and outputs a decryption key, $dk_{ID,T}$ for the time period $T$ or $\perp$ indicating that $ID$ or some of its ancestors has been revoked.

$\mathsf{EnCap}(mpk, ID, T, r_e) \rightarrow (K, C)$ : It takes $mpk$, $ID$, a time period, $T$, and randomness, $r_e \in \mathcal{R}_E$, and outputs a key, $K \in \mathcal{KS}_{RH}$ and a ciphertext, $C$, where $\mathcal{R}_E$ is a randomness space, and $\mathcal{KS}_{RH}$ a key space.

$\mathsf{DeCap}(mpk, dk_{ID,T}, C) \to K'$ : It takes $mpk$, $dk_{ID,T}$, and $C$, and outputs a key, $K' \in \mathcal{KS}_{RH}$.

We require the correctness condition. That is, we have $K = \mathsf{DeCap}(mpk, dk_{ID,T}, C)$ for any $\kappa \in \mathbb{N}$, $L \in \mathbb{N}$, and $(mpk, msk) \leftarrow \mathsf{Setup}(1^\kappa, L)$, any identity $ID = (ID_1, \ldots, ID_t)$, any time period $T$, any pair of a key and a ciphertext $(K, C) \leftarrow \mathsf{EnCap}(mpk, ID, T)$ if $ID' \notin RL_{pa(ID'),T}$ holds for any identity $ID' \succ ID$.

*Security Definition.* We give a formal security definition for RHIB-KEM. Let $\Sigma = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{KeyUp}, \mathsf{DKGen}, \mathsf{EnCap}, \mathsf{DeCap})$ be an RHIB-KEM. This security definition is defined via a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. This game is parameterized by a security parameter $\kappa$ and a polynomial $L = L(\kappa)$ representing the maximum depth of the identity hierarchy. The game also has the global counter $T_{cu}$ initialized with 1, which denotes the current time period. The game proceeds as follows:

At the beginning, adversary $\mathcal{A}$ sends the challenge identity/time period pair $(ID^*, T^*)$ to challenger $\mathcal{C}$. Next, $\mathcal{C}$ runs $(mpk, msk) \leftarrow \mathsf{Setup}(1^\kappa, L)$, and prepares two lists: $\mathsf{SKList}$ that initially contains $(\mathsf{pkg}, msk)$ and $\mathsf{RevList} = \emptyset$. $\mathcal{C}$ also initializes $\mathsf{cflag} = 0$. Challenger $\mathcal{C}$ then executes $(ku_{\mathsf{pkg},1}, sk'_{\mathsf{pkg}}) \leftarrow \mathsf{KeyUp}(mpk, T_{cu} = 1, sk_{\mathsf{pkg}}, \emptyset, \perp)$ for generating a key update information for the initial time period $T_{cu} = 1$ and gives $(mpk, ku_{\mathsf{pkg},1})$ to $\mathcal{A}$. After that, adversary $\mathcal{A}$ may adaptively make queries in Figure 1. At some point, $\mathcal{A}$ outputs $b' \in \{0, 1\}$ as its guess for $b$ and terminates.

This completes the description of the game. In this game, $\mathcal{A}$'s advantage is defined by $\mathrm{Adv}^{\mathsf{s\text{-}ind\text{-}cca}}_{\mathcal{A},L}(\kappa) = |2\dot{\Pr}[b' = b] - 1|$.

If $\mathcal{A}$ never issues $\mathsf{DeCap}$ during this game, then the advantage is defined as $\mathrm{Adv}^{\mathsf{s\text{-}ind\text{-}cpa}}_{\mathcal{A},L}(\kappa)$ instead of $\mathrm{Adv}^{\mathsf{s\text{-}ind\text{-}cca}}_{\mathcal{A},L}(\kappa)$.

**Definition 5 (Selective-IND-CCA Security for RHIB-KEM).** *We say that an RHIB-KEM is selective-IND-CCA secure if for any PPT adversary $\mathcal{A}$, it holds that $\mathrm{Adv}^{\mathsf{s\text{-}ind\text{-}cca}}_{\mathcal{A},L}(\kappa)$ is negligible for security parameter $\kappa$.*

**Definition 6 (Selective-IND-CPA Security for RHIB-KEM).** *We say that an RHIB-KEM is selective-IND-CPA secure if for any PPT adversary $\mathcal{A}$, it holds that $\mathrm{Adv}^{\mathsf{s\text{-}ind\text{-}cpa}}_{\mathcal{A},L}(\kappa)$ is negligible for security parameter $\kappa$.*

Similar to KEM in Section 3.1, we define the min-entropy of RHIB-KEM key. We use RHIB-KEM keys as keys of PRFs via a KDF.

**Definition 7 (Min-Entropy of RHIB-KEM Key).** *An RHIB-KEM is $\kappa$-min-entropy RHIB-KEM if for any identity $ID$, any time period $T$, any randomness $r_e \in \mathcal{R}_E$, any distribution $D_{\mathcal{KS}}$ of variable $K$ defined by $(K, C) \leftarrow \mathsf{EnCap}(ID, T, r_e)$, and any distribution $D_{pub}$ of public information, $H_\infty(D_{\mathcal{KS}_{KEM}} | D_{pub}) \geq \kappa$ holds.*

### 3.3 Construction

Our construction is based on Fujioka et al.'s scheme [11] and generically consists of a selective-IND-CCA secure RHIB-KEM, an IND-CPA secure KEM, a PRF, a TPRF, and a KDF. This construction satisfies selective rhid-eCK security.

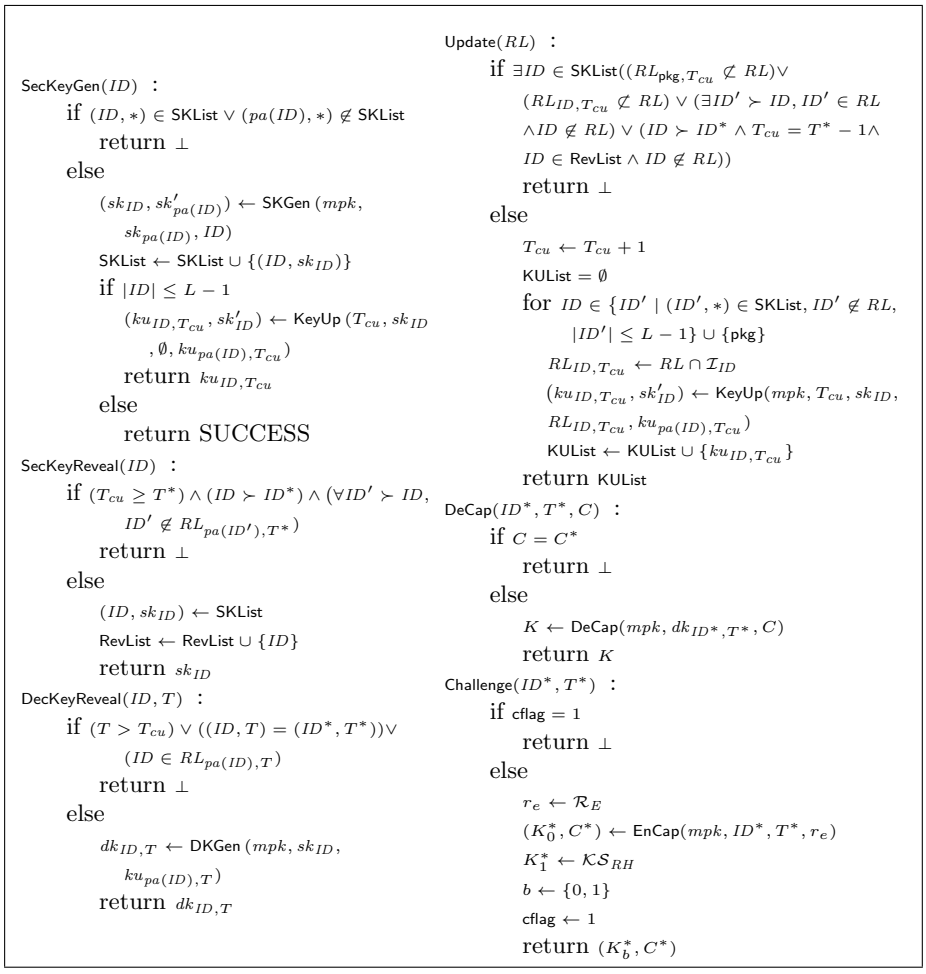


**Fig. 1.** Adversarial queries

**Design Principle.** We design our construction to solve three problems: 1) the exposure resistance of ESKs, 2) the exposure resistance of SSKs for already revoked identities, and 3) the exposure resistance of the MSK, SSKs, and CSKs.

For the first problem, we use a TPRF. In the construction, an ESK and a CSK are input to the TPRF. The output of the TPRF cannot be computed if the ESK is revealed but the CSK is not. We use the outputs of a TPRF as randomness to generate EPKs. This makes it impossible for an adversary to know the randomness since neither the ESK nor the CSK can be revealed by the freshness definition.

For the second problem, we use a selective-IND-CCA secure RHIB-KEM. It is difficult to obtain the RHIB-KEM key from the ciphertext associated with an

identity even if an adversary obtains the RHIB-KEM secret key of the SSK such that the corresponding identity has already been revoked. In the construction, two-session parties use RHIB-KEM keys generated with each other's peer identity to compute a session key. Therefore, we can provide the exposure resistance of SSKs for already revoked identities.

For the third problem, we use an IND-CPA secure KEM. An initiator generates session-specific encapsulation and decapsulation keys of the KEM, and the responder generates a KEM key for the encapsulation key. The initiator and the responder use the KEM key to compute a session key. This allows our construction to satisfy the exposure resistance of the MSK, SSKs, and CSKs.

**Our Generic Construction.** We construct an RHIB-AKE scheme $\Pi$ from an RHIB-KEM ($\mathsf{Setup}$, $\mathsf{SKGen}$, $\mathsf{KeyUp}$, $\mathsf{DKGen}$, $\mathsf{EnCap}$, $\mathsf{DeCap}$), a KEM ($\mathsf{wKeyGen}$, $\mathsf{wEnCap}$, $\mathsf{wDeCap}$) with two randomness spaces $\mathcal{R}_{wK}$ and $\mathcal{R}_{wE}$. Let $\mathcal{KS}$ be the key space of the outputs of $\mathsf{EnCap}$ and $\mathsf{wEnCap}$.

$\mathsf{ParGen}(1^\kappa, L) \to (MSK, MPK)$ : The PKG selects PRF $F : \mathcal{FS} \times \{0,1\}^* \to \{0,1\}^\kappa$ with key space $\mathcal{FS}$ of size $\kappa$, TPRF $G : LD_\kappa \times RD_\kappa \to \mathcal{R}_{wE}$ with $|LD_\kappa| = |RD_\kappa| = 2^\kappa$, and KDF $KDF : \{0,1\}^\kappa \times \mathcal{KS} \to \mathcal{FS}$ with randomly chosen public salt $s \in_R \{0,1\}^\kappa$. Moreover, the PKG generates master key pair $(mpk, msk) \leftarrow \mathsf{Setup}(1^\kappa, L)$. The PKG outputs $MSK = msk$ and $MPK = (F, G, KDF, s, mpk)$.

$\mathsf{SSKGen}(MPK, ID'_i, ID_i, SSK_{ID'_i}) \to (SSK_{ID_i}, SSK'_{ID'_i})$ : If $|ID| = 1$ i.e. the PKG is a parent of $ID$, the PKG generates $(sk_{ID}, msk') \leftarrow \mathsf{SKGen}(mpk, msk, ID)$ and outputs $SSK_{ID} = sk_{ID}$, $MSK' = msk'$. Otherwise, party $U_{pa(ID)}$ whose identity is the parent of identity $ID$ generates key pair $(sk_{ID}, sk'_{pa(ID)}) \leftarrow \mathsf{SKGen}(mpk, sk_{pa(ID)}, ID)$ with its secret key $SSK_{pa(ID)} = sk_{pa(ID)}$ and outputs $SSK_{ID} = sk_{ID}$ and $SSK'_{pa(ID)} = sk'_{pa(ID)}$.

$\mathsf{Rev}(MPK, T, RL_{ID_i,T-1}, Add) \to RL_{ID_i,T}$ : Party $U_i$ with identity $ID_i$ outputs $RL_{ID_i,T} = RL_{ID_i,T-1} \cup Add$.

$\mathsf{KUGen}(MPK, T, SSK_{ID_i}, RL_{ID_i,T}, ku_{ID'_i,T}) \to (ku_{ID_i,T}, SSK'_{ID_i})$ : If the PKG runs this algorithm with its revocation list $RL_{\mathsf{pkg},T}$, it runs $(ku_{\mathsf{pkg},T}, msk') \leftarrow \mathsf{KeyUp}(mpk, T, msk, RL_{\mathsf{pkg},T})$ and outputs $ku_{\mathsf{pkg},T}$ and $MSK' = msk'$. Otherwise, party $U_i$ with identity $ID_i$ runs this algorithm with its revocation list $RL_{ID_i,T}$. $U_i$ runs $(ku_{ID_i,T}, sk'_{ID_i}) \leftarrow \mathsf{KeyUp}(T, sk_{ID_i}, RL_{ID_i,T}, ku_{pa(ID_i),T})$ and outputs $ku_{ID_i,T}$ and $SSK'_{ID_i} = sk'_{ID_i}$.

$\mathsf{CSKGen}(MPK, SSK_{ID_i}, ku_{pa(ID_i),T}) \to CSK_{ID_i,T} / \perp$ : Party $U_i$ with identity $ID_i$ runs this algorithm. $U_i$ runs $dk_{ID_i,T} \leftarrow \mathsf{DKGen}(mpk, sk_{ID_i}, ku_{pa(ID_i),T})$ with its SSK $SSK_{ID_i} = sk_{ID_i}$ and generates $\sigma_{ID_i,T} \in_R LD_\kappa$ uniformly at random. $U_i$ outputs its CSK $CSK_{ID_i,T} = (dk_{ID_i,T}, \sigma_{ID_i,T})$.

*Key Exchange Protocol.* We describe our key exchange in Fig. 2. Here, we describe the detail. In the time period $T$, initiator $U_A$ with identity $ID_A$ and CSK $CSK_{A,T} = (dk_{A,T}, \sigma_{A,T})$, and responder $U_B$ with identity $ID_B$ and CSK $CSK_{B,T} = (dk_{B,T}, \sigma_{B,T})$ performs the following algorithms.

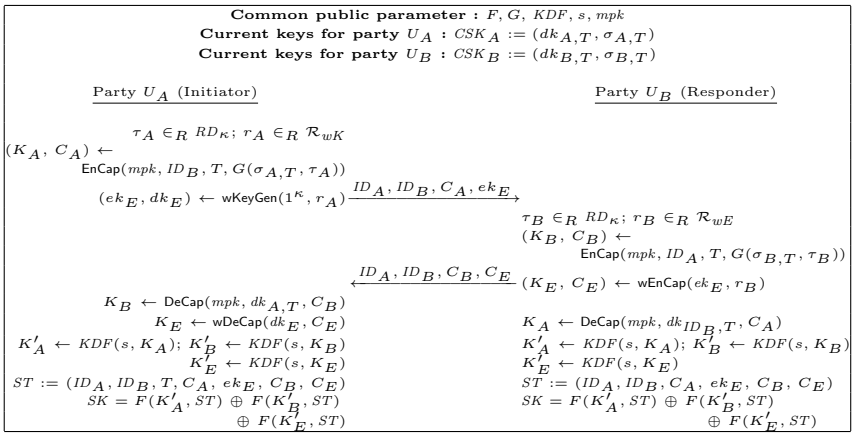


**Fig. 2.** Our key exchange

InitEK$(MPK, T, ID_A, SSK_A, CSK_{A,T}, ID_B) \to (ESK_A, EPK_A)$ : Party $U_A$ generates random elements $\tau_A \in_R RD_\kappa$ and $r_A \in \mathcal{R}_{wK}$, and computes a pair of a key and a ciphertext $(K_A, C_A) \leftarrow$ EnCap$(mpk, ID_B, T, G(\sigma_{A,T}, \tau_A))$ of the RHIB-KEM, and a pair of an encapsulation key and a decapsulation key $(ek_E, dk_E) \leftarrow$ wKeyGen$(1^\kappa, r_A)$ of the KEM. $U_A$ sets $ESK_A = (\tau_A, r_A)$ as its ESK and sends EPK $EPK_A = (ID_A, ID_B, T, C_A, ek_E)$ to $U_B$.

ResEK$(MPK, T, ID_B, SSK_B, CSK_{B,T}, ID_A, EPK_A) \to (ESK_B, EPK_B)$ : Party $U_B$ generates random elements $\tau_B \in_R RD_\kappa$ and $r_B \in \mathcal{R}_{wE}$, and computes a pair of a key and a ciphertext $(K_B, C_B) \leftarrow$ EnCap$(mpk, ID_A, T, G(\sigma_{B,T}, \tau_B))$ of the RHIB-KEM, and a pair of a key and a ciphertext $(K_E, C_E) \leftarrow$ wEnCap$(ek_E, r_B)$ of the KEM. $U_B$ sets $ESK_B = (\tau_B, r_B)$ as its ESK and sends EPK $EPK_B = (ID_A, ID_B, T, C_B, C_E)$ to $U_A$.

ResSK$(MPK, T, ID_B, SSK_B, CSK_{B,T}, ID_A, ESK_B, EPK_B, EPK_A) \to SK$ : Party $U_B$ decrypts $K_A \leftarrow$ DeCap$(mpk, dk_{ID_B,T}, C_A)$ and computes $K'_A \leftarrow KDF(s, K_A)$, $K'_B \leftarrow KDF(s, K_B)$, and $K'_E \leftarrow KDF(s, K_E)$. $U_B$ computes session key $SK = F(K'_A, ST) \oplus F(K'_B, ST) \oplus F(K'_E, ST)$ with session transcript $ST = (ID_A, ID_B, T, C_A, ek_E, C_B, C_E)$.

InitSK$(MPK, T, ID_A, SSK_A, CSK_{A,T}, ID_B, ESK_A, EPK_A, EPK_B) \to SK$ : Party $U_A$ decrypts $K_B \leftarrow$ DeCap$(mpk, dk_{A,T}, C_B)$ and $K_E \leftarrow$ wDeCap$(dk_E, C_E)$, and computes $K'_A \leftarrow KDF(s, K_A)$, $K'_B \leftarrow KDF(s, K_B)$, and $K'_E \leftarrow KDF(s, K_E)$. $U_A$ computes session key $SK = F(K'_A, ST) \oplus F(K'_B, ST) \oplus F(K'_E, ST)$ with session transcript $ST = (ID_A, ID_B, T, C_A, ek_E, C_B, C_E)$.

**Theorem 1.** *If RHIB-KEM* (Setup, SKGen, KeyUp, DKGen, EnCap, DeCap) *is selective-IND-CCA secure and is $\kappa$-min-entropy RHIB-KEM, KEM* (wKeyGen, wEnCap, wDeCap) *si IND-CPA secure and is $\kappa$-min-entropy KEM, function $F$ is a PRF, function $G$ is a TPRF, and function KDF is a KDF, then RHIB-AKE scheme $\Pi$ is selective secure in the rhid-eCK model.*

We show the proof of Theorem 1 in Appendix A. We give an overview of the security proof. We consider the following six maximal exposure patterns in the rhid-eCK model and the presence of the matching session: a) both SSKs or both CSKs, b) the SSK or CSK of $U_A$, the ESK of $U_B$, and the SSK of $U_B$ such that $ID_B$ has been revoked, c) the SSK or CSK of $U_A$ and the ESK of $U_B$, d) the SSK or CSK of $U_B$ and the ESK of $U_A$, e) the ESK of $U_A$, the ESK of $U_B$, and the SSK of $U_B$ such that $ID_B$ has been revoked, and f) both ESKs.

In case a), $K_E$ is protected by the security of KEM since $r_A$ and $r_B$ are not exposed. In case b), $\tau_A$ is not exposed and $dk_B$ is not generated. Therefore, $G(\sigma_{A,T}, \tau_A)$ is hidden and $K_A$ is protected by the security of RHIB-KEM. In case c), $\tau_A$ and $dk_B$ are not exposed. Therefore, $G(\sigma_{A,T}, \tau_A)$ is hidden and $K_A$ is protected by the security of RHIB-KEM. In case d), $\tau_B$ and $dk_A$ are not exposed. Therefore, $G(\sigma_{B,T}, \tau_B)$ is hidden and $K_B$ is protected by the security of RHIB-KEM. In case e), $\sigma_A$ is not exposed and $dk_B$ is not generated. Therefore, $G(\sigma_{A,T}, \tau_A)$ is hidden and $K_A$ is protected by the security of RHIB-KEM. In case f), $\sigma_A$ and $dk_A$ are not exposed. Therefore, $G(\sigma_{A,T}, \tau_A)$ is hidden and $K_A$ is protected by the security of RHIB-KEM.

Then, we transform the rhid-eCK security game so that the session key in the test session is randomly distributed. First, we change the output of the TPRF into a randomly chosen value since one input of the TPRF is hidden from the adversary; therefore, the randomness of the protected RHIB-KEM can be randomly distributed. Second, we change the protected RHIB-KEM key or KEM key into a random key; therefore, the input of KDF is randomly distributed and has sufficient min-entropy. Third, we change the output of KDF into randomly chosen values. Finally, we change one of the PRFs (corresponding to the protected RHIB-KEM or KEM) into a random function. Therefore, the session key in the test session is randomly distributed; thus, there is no advantage to the adversary. We can show a similar proof in non-matching cases.

## 4 Instantiations

In this section, we provide some RHIB-AKE schemes as concrete instantiations based on the pairings and lattices from our construction in Section 3. We consider instantiations of selective-IND-CCA secure RHIB-KEM and IND-CPA secure KEM from Theorem 1. Note that a PRF can be instantiated with HMAC [1] or a block cipher such as AES [7] and both TPRF and KDF can be constructed from a PRF [11, 6]. To the best of our knowledge, no selective-IND-CCA secure RHIB-KEM has ever been proposed, but selective-IND-CPA secure RHIBE does exist. We give a method for constructing a selective-IND-CCA secure RHIB-KEM from a selective-IND-CPA secure RHIBE. We then give instantiations of selective-IND-CPA secure RHIBE and IND-CPA secure KEM.

### 4.1 Construction of RHIB-KEM

We construct a selective IND-CCA secure RHIB-KEM $\Sigma = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{KeyUp}, \mathsf{DKGen}, \mathsf{EnCap}, \mathsf{DeCap})$ from a selective IND-CPA secure RHIBE scheme

$\Sigma' = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{KeyUp}, \mathsf{DKGen}, \mathsf{Enc}, \mathsf{Dec})$. The syntax and security definition of RHIBE follows Katsumata et al. [19]. RHIBE differs from RHIB-KEM in Enc and Dec, and Enc is an algorithm that takes the input to EnCap of RHIB-KEM and a message as additional input, and outputs a ciphertext, while the input and output of Dec is the same as DeCap of RHIB-KEM. The selective-IND-CPA security game for RHIBE is identical to the selective-IND-CPA security game for RHIB-KEM except that instead of an adversary distinguishing a real KEM key and a random key, the adversary distinguishes a ciphertext in one of two messages of its choice.

Let the randomness space of Enc in $\Sigma'$ be $\mathcal{R}_{Enc}$, $H : \{0,1\}^* \to \mathcal{R}_{Enc}$ be a hash function with an appropriate output length, modeled as a random oracle. The construction is as follows: Setup, SKGen, KeyUp, and DKGen are the same as $\mathsf{Setup}'$, $\mathsf{SKGen}'$, $\mathsf{KeyUp}'$, and $\mathsf{DKGen}'$ respectively.

$\mathsf{EnCap}(mpk, ID, T, r_e) \to (K, C)$ : It chooses $K$ from $\mathcal{KS}_{RH}$ uniformly at random. It computes $C \leftarrow \mathsf{Enc}(mpk, ID, T, K||ID||T||r_e, H(m||ID||T||r_e))$ and outputs $K, C$.

$\mathsf{DeCap}(mpk, dk_{ID,T}, C) \to K'$ : It computes $K'||ID'||T'||r'_h \leftarrow \mathsf{Dec}(mpk, dk_{ID,T}, C)$. If $C = \mathsf{Enc}(mpk, ID', T', K'||ID'||T'||r'_h, H(K'||ID'||T'||r'_h))$ holds, it outputs $K'$. Otherwise, it outputs $\perp$.

This construction is the Fujisaki-Okamoto transformation [13] of $\Sigma'$. Therefore, RHIB-KEM $\Sigma$ is selective-IND-CCA secure in the random oracle model if RHIBE $\Sigma'$ is selective-IND-CPA secure. It is also $\kappa$-min-entropy RHIB-KEM when the size of the keyspace is larger than $2^\kappa$ and keys are chosen uniformly at random. Note that when encrypting a $\kappa$-bit key with a 1-bit encryption scheme such as some lattice-based RHIBEs [19, 35], we can encrypt all bits to $\kappa$ ciphertexts, which can be seen as a ciphertext of a $\kappa$-bit key.

### 4.2 Pairing-based instantiations

We can apply selective-IND-CPA secure RHIBE schemes to our RHIB-AKE from the symmetric external Diffie-Hellman (SXDH) assumption [8, 9, 33] and the ElGamal KEM as an IND-CPA KEM to our RHIB-AKE from the decisional Diffie-Hellman (DDH) assumption. We compared our instantiations of RHIB-AKE scheme with some RHIBEs and the ElGamal KEM in terms of the computational costs and the communication sizes in Table 1 for reference, where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is the asymmetric pairing with prime order $p$ used in those RHIBEs, the ElGamal KEM is the scheme on group $\mathbb{G}_1$, $\ell$ is the hierarchical level of initiator and responder IDs, $D = \lceil \log_2 N \rceil$ for the integer $N$ which is the maximum number of child nodes in each node, P means pairing operation, $\mathrm{E}_t$ $(t = 1, 2, T)$ means exponentiation operation on $\mathbb{G}_t$, $\mathrm{len}(g_t)$ $(t = 1, 2, T)$ is the length of a group element of $\mathbb{G}_t$, and $\mathrm{len}(p)$ is the length of a field element of the prime field $\mathbb{Z}_p$. We use Emura et al.'s scheme [8] instantiated by Chen and Wee's hierarchical identity-based encryption (HIBE) scheme [4], Emura et al.'s basic scheme [9] instantiated by Chen and Wee's HIBE scheme [4], and Takayasu's

**Table 1.** Comparison of our instantiations

| Instantiation of RHIBE | Computation (per party) | Communication size |
|---|---|---|
| Emura et al. [8] + Chen and Wee [4] | $2((\ell+4)\mathrm{E}_1+\mathrm{E}_T)$ $+3\mathrm{P}+2\mathrm{E}_1$ | $8\mathrm{len}(g_1)+2\mathrm{len}(g_T)$ |
| Emura et al. [9] + Chen and Wee [4] | $2(\ell(D+1)+1)$ $\times((\ell+4)\mathrm{E}_1+\mathrm{E}_T)$ $+3(\ell+1)\mathrm{P}+2\mathrm{E}_1$ | $2(3\ell(D+1)+4)\mathrm{len}(g_1)$ $+2(\ell(D+1)+1)\mathrm{len}(g_T)$ |
| Takayasu [33] | $2((\ell+7)\mathrm{E}_1+\mathrm{E}_T)$ $+(4\mathrm{P}+4\mathrm{E}_2)+2\mathrm{E}_1$ | $10\mathrm{len}(g_1)+2\mathrm{len}(g_T)$ $+4\mathrm{len}(p)$ |

scheme [33] as RHIBEs. In our construction, both initiator and responder need to perform one encapsulation and one decapsulation algorithm of selective-IND-CCA secure RHIB-KEM to perform a key exchange. In other words, they need to perform two encryption algorithms and one decryption algorithm of selective-IND-CPA secure RHIBE. In addition, they require two exponentiation operations on $\mathbb{G}_1$ to be performed in the process of using the ElGamal KEM. The communication size in Table 1 is the sum of the sizes of two ciphertexts of the RHIB-KEM scheme, one encapsulation key of the KEM, and one ciphertext of the KEM, excluding the size of initiator and responder IDs and the time period in the EPKs, sent by them.

In Appendix B, we estimate the performance of implementing instantiated RHIB-AKE scheme as shown in Table 1 on Rasberry Pi3, which is a well-known IoT device. If the hierarchical level $\ell$ is small (i.e., $\ell$=2 or 3) and the device is the same spec as the Raspberry Pi3, using RHIB-AKE is quite practical.

### 4.3 Lattice-based instantiations

We can apply selective-IND-CPA secure RHIBE schemes to our RHIB-AKE from the learning with errors (LWE) assumption [19, 35]. We can easily obtain IND-CPA secure KEMs with min-entropy $\kappa$ from IND-CPA secure public-key encryption (PKE) scheme with the message space size larger than $2^\kappa$. We can use lattice-based IND-CPA secure PKEs from the (Ring-)LWE assumption [25, 26, 28, 32]. Estimating performance of instantiated schemes in the same way as pairing-based schemes is a future work.

## 5 Conclusion

We gave the first definition of an RHIB-AKE scheme and its security model called the rhid-eCK model. An RHIB-AKE scheme allows parties under different PKGs to share a session key by simply trusting a master public key generated by the PKG at the root node, and also allows parties to be revoked. The rhid-eCK security implies that a scheme resists against leakage of all combinations of master, static, current, and ephemeral secret keys except ones trivially break

the security. We also proposed the first construction of RHIB-AKE that satisfies the rhid-eCK model and its instantiations based on the parings and lattices.

In this work we only estimated the performance of pairing-based RHIB-AKE schemes. Therefore it is necessary to actually implement schemes and check the performance as future works. We also leave implementing and evaluating the performance of lattice-based RHIB-AKE schemes which are essential to realizing post-quantum authentication.

We believe that our RHIB-AKE scheme can be built as an authentication system consisting of a huge number of IoT devices. Password-based authentication with default or weak passwords is widely used in IoT systems, and we consider replacing such insecure methods with RHIB-AKE will be a new remedy.

## References

1. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) Advances in Cryptology — CRYPTO '96. pp. 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg (1996)
2. Boldyreva, A., Goyal, V., Kumar, V.: Identity-based encryption with efficient revocation. In: Proceedings of the 15th ACM Conference on Computer and Communications Security. pp. 417–426. CCS '08, Association for Computing Machinery, New York, NY, USA (2008). https://doi.org/10.1145/1455770.1455823, https://doi.org/10.1145/1455770.1455823
3. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) Advances in Cryptology — CRYPTO 2001. pp. 213–229. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
4. Chen, J., Wee, H.: Dual system groups and its applications — compact hibe and more. Cryptology ePrint Archive, Report 2014/265 (2014), https://eprint.iacr.org/2014/265
5. Chen, L., Cheng, Z., Smart, N.P.: Identity-based key agreement protocols from pairings. International Journal of Information Security **6**(4), 213–241 (2007). https://doi.org/10.1007/s10207-006-0011-9, https://doi.org/10.1007/s10207-006-0011-9
6. Dachman-Soled, D., Gennaro, R., Krawczyk, H., Malkin, T.: Computational extractors and pseudorandomness. In: Cramer, R. (ed.) Theory of Cryptography. pp. 383–403. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
7. Daemen, J., Rijmen, V.: The design of Rijndael: AES-Advanced Encryption Standard. Springer (2002)
8. Emura, K., Takayasu, A., Watanabe, Y.: Adaptively secure revocable hierarchical ibe from $k$-linear assumption. Cryptology ePrint Archive, Report 2020/886 (2020), https://eprint.iacr.org/2020/886
9. Emura, K., Takayasu, A., Watanabe, Y.: Generic constructions of revocable hierarchical identity-based encryption. Cryptology ePrint Archive, Report 2021/515 (2021), https://eprint.iacr.org/2021/515
10. Fujioka, A., Hoshino, F., Kobayashi, T., Suzuki, K., Ustaoglu, B., Yoneyama, K.: id-eck secure id-based authenticated key exchange on symmetric and asymmetric pairing. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E96.A**(6), 1139–1155 (2013). https://doi.org/10.1587/transfun.E96.A.1139

11. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. Designs, Codes and Cryptography **76**(3), 469–504 (2015), https://doi.org/10.1007/s10623-014-9972-2
12. Fujioka, A., Suzuki, K., Yoneyama, K.: Hierarchical id-based authenticated key exchange resilient to ephemeral key leakage. In: Echizen, I., Kunihiro, N., Sasaki, R. (eds.) Advances in Information and Computer Security. pp. 164–180. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
13. Fujisaki, E., Okamoto, T.: How to enhance the security of public-key encryption at minimum cost. In: Public Key Cryptography. pp. 53–68. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
14. Galbraith, S.D., Lin, X., Scott, M.: Endomorphisms for faster elliptic curve cryptography on a large class of curves. Journal of Cryptology **24**(3), 446–469 (Jul 2011). https://doi.org/10.1007/s00145-010-9065-y, https://doi.org/10.1007/s00145-010-9065-y
15. Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Faster point multiplication on elliptic curves with efficient endomorphisms. In: Kilian, J. (ed.) Advances in Cryptology — CRYPTO 2001. pp. 190–200. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
16. Huang, H., Cao, Z.: An id-based authenticated key exchange protocol based on bilinear diffie-hellman problem. In: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security. pp. 333–342. ASIACCS '09, Association for Computing Machinery, New York, NY, USA (2009), https://doi.org/10.1145/1533057.1533101
17. Ishibashi, R., Yoneyama, K.: Adaptive-id secure hierarchical id-based authenticated key exchange under standard assumptions without random oracles. In: Sako, K., Tippenhauer, N.O. (eds.) Applied Cryptography and Network Security. pp. 3–27. Springer International Publishing, Cham (2021)
18. Ishida, Y., Watanabe, Y., Shikata, J.: Constructions of cca-secure revocable identity-based encryption. In: Foo, E., Stebila, D. (eds.) Information Security and Privacy. pp. 174–191. Springer International Publishing, Cham (2015)
19. Katsumata, S., Matsuda, T., Takayasu, A.: Lattice-based revocable (hierarchical) ibe with decryption key exposure resistance. In: Lin, D., Sako, K. (eds.) Public-Key Cryptography – PKC 2019. pp. 441–471. Springer International Publishing, Cham (2019)
20. Krawczyk, H.: Cryptographic extraction and key derivation: The hkdf scheme. In: Rabin, T. (ed.) Advances in Cryptology – CRYPTO 2010. pp. 631–648. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
21. Kurosawa, K., Furukawa, J.: 2-pass key exchange protocols from cpa-secure kem. In: Benaloh, J. (ed.) Topics in Cryptology – CT-RSA 2014. pp. 385–401. Springer International Publishing, Cham (2014)
22. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) Provable Security. pp. 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
23. Lee, K., Kim, J.S.: A generic approach to build revocable hierarchical identity-based encryption. Cryptology ePrint Archive, Report 2021/502 (2021), https://eprint.iacr.org/2021/502
24. Lee, K., Park, S.: Revocable hierarchical identity-based encryption with shorter private keys and update keys. Designs, Codes and Cryptography **86**(10), 2407–2440 (2018). https://doi.org/10.1007/s10623-017-0453-2, https://doi.org/10.1007/s10623-017-0453-2

25. Lindner, R., Peikert, C.: Better key sizes (and attacks) for lwe-based encryption. In: Kiayias, A. (ed.) Topics in Cryptology – CT-RSA 2011. pp. 319–339. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
26. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. pp. 1–23. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
27. McCullagh, N., Barreto, P.S.L.M.: A new two-party identity-based authenticated key agreement. In: Menezes, A. (ed.) Topics in Cryptology – CT-RSA 2005. pp. 262–274. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
28. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing. pp. 187–196. STOC '08, Association for Computing Machinery, New York, NY, USA (2008), https://doi.org/10.1145/1374376.1374406
29. Seo, J.H., Emura, K.: Efficient delegation of key generation and revocation functionalities in identity-based encryption. In: Dawson, E. (ed.) Topics in Cryptology – CT-RSA 2013. pp. 343–358. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
30. Seo, J.H., Emura, K.: Revocable hierarchical identity-based encryption. Theoretical Computer Science **542**, 44–62 (2014)
31. Seo, J.H., Emura, K.: Revocable hierarchical identity-based encryption: History-free update, security against insiders, and short ciphertexts. In: Nyberg, K. (ed.) Topics in Cryptology — CT-RSA 2015. pp. 106–123. Springer International Publishing, Cham (2015)
32. Stehlé, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public key encryption based on ideal lattices. In: Matsui, M. (ed.) Advances in Cryptology – ASIACRYPT 2009. pp. 617–635. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
33. Takayasu, A.: More efficient adaptively secure revocable hierarchical identity-based encryption with compact ciphertexts: Achieving shorter keys and tighter reductions. Cryptology ePrint Archive, Report 2021/539 (2021), https://eprint.iacr.org/2021/539
34. Tomida, J., Fujioka, A., Nagai, A., Suzuki, K.: Strongly secure identity-based key exchange with single pairing operation. In: Sako, K., Schneider, S., Ryan, P.Y.A. (eds.) Computer Security – ESORICS 2019. pp. 484–503. Springer International Publishing, Cham (2019)
35. Wang, S., Zhang, J., He, J., Wang, H., Li, C.: Simplified revocable hierarchical identity-based encryption from lattices. In: Mu, Y., Deng, R.H., Huang, X. (eds.) Cryptology and Network Security. pp. 99–119. Springer International Publishing, Cham (2019)
36. Yoneyama, K.: Practical and exposure-resilient hierarchical id-based authenticated key exchange without random oracles. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E97.A**(6), 1335–1344 (2014). https://doi.org/10.1587/transfun.E97.A.1335

## A Proof of Theorem 1

In the rhid-eCK security experiment, we suppose that $\mathsf{sid}^*$ is the session identifier for the test session and that at most $\mu$ sessions are activated. Let $\kappa$ be the security parameter, and $\mathcal{A}$ be a PPT (in $\kappa$) bounded adversary. We also suppose that $\mathcal{A}$ outputs $(T^*, ID_A, ID_B)$ at the beginning of this experiment, where $ID_A$ (resp. $ID_B$) is party $U_A$'s (resp. $U_B$'s) ID. *Suc* denotes the event that $\mathcal{A}$ wins.

We consider 14 events covering all cases of $\mathcal{A}$'s behavior: 8 events when the test session has no matching session, and 6 events when it has a matching session. The former can be divided into two main cases, depending on whether the session owner is an initiator or a responder. In each case, we consider the following exposure patterns: 1) the CSK or SSK of the owner and the SSK of the peer such that the peer has been revoked, 2) the ESK of the owner and the SSK of the peer such that the peer has been revoked, 3) the CSK or SSK of the owner, and 4) the ESK of the owner. We consider the event where the owner of the test session is an initiator and the exposure pattern in 1) occurs. That is,

$E_1$ : $U_A$ is the initiator and the owner of $\mathsf{sid}^*$, $\mathsf{sid}^*$ has no matching session $\overline{\mathsf{sid}^*}$, $\mathcal{A}$ issues $\mathsf{CSKReveal}(ID_A, T^*)$ or $\mathsf{SSKReveal}(ID)$ for some $ID \succ ID_A$, and $\mathcal{A}$ issues $\mathsf{SSKReveal}(ID)$ such that $ID \succ ID_B$ and $ID \in RL_{pa(ID),T^*-1}$.

We can evaluate the probability of an adversary winning if the other seven events occur as well as if $E_1$ occurs. The latter, i.e. the events where the test session has a matching session, consists of the six exposure patterns shown in the overview of the proof for Theorem 1 of Section 3. We consider the following event.

$E_2$ : There exists a matching session $\overline{\mathsf{sid}^*}$ of $\mathsf{sid}^*$, and $\mathcal{A}$ issues $\mathsf{CSKReveal}(ID_X, T^*)$ or $\mathsf{SSKReveal}(ID)$ for some $ID \succ ID_X$, where $X = A, B$.

We can evaluate the probability of an adversary winning if the other five events occur as well as if $E_1$ or $E_2$ occurs. Therefore, we evaluate $|2\Pr[Suc \mid E_i] - 1|$ given the event $E_i$ for $i = 1, 2$ to finish the proof.

**Event $E_1$.** We change the interface of oracle queries and the computation of the session key. These instances are gradually changed over hybrid experiments, depending on specific sub-cases. In the last hybrid experiment, the session key in the test session does not contain information of the bit $b$. Thus, $\mathcal{A}$ only outputs a random guess. Let $\mathbf{H}_0, \ldots, \mathbf{H}_6$ be these hybrid experiments, $S_i$ be the event that $\mathcal{A}$ wins in $\mathbf{H}_i$, $T_{cu}$ be the current time period, and $s_A$ be the number of sessions of $U_A$ which have been activated, which is initialized with 0.

*Hybrid experiment* $\mathbf{H}_0$. This experiment is the real experiment for rhid-eCK security and in this experiment, the environment for $\mathcal{A}$ is as defined in the scheme. Thus, $|2\Pr[Suc \mid E_3] - 1| = |2\Pr[S_0 \mid E_3] - 1|$.

*Hybrid experiment* $\mathbf{H}_1$. If session identifiers in two sessions are identical, the experiment halts, a bit $b'$ is randomly selected, and $\mathcal{A}$ is considered to output $b'$. Two session identifiers are identical if and only if the initiators and responders of the two sessions match and the EPKs $(C_A, ek_E, C_B, C_E)$ output by the two sessions are equal. When $ek_E$ and $C_E$ are equal in the two sessions, these $K_E$ are also equal by the correctness of KEM. The probability that these $K_E$ are equal is at most $1/2^\kappa$ by the $\kappa$-min-entropy property of KEM. Therefore, $|\Pr[S_0 \mid E_3] - \Pr[S_1 \mid E_3]|$ is negligible for $\kappa$.

*Hybrid experiment* $\mathbf{H}_2$. The experiment selects an integer $i \in [1, \mu]$ randomly in advance. If $\mathcal{A}$ issues $\mathsf{Test}$ query to a session except $i$-th session of party $U_A$, the experiment halts, a bit $b'$ is randomly selected, and $\mathcal{A}$ is considered to output $b'$. Since guess of the test session matches with $\mathcal{A}$'s choice with probability $1/\mu$, $|2\Pr[S_1 \mid E_3] - 1| = \mu \cdot |2\Pr[S_2 \mid E_3] - 1|$.

*Hybrid experiment* $\mathbf{H}_3$. The computation of $(K_A^*, C_A^*)$ in the test session is changed. Instead of computing $(K_A^*, C_A^*) \leftarrow \mathsf{EnCap}(mpk, ID_B, T^*, G(\sigma_{A,T}, \tau_A))$, it is changed as $(K_A^*, C_A^*) \leftarrow \mathsf{EnCap}(mpk, ID_B, T^*, R)$, where $R \in_R \mathcal{R}_E$.

Adversary $\mathcal{A}$ does not issue $\mathsf{ESKReveal}(\mathsf{sid}^*)$ from the freshness definition. Hence, we construct a distinguisher $\mathcal{D}$ between $(\sigma_{A,T}, G(\sigma_{A,T}, \tau_A))$ and $(\sigma_{A,T}, R)$ for TPRF $G$ from $\mathcal{A}$ in $\mathbf{H}_2$ or $\mathbf{H}_3$. $\mathcal{D}$ simulates obeying the scheme, except that $\mathcal{D}$ computes $(K_A^*, C_A^*) \leftarrow \mathsf{EnCap}(mpk, ID_B, T^*, R)$ for $\mathsf{Send}(\Pi, \mathcal{I}, T^*, ID_A, ID_B)$, where $R$ is either of the output of TPRF $G$ or random element. From $\mathcal{A}$'s point of view, the simulation by $\mathcal{D}$ is same as $\mathbf{H}_2$ if $R$ input to $\mathcal{D}$ is the output of TPRF $G$. Otherwise, the simulation by $\mathcal{D}$ is the same as $\mathbf{H}_3$. Thus, $|\Pr[S_2 \mid E_3] - \Pr[S_3 \mid E_3]|$ is negligible for $\kappa$ since the advantage of $\mathcal{D}$ is negligible.

*Hybrid experiment* $\mathbf{H}_4$. The computation of $K_A^*$ in the test session is changed again. Instead of computing $(K_A^*, C_A^*) \leftarrow \mathsf{EnCap}(mpk, ID_B, T^*, R)$, it is changed as choosing $K_A^* \leftarrow \mathcal{KS}_{RH}$ randomly.

We construct a selective-IND-CCA adversary $\mathcal{B}$ against RHIB-KEM from $\mathcal{A}$ in $\mathbf{H}_3$ or $\mathbf{H}_4$. $\mathcal{B}$ synchronizes the time period in the selective-IND-CCA game with the time period in the rhid-eCK game. $\mathcal{B}$ performs the following steps.

Adversary $\mathcal{B}$ gives the challenge identity/time period pair $(ID_B, T^*)$ to challenger $\mathcal{C}$ and receives a master public key $mpk$ and the key update information $ku_{PKG,1}$ in $T_{cu} = 1$. Then, $\mathcal{B}$ chooses a PRF $F : \mathcal{FS} \times \{0,1\}^* \to \{0,1\}^\kappa$ with key space $\mathcal{FS}$, a TPRF $G : LD_\kappa \times RD_\kappa \to \mathcal{R}_{wE}$ and a KDF $KDF : \{0,1\}^\kappa \times \mathcal{KS} \to \mathcal{FS}$ with randomly chosen public salt $s \in \{0,1\}^\kappa$. $\mathcal{B}$ also sets $MPK = (F, G, s, KDF, mpk)$. $\mathcal{B}$ choose a set of identities $\mathcal{ID}$ for honest parties, including $ID_A$ and $ID_B$, issues $\mathsf{SecKeyGen}(ID)$ for all $ID \in \mathcal{ID}$, and gives $\{ku_{ID,1}\}_{ID \in \mathcal{ID}}$ answered by the oracle to $\mathcal{A}$. $\mathcal{B}$ then gives $MPK$, $\mathcal{ID}$, and $\{ku_{ID,1}\}_{ID \in \mathcal{ID}}$ to $\mathcal{A}$.

In preparation for $\mathcal{A}$'s oracle queries, $\mathcal{B}$ creates a list $\mathcal{L}_S$ of sessions, a list $\mathcal{L}_{SK}$ of completed session $\mathsf{sid}$ and session key $SK$ pairs, and a list $\mathcal{L}_{NCS}$ of non-completed sessions. Initially, these lists are empty sets. $\mathcal{B}$ simulates oracle queries by $\mathcal{A}$ as Figure 3. When $\mathcal{A}$ outputs a guess $b'$, if $\mathcal{A}$'s guess is correct, $\mathcal{B}$ answers that $K^*$ received by the challenge query is the real key, otherwise it answers that $K^*$ is the random key.

From $\mathcal{A}$'s point of view, the simulation by $\mathcal{B}$ is the same as $\mathbf{H}_3$ if $K^*$ that $\mathcal{B}$ received in the challenge was the real key. Otherwise, the simulation by $\mathcal{B}$ is the same as $\mathbf{H}_4$. Thus, $|\Pr[S_3 \mid E_3] - \Pr[S_4 \mid E_3]|$ is negligible for $\kappa$ since the advantage of $\mathcal{B}$ is negligible.

*Hybrid experiment* $\mathbf{H}_5$. The computation of $K'^*_A$ in the test session is changed. Instead of computing $K'^*_A \leftarrow KDF(s, K_A^*)$, it is changed as choosing $K'^*_A \in \mathcal{KS}$ randomly. $K_A^*$ has sufficient min-entropy since it is randomly chosen in $\mathbf{H}_4$. Thus, $|\Pr[S_4 \mid E_3] - \Pr[S_5 \mid E_3]|$ is negligible for $\kappa$ by the definition of the KDF.

*Hybrid experiment* $\mathbf{H}_6$. The computation of $SK^*$ in the test session is changed. Instead of computing $SK^* = F(K'^*_A, ST) \oplus F(K'^*_B, ST) \oplus F(K'^*_E, ST)$, it is changed as $SK^* = x \oplus F(K'^*_B, ST) \oplus F(K'^*_E, ST)$ where $x \in \{0,1\}^\kappa$ is chosen randomly. We construct a distinguisher $\mathcal{D}'$ between PRF $F$ and a random function from $\mathcal{A}$ in $\mathbf{H}_5$ or $\mathbf{H}_6$. $\mathcal{D}$ simulates the security game obeying the scheme, except that $\mathcal{D}$ computes $SK^* = x \oplus F(K'^*_B, ST) \oplus F(K'^*_E, ST)$ for $\mathsf{Send}(\Pi, \mathcal{I}, T^*, ID_A, ID_B)$, where $x$ is either of the output of $F$ or $RF$. From $\mathcal{A}$'s point of view, the simulation by $\mathcal{D}'$ is the same as $\mathbf{H}_5$ if the oracle it accesses is PRF $F$. Otherwise, the simulation by $\mathcal{D}'$ is the same as $\mathbf{H}_6$. Thus, $|\Pr[S_5 \mid E_3] - \Pr[S_6 \mid E_3]|$ is negligible for $\kappa$ since the advantage of $\mathcal{D}'$ is negligible.

The session key in the test session is perfectly randomized in $\mathbf{H}_6$. We have $\Pr[S_6 \mid E_3] = 1/2$ since $\mathcal{A}$ cannot obtain any advantage from $\mathsf{Test}$ query. Thus, $|2\Pr[Suc \mid E_3] - 1|$ is negligible for $\kappa$.

**Event $E_2$.** We change the interface of oracle queries and the computation of the session key as in the case of $E_1$. Let $\mathbf{H}'_0, \ldots, \mathbf{H}'_5$ be these hybrid experiments and $S'_i$ be the event that $\mathcal{A}$ wins in experiment $\mathbf{H}'_i$. Hybrid experiments $\mathbf{H}'_0$, $\mathbf{H}'_1$, and $\mathbf{H}'_2$ are the same as $\mathbf{H}_0$, $\mathbf{H}_1$, and $\mathbf{H}_2$ in $E_1$ respectively.

*Hybrid experiment $\mathbf{H}'_3$.* The computation of $K^*_E$ in the test session is changed. Instead of computing $(K^*_E, C^*_E) \leftarrow \mathsf{wEnCap}(ek^*_E, r_B)$, it is changed as choosing $K^*_E \in_R \mathcal{KS}_{RH}$ randomly.

We construct an IND-CPA adversary $\mathcal{B}$ from $\mathcal{A}$ in $\mathbf{H}'_2$ or $\mathbf{H}'_3$. $\mathcal{B}$ simulates obeying the scheme, except that $\mathcal{B}$ sets $K^*_E = K^*$ for $\mathsf{Send}(\Pi, \mathcal{R}, T^*, ID_B, ID_A, (C^*_A, ek^*_E))$ and $\mathsf{Send}(\Pi, \mathcal{I}, T^*, ID_A, ID_B, (C^*_A, ek^*_E), (C^*_B, C^*_E))$. From $\mathcal{A}$'s point of view, the simulation by $\mathcal{B}$ is same as $\mathbf{H}'_2$ if $K^*$ received in the challenge is the real key from $\mathsf{wEnCap}$. Otherwise, the simulation by $\mathcal{B}$ is same as $\mathbf{H}'_3$. Thus, $|\Pr[S'_2 \mid E_2] - \Pr[S'_3 \mid E_2]|$ is negligible for $\kappa$ since the KEM is IND-CPA secure.

*Hybrid experiments $\mathbf{H}'_4$ and $\mathbf{H}'_5$.* Hybrid experiments $\mathbf{H}'_4$ and $\mathbf{H}'_5$ are similar to $\mathbf{H}_5$ and $\mathbf{H}_6$ in $E_1$ respectively, except that the computation of $K'^*_E$ and $F(K'^*_E, ST)$ are changed. Therefore, both $|\Pr[S_3 \mid E_2] - \Pr[S_4 \mid E_2]|$ and $|\Pr[S_4 \mid E_2] - \Pr[S_5 \mid E_2]|$ are also negligible for $\kappa$ in the same way as $E_1$.

The session key in the test session is perfectly randomized in $\mathbf{H}_5$. We have $\Pr[S_5 \mid E_2] = 1/2$. Thus, $|2\Pr[Suc \mid E_2] - 1|$ is negligible for $\kappa$.

## B Estimation

In this appendix, we estimate the performance of our proposed protocol by using our cryptographic library. Our software cryptographic library is written in C, using OpenSSL C library for operations of a multiple precision integer. We used the Gallant–Lambert–Vanstone (GLV) [15] and Galbraith–Lin–Scott (GLS) [14] techniques for the scalar multiplication. We also applied the optimal ate pairing on Barreto-Naehrig curve to the pairing operation. In this work we chose the parameters at the 128-bit security level.

**Table 2.** Execution Environment

| | |
|---|---|
| CPU | ARMv8 Cortex-A53 |
| Clock | 1.2 GHz |
| RAM | 1GB |
| Development Board | Raspberry Pi3 |
| OS | 32-bit Raspbian |

**Table 3.** Experimental results (msec)

| | |
|---|---|
| Scalar Mult. on $G_1$ | 9.838 |
| Scalar Mult. on $G_2$ | 18.661 |
| Pairing | 57.088 |

We summarize our execution environment for our experiment in Table 2. Table 3 contains the average time (in milliseconds) of 100 iterations, and also shows the timing of computing pairing and scalar multiplication on $G_1$, $G_2$. An estimated total time for instantiations of RHIB-AKE in Table 1 is within 1 second when the hierarchical level $\ell$ is small (i.e., $\ell = 2$ or 3). In conclusion, if the device is the same spec as the Raspberry Pi3, using RHIB-AKE is quite practical.

$\mathsf{Send}(\Pi, \mathcal{I}, T_{cu}, ID_\alpha, ID_\beta)$ :
- if $(T_{cu} = T^*) \wedge (\alpha = A) \wedge (\beta = B) \wedge (s_A = i-1)$
  - $s_A \leftarrow s_A + 1$
  - $(K^*, C_A^*) \leftarrow \mathsf{Challenge}(ID_B, T^*)$
  - $r_A^* \leftarrow \mathcal{R}_{wK}$
  - $(ek_E^*, dk_E^*) \leftarrow \mathsf{wKeyGen}(1^\kappa, r_A^*)$
  - $\mathcal{L}_S \leftarrow \mathcal{L}_S \cup \{(\Pi, T^*, ID_A, ID_B, C_A^*, ek_E^*)\}$
  - return $(ID_A, ID_B, T^*, C_A^*, ek_E^*)$
- else
  - $(ESK_\alpha, EPK_\alpha) \leftarrow \mathsf{InitEK}(MPK, T_{cu}, ID_\alpha, SSK_\alpha, CSK_{\alpha,T}, ID_\beta)$
  - $\mathcal{L}_S \leftarrow \mathcal{L}_S \cup \{(\Pi, T_{cu}, ID_\alpha, ID_\beta, C_\alpha, ek_E)\}$
  - if $(\alpha = A)$
    - $s_A \leftarrow s_A + 1$
  - return $EPK_\alpha$

$\mathsf{Send}(\Pi, \mathcal{R}, T_{cu}, ID_\beta, ID_\alpha, (C_\alpha, ek_E))$ :
- if $(T_{cu} \geq T^*) \wedge (\beta = B)$
  - $(ESK_B, EPK_B) \leftarrow \mathsf{ResEK}(MPK, T_{cu}, ID_B, SSK_B, CSK_{B,T_{cu}}, ID_\alpha, EPK_\alpha)$
  - $\mathcal{L}_{NCS} \leftarrow \mathcal{L}_{NCS} \cup \{(\Pi, T_{cu}, ID_B, ID_\alpha, C_\alpha, ek_E, C_B, C_E)\}$ /* $\mathcal{B}$ cannot compute the session key since $ID_B$ has been revoked.*/
  - return $EPK_B$
- else
  - $(ESK_\beta, EPK_{beta}) \leftarrow \mathsf{ResEK}(MPK, T_{cu}, ID_\beta, SSK_\beta, CSK_{\beta,T_{cu}}, ID_\alpha, EPK_\alpha)$
  - if $\exists ID \succ ID_\beta (ID \in RL_{pa(ID),T_{cu}-1})$
    - $\mathcal{L}_{NCS} \leftarrow \mathcal{L}_{NCS} \cup \{(\Pi, T_{cu}, ID_\beta, ID_\alpha, C_\alpha, ek_E, C_\beta, C_E)\}$
  - else
    - $SK \leftarrow \mathsf{ResSK}(MPK, T_{cu}, ID_\beta, SSK_\beta, CSK_{\beta,T_{cu}}, ID_\alpha, ESK_\beta, EPK_\beta, EPK_\alpha)$
    - $\mathsf{sid} \leftarrow (\Pi, T_{cu}, ID_\beta, ID_\alpha, C_\alpha, ek_E, C_\beta, C_E)$
    - $\mathcal{L}_{SK} \leftarrow \mathcal{L}_{SK} \cup \{(\mathsf{sid}, SK)\}$
  - return $EPK_\beta$

$\mathsf{SKReveal}(\mathsf{sid})$ :
- if $(\mathsf{sid}, SK) \in \mathcal{L}_{SK}$
  - return $SK$
- else
  - return $\perp$

$\mathsf{ESKReveal}(\mathsf{sid})$ :
- parse $\mathsf{sid} =: (*, *, ID_\alpha, *, *, *)$
- find $ESK_\alpha$
- return $ESK_\alpha$

$\mathsf{Send}(\Pi, \mathcal{I}, T_{cu}, ID_\alpha, ID_\beta, (C_\alpha, ek_E), (C_\beta, C_E))$ :
- if $((\Pi, T_{cu}, ID_\alpha, ID_\beta, C_\alpha, ek_E) \notin \mathcal{L}_S) \vee (\perp \leftarrow \mathsf{DecKeyReveal}(ID_\alpha, T_{cu}))$
  - $\mathcal{L}_{NCS} \leftarrow \mathcal{L}_{NCS} \cup \{(\Pi, T_{cu}, ID_\alpha, ID_\beta, C_\alpha, ek_E, C_\beta, C_E)\}$
  - return $\perp$
- else if $(T_{cu} = T^*) \wedge (\alpha = A) \wedge (\beta = B) \wedge (s_A = i)$
  - $K_A^* \leftarrow K^*$
  - $K_B^* \leftarrow \mathsf{DeCap}(mpk, dk_{A,T^*}, C_B^*)$
  - $K_E^* \leftarrow \mathsf{wDeCap}(dk_E^*, C_E^*)$
  - for $X = A, B, E$
    - $K'^*_X \leftarrow KDF(s, K_X^*)$
  - $SK \leftarrow F(K'^*_A, ST) \oplus F(K'^*_B, ST) \oplus F(K'^*_E, ST)$
  - $\mathsf{sid} \leftarrow (\Pi, T^*, ID_B, ID_A, C_A^*, ek_E^*, C_B^*, C_E^*)$
  - $\mathcal{L}_{SK} \leftarrow \mathcal{L}_{SK} \cup \{(\mathsf{sid}, SK)\}$
- else
  - $SK \leftarrow \mathsf{InitSK}(MPK, T_{cu}, ID_\alpha, SSK_\alpha, CSK_{\alpha,T_{cu}}, ID_\beta, ESK_\alpha, EPK_\alpha, EPK_\beta)$
  - $\mathsf{sid} \leftarrow (\Pi, T_{cu}, ID_\alpha, ID_\beta, C_\alpha, ek_E, C_\beta, C_E)$
  - $\mathcal{L}_{SK} \leftarrow \mathcal{L}_{SK} \cup \{(\mathsf{sid}, SK)\}$
- return SUCCESS

$\mathsf{SSKReveal}(ID_\alpha)$ :
- $sk_\alpha \leftarrow \mathsf{SecKeyReveal}(ID_\alpha)$
- return $sk_\alpha$

$\mathsf{CSKReveal}(ID_\alpha, T)$ :
- $dk_{\alpha,T} \leftarrow \mathsf{DecKeyReveal}(ID_\alpha, T)$
- $\sigma_{\alpha,T} \leftarrow LD_\kappa$
- return $(dk_{\alpha,T}, \sigma_{\alpha,T})$

$\mathsf{EstablishParty}(\alpha, ID_\alpha)$ :
- if $(pa(ID_\alpha) \in \mathcal{ID}) \wedge (ID_\alpha \notin \mathcal{ID})$
  - issue $\mathsf{SecKeyGen}(ID_\alpha)$
  - $sk_\alpha \leftarrow \mathsf{SecKeyReveal}(ID_\alpha)$
  - $\mathcal{ID} \leftarrow \mathcal{ID} \cup \{ID_\alpha\}$
  - return $sk_\alpha$

$\mathsf{Revoke}(RL)$ :
- $\{ku_{ID,T_{cu}}\}_{ID \in \in \mathcal{ID} \setminus RL} \leftarrow \mathsf{Update}(RL)$
- return $\{ku_{ID,T_{cu}}\}_{ID \in \in \mathcal{ID} \setminus RL}$

$\mathsf{Test}(\mathsf{sid})$ :
- $(\mathsf{sid}, SK_1) \leftarrow \mathcal{L}_{SK}$
- $SK_0 \leftarrow \{0,1\}^\kappa$
- $b \leftarrow \{0,1\}$
- return $SK_b$

**Fig. 3.** Query Simulation