Unbounded Hierarchical Identity-Based Encryption with Efficient Revocation

Geumsook Ryu* Kwangsu Lee† Seunghwan Park‡ Dong Hoon Lee§

Abstract

Hierarchical identity-based encryption (HIBE) is an extension of identity-based encryption (IBE) where an identity of a user is organized as a hierarchical structure and a user can delegate the private key generation to another user. Providing a revocation mechanism for HIBE is highly necessary to keep a system securely. Revocable HIBE (RHIBE) is an HIBE scheme that can revoke a user's private key if his credential is expired or revealed. In this paper, we first propose an unbounded HIBE scheme where the maximum hierarchy depth is not limited and prove its selective security under a q-type assumption. Next, we propose an efficient unbounded RHIBE scheme by combining our unbounded HIBE scheme and a binary tree structure, and then we prove its selective security. By presenting the unbounded RHIBE scheme, we solve the open problem of Seo and Emura in CT-RSA 2015.

Keywords: Identity-based encryption, Hierarchical identity-based encryption, Revocation, Unbounded hierarchy depth, Bilinear maps

*Korea University, Korea. Email: madeby_r@korea.ac.kr.
†Korea University, Korea. Email: kwangsu.lee@korea.ac.kr.
‡Korea University, Korea. Email: sgusa@korea.ac.kr.
§Korea University, Korea. Email: donghlee@korea.ac.kr.

1 Introduction

Identity-based encryption (IBE) is a kind of public key encryption (PKE) that uses any bit-string (e.g., e-mail address, phone number, or identity) as a public key of a user. Although the concept of IBE was introduced by Shamir [26], the first realization of IBE was achieved by Boneh and Franklin [4] by using bilinear maps. In IBE, a single key generation center (KGC) should issue private keys and establish secure channels to transmit private keys of users. To reduce the cost of private key generation of the KGC in IBE, the concept of hierarchical IBE (HIBE) was introduced such that the KGC delegates the key generation functionality to a lower level KGC [8, 9]. After that, many IBE and HIBE schemes were suggested with additional functionalities [2, 3, 5, 15, 27, 28].

To maintain a whole system securely, a revocation mechanism is absolutely necessary when a user's contract is expired or the user's private key is revealed. Boldyreva, Goyal and Kumar [1] introduced the concept of revocable IBE (RIBE) and proposed a scalable RIBE scheme by combining a fuzzy IBE scheme of Sahai Waters [21] and a tree based revocation system of Naor et al. [17]. In RIBE, each user initially obtains a private key from a KGC, and then the KGC periodically publishes an update key for non-revoked users. If a user is not revoked in the update key, then he can derive a decryption key from his private key and the update key. After the work of Boldyreva et al., many different RIBE scheme were proposed [12, 16, 18, 23].

It is a natural research direction to devise an efficient revocation mechanism for HIBE. By following the design strategy of Boldyreva et al. [1], Seo and Emura proposed efficient revocable HIBE (RHIBE) schemes [22, 24]. In RHIBE, a KGC can delegate the key generation functionality and the revocation functionality to a lower level KGC or a user. Seo and Emura [22] first proposed a concrete RHIBE scheme by combining the HIBE scheme of Boneh and Boyen and a binary tree structure. After that, they also proposed new efficient RHIBE schemes by using the history-free update approach to reduce the size of private keys [24]. Although they proposed efficient RHIBE schemes, their RHIBE schemes have the inherent limitation that the size of public parameters linearly grows to the maximum hierarchy depth. Thus, they left it as an interesting problem to devise an unbounded RHIBE scheme [24].

1.1 Our Contributions

In this paper, we give an answer to the above problem of Seo and Emura by presenting an unbounded RHIBE scheme. Before presenting an unbounded RHIBE scheme, we first propose an HIBE scheme with no limitation in maximum hierarchy, denoted by unbounded HIBE. Our unbounded HIBE scheme is derived from the key-policy attribute-based encryption (KP-ABE) scheme of Rouselakis and Waters [19]. We use the observation that an HIBE scheme can be derived from a KP-ABE scheme if the KP-ABE scheme can be modified to support the delegation of private key generation. We prove the selective security of our unbounded HIBE scheme under the q-type assumption introduced by Rouselakis and Waters. Next, we propose an unbounded RHIBE scheme by combining our unbounded HIBE scheme and a tree-based revocation system. Mainly we follow the design strategy of the previous RHIBE scheme of Seo and Emura [24]. To prove the selective security of our RHIBE scheme, we show that our RHIBE scheme is selectively secure if our HIBE scheme is selectively secure.

1.2 Related Work

IBE and Its Extensions. As mentioned before, the concept of IBE was introduced by Shamir [26] where a public key can be the identity string of a user such as an e-mail address. The first IBE scheme that uses

bilinear maps was constructed by Boneh and Franklin [4]. Since the pioneering work of Boneh and Franklin, many IBE schemes were proposed in bilinear maps [2, 6, 27]. The notion of IBE has been extended to several other encryption systems like HIBE [9], attribute-based encryption (ABE) [21], predicate encryption (PE), and functional encryption (FE). The concept of HIBE was introduced by Horwitz and Lynn [9] and it additionally provides a key delegation mechanism by which the private key of a low level user is generated by a upper level user. After the introduction of HIBE, many HIBE schemes with different properties have been suggested in bilinear maps [2, 3, 5, 7, 8, 13, 14, 25, 28]. One inherent limitation of previous HIBE schemes is that the maximum hierarchy depth should be fixed in the setup phase. To remove this restriction, an unbounded HIBE scheme was proposed by Lewko and Waters [15].

Revocation in IBE. Boneh and Franklin [4] proposed the first IBE scheme that supports key revocation, but their scheme is not scalable since each user periodically connects to a KGC to receive a new private key. Boldyreva et al. [1] proposed a scalable RIBE scheme by combining the fuzzy IBE scheme of Sahai and Waters [21] and the tree based revocation system of Naor et al. [17]. Libert and Vergnaud [16] proposed first fully secure RIBE scheme by using a fully secure IBE scheme that is a variant of the Waters IBE [27]. Seo and Emura [23] refined the security model of RIBE by considering decryption key exposure attacks and proposed a fully secure RIBE scheme in their security model. To improve the efficiency of RIBE, Lee et al. [12] proposed a new RIBE scheme based on the subset difference method and Park et al. [18] proposed an RIBE scheme from multilinear maps. An efficient RHIBE scheme was first presented by Seo and Emura [22] and its improvement was also proposed by using the history-free update approach [24]. In RIBE, revoked user on the time T is still accessible to ciphertext that were encrypted before the time T in the cloud storage environment. To solve this problem, Sahai et al. [20] introduced revocable storage ABE (RS-ABE) for cloud storage by using the idea of RIBE. The improved RS-ABE schemes were presented in [10, 11].

2 Preliminaries

In this section, we introduce the complexity assumption for our schemes and define the syntax of RHIBE and its security model.

2.1 Bilinear Groups

Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative cyclic groups of prime order p and g be a generator of $\mathbb{G}$. The bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ has the following properties:

- Bilinearity: for all $u, v \in \mathbb{G}$ and for all $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$
- Non-degeneracy: for generator $g \in \mathbb{G}$, $e(g, g) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ is an identity element in $\mathbb{G}_T$

Furthermore, we assume the existence of a group generator algorithm $\mathcal{G}$ which takes as input a security parameter λ and outputs a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e)$ where p is a prime of $\Theta(\lambda)$ bits.

2.2 Complexity Assumption

For the proof of our schemes, we introduce the q-RW2 assumption of Rouselakis and Waters [19] that was used to prove the security of their attribute-based encryption schemes.

Assumption 2.1 (q-RW2, [19])**.** *Let $(p,\mathbb{G},\mathbb{G}_T,e)$ be a description of the bilinear grouops of prime order p. Let g be a random generator of $\mathbb{G}$. The q-RW2 assumption is that if the challenge tuple*

$$D = \Big((p,\mathbb{G},\mathbb{G}_T,e), g, g^x, g^y, g^z, g^{(xz)^2}, \{g^{b_i}, g^{xzb_i}, g^{xz/b_i}, g^{x^2zb_i}, g^{y/b_i^2}, g^{y^2/b_i^2}\}_{\forall\, i\in[q]},$$
$$\{g^{xzb_i/b_j}, g^{yb_i/b_j^2}, g^{xyzb_i/b_j^2}, g^{(xz)^2b_i/b_j}\}_{\forall\, i,j\in[q], i\neq j}\Big) \text{ and } Z$$

are given, no probabilistic polynomial time (PPT) algorithm $\mathcal{A}$ can distinguish $Z = Z_0 = e(g,g)^{xyz}$ from $Z = Z_1 = e(g,g)^f$ with more than a negligible advantage. The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}_{\mathcal{A}}^{q\text{-}RW2}(\lambda) = \big|\Pr[\mathcal{A}(D,Z_0)=0] - \Pr[\mathcal{A}(D,Z_1)=0]\big|$ where the probability is taken over random choices of $x,y,z,\{b_i\}_{i\in[q]}, f \in \mathbb{Z}_p$.

Lemma 2.2 ([19])**.** *The q-RW2 assumption holds in the generic group model.*

2.3 Hierarchical IBE

HIBE is an extension of IBE where an identity of a user is represented as a hierarchical structure such as $ID|_k = (I_1,\ldots,I_k)$ [8]. In HIBE, a user with an identity $ID|_k$ can receives his private key $SK_{ID|_k}$ from a KGC and he can also delegate his private key to lower level users with an identity $ID|_{k+1} = (I_1,\ldots,I_k,I_{k+1})$. A ciphertext of HIBE is associated with a receiver's identity $ID|_\ell$ and a user with a private key $SK_{ID|_k}$ can decrypt this ciphertext if $ID|_k$ is a prefix of $ID|_\ell$. The syntax of HIBE is given as follows:

Definition 2.3 (HIBE)**.** *An HIBE scheme consists of five algorithms **Setup**, **GenKey**, **Delegate**, **Encrypt**, and **Decrypt**, which are defined as follows:*

- ***Setup**(1^λ). The setup algorithm takes as input a security parameter 1^λ. It outputs a master key MK and public parameters PP.*
- ***GenKey**($ID|_k, MK, PP$). The key generation algorithm takes as input an identity $ID|_k = (I_1,\ldots,I_k) \in \mathcal{I}^k$, the master key MK, and the public parameters PP. It outputs a private key $SK_{ID|_k}$ for $ID|_k$.*
- ***Delegate**($ID|_k, SK_{ID|_{k-1}}, PP$). The delegation algorithm takes as input an identity $ID|_k$, a private key $SK_{ID|_{k-1}}$ for an identity $ID|_{k-1}$, and the public parameters PP. It outputs a delegated private key $SK_{ID|_k}$ for $ID|_k$.*
- ***Encrypt**($ID|_k, M, PP$). The encryption algorithm takes as input an identity $ID|_k$, a message $M \in \mathcal{M}$, and the public parameters PP. It outputs a ciphertext $CT_{ID|_k}$ for $ID|_k$ and M.*
- ***Decrypt**($CT_{ID|_k}, SK_{ID'|_\ell}, PP$). The decryption algorithm takes as input a ciphertext $CT_{ID|_k}$ for an identity $ID|_k$, a private key $SK_{ID'_\ell}$ for an identity ID'_ℓ, and the public parameters PP. It outputs an encrypted message M.*

*The correctness of HIBE is defined as follows: For all MK, PP generated by **Setup**, all $ID|_k, ID'|_\ell$, any $SK_{ID|_k}$ generated by **GenKey**, and any M, it is required that*

- *If $ID|_k$ is a prefix of $ID'|_\ell$, then $\mathbf{Decrypt}(\mathbf{Encrypt}(ID'|_\ell, M, PP), SK_{ID|_k}, PP) = M$.*
- *If $ID|_k$ is not a prefix of $ID'|_\ell$, then $\mathbf{Decrypt}(\mathbf{Encrypt}(ID'|_\ell, M, PP), SK_{ID}, PP) = \perp$.*

The security model of HIBE that provides collusion resistance was defined by Gentry and Silverberg [8]. We follow the selective security model of Boneh and Boyen [2]. In this model, an adversary initially submits a challenge identity $ID^*|_\ell$ and he may request private key queries for some identities that are not a prefix of $ID^*|_\ell$. In the challenge step, the adversary submits two challenge messages and receives a challenge ciphertext. Finally, the adversary outputs a guess of an encrypted message in the challenge ciphertext. The adversary wins the game if he correctly guesses the encrypted message. The detailed definition of the security model is given as follows:

Definition 2.4 (Selective IND-CPA Security)**.** *The selective IND-CPA security of HIBE is defined in terms of the following experiment between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$:*

1. ***Init**: $\mathcal{A}$ initially submits a challenge identity $ID^*|_\ell$.*
2. ***Setup**: $\mathcal{C}$ runs **Setup**(1^λ) to generate a master key MK and public parameters PP. It keeps MK to itself and gives PP to $\mathcal{A}$.*
3. ***Phase 1**: $\mathcal{A}$ may adaptively request a polynomial number of private key queries. If this is a private key query for an identity $ID|_k$ with the restriction that $ID|_k$ is not a prefix of $ID^*|_\ell$, then it creates a private key $SK_{ID|_k}$ by calling **GenKey**$(ID|_k, MK, PP)$.*
4. ***Challenge**: $\mathcal{A}$ submits two challenge messages M_0^*, M_1^* with equal length. $\mathcal{C}$ flips a random coin $\mu \in \{0,1\}$ and gives the challenge ciphertext CT^* to $\mathcal{A}$ by running **Encrypt**$(ID^*|_\ell, M_\mu^*, PP)$.*
5. ***Phase 2**: $\mathcal{A}$ continues to request a polynomial number of private key queries subject to the restriction as before.*
6. ***Guess**: $\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$ of μ, and wins the game if $\mu = \mu'$.*

The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}_{\mathcal{A}}^{HIBE}(\lambda) = \left| \Pr[\mu = \mu'] - \frac{1}{2} \right|$ where the probability is taken over all the randomness of the experiment. An HIBE scheme is selectively secure under a chosen plaintext attack if for all PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the above experiment is negligible in the security parameter λ.

2.4 Revocable HIBE

RHIBE is an extension of HIBE that provides revocation functionality [22]. In RHIBE, a user with an identity $ID|_{k-1}$ can generate a private key SK for any lower level users and he also broadcasts an update key $UK_{T,R}$ for non-revoked users per each time period T where R is the set of revoked users. A lower level user with an identity $ID|_k$ can derive a decryption key $DK_{ID|_k,T}$ from his private key $SK_{ID|_k}$ and the update key $UK_{T,R}$ if his private key is not revoked in the update key. By using $DK_{ID|_k,T}$, the user can decrypt a ciphertext that matches to his decryption key. The syntax of RHIBE is given as follows:

Definition 2.5 (Revocable HIBE)**.** *An RHIBE scheme for the identity space $\mathcal{I}$, the time space $\mathcal{T}$, and the message space $\mathcal{M}$, consists of seven algorithms **Setup**, **GenKey**, **UpdateKey**, **DeriveKey**, **Encrypt**, **Decrypt**, and **Revoke**, which are defined as follows:*

***Setup**(1^λ): This algorithm takes as input a security parameter 1^λ. It outputs a master key MK, an (empty) revocation list RL, a state information ST, and public parameters PP.*

***GenKey**$(ID|_k, ST_{ID|_{k-1}}, PP)$: This algorithm takes as input an identity $ID|_k = (I_1, \ldots, I_k) \in \mathcal{I}^k$, the state $ST_{ID|_{k-1}}$, and public parameters PP. It outputs a private key $SK_{ID|_k}$.*

UpdateKey$(T, RL_{ID|_{k-1}}, DK_{ID|_{k-1},T}, ST_{ID|_{k-1}}, PP)$*: This algorithm takes as input time* $T \in \mathcal{T}$*, the revocation list* $RL_{ID|_{k-1}}$*, the decryption key* $DK_{ID|_{k-1},T}$*, and public parameters PP. It outputs an update key* $UK_{ID|_{k-1},T}$*.*

DeriveKey$(SK_{ID|_k}, UK_{ID|_{k-1},T}, PP)$*: This algorithm takes as input a private key* $SK_{ID|_k}$ *for an identity* $ID|_k$*, an update key* $UK_{ID|_{k-1},T}$ *for time* T*, and the public parameters PP. It outputs a decryption key* $DK_{ID|_k,T}$*.*

Encrypt$(ID|_\ell, T, M, PP)$*: This algorithm takes as input an identity* $ID|_\ell = (I_1, \ldots, I_\ell) \in \mathcal{I}^\ell$*, time* T*, a message* M*, and the public parameters PP. It outputs a ciphertext* $CT_{ID|_\ell,T}$*.*

Decrypt$(CT_{ID|_\ell,T}, DK_{ID'|_k,T'}, PP)$*: This algorithm takes as input a ciphertext* $CT_{ID|_\ell,T}$*, a decryption key* $DK_{ID'|_k,T'}$ *and the public parameters PP. It outputs an encrypted message* M*.*

Revoke$(ID|_k, T, RL_{ID|_{k-1}}, ST_{ID|_{k-1}})$*: This algorithm takes as input an identity* $ID|_k$*, revocation time* T*, the revocation list* $RL_{ID|_{k-1}}$*, and the state* $ST_{ID|_{k-1}}$*. It outputs the updated revocation list* $RL_{ID|_{k-1}}$*.*

The correctness of RHIBE is defined as follows: For all MK, RL, ST, and PP generated by ***Setup***(1^λ)*,* SK_{ID} *generated by* ***GenKey***(ID, MK, ST, PP) *for any ID,* $UK_{T,R}$ *generated by* ***UpdateKey***(T, RL, MK, ST, PP) *for any* T *and RL,* $CT_{ID',T'}$ *generated by* ***Encrypt***(ID', T', M, PP) *for any* ID'*,* T'*, and M, it is required that*

- *If* $ID|_k$ *is not revoked on time* T*, then* ***DeriveKey***$(SK_{ID|_k}, UK_{ID|_{k-1},T}, PP) = DK_{ID|_k,T}$*.*
- *If* $ID|_k$ *is revoked on time* T*, then* ***DeriveKey***$(SK_{ID|_k}, UK_{ID|_{k-1},T}, PP) = \perp$*.*
- *If* $(ID' = ID) \wedge (T' = T)$*, then* ***Decrypt***$(CT_{ID',T'}, DK_{ID,T}, PP) = M$*.*
- *If* $(ID' \neq ID) \vee (T' \neq T)$*, then* ***Decrypt***$(CT_{ID',T'}, DK_{ID,T}, PP) = \perp$*.*

The security model of RHIBE was introduced by Seo and Emura [22]. We follow the stronger security model of Seo and Emura [24] that considers decryption key exposure attackers and inside attackers. In this model, an adversary initially submits a challenge identity, a challenge time. After that, the adversary can request private key, update key, decryption key, and revocation queries with some restrictions to prevent obvious attacks. In the challenge step, the adversary submits two challenge messages and receives a challenge ciphertext that is an encryption of one challenge message. The adversary wins the game if he correctly guesses the encrypted message. The detailed definition of the security model is given as follows:

Definition 2.6 (Selective IND-CPA Security)**.** *The selective IND-CPA security of RHIBE is defined in terms of the following experiment between a challenger* $\mathcal{C}$ *and a PPT adversary* $\mathcal{A}$*:*

1. ***Init****:* $\mathcal{A}$ *initially submits a challenge identity* $ID^*|_k = (I_1^*, \ldots, I_k^*)$ *and challenge time* T^**.*
2. ***Setup****:* $\mathcal{C}$ *runs* ***Setup***(1^λ) *and obtains a master key MK, a revocation list RL, a state information ST, and public parameters PP. It keeps* MK, RL, ST *to itself and gives PP to* $\mathcal{A}$*.*
3. ***Phase 1****:* $\mathcal{A}$ *adaptively requests a polynomial number of queries. These queries are processed as follows:*
   - *If it is a private key query for an identity* $ID|_k$*, then* $\mathcal{C}$ *gives a private key* $SK_{ID|_k}$ *and a state information* $ST_{ID|_k}$ *by running* ***GenKey***$(ID|_k, ST_{ID|_{k-1}}, PP)$*. There is a restriction: If* $\mathcal{A}$ *requested a private key query for* $ID^*|_{k'}$ *that is a prefix of* $ID^*|_k$ *where* $k' \leq k$*, then the identity* $ID^*|_{k'}$ *or one of its ancestors should be revoked at some time* T *where* $T \leq T^*$*.*

- *If it is an update key query for an identity $ID|_{k-1}$ and time T, then $\mathcal{C}$ gives an update key $UK_{ID|_{k-1},T}$ by running* ***UpdateKey***$(T, RL_{ID|_{k-1}}, DK_{ID|_{k-1}}, ST_{ID|_{k-1}}, PP)$.
- *If it is a decryption key query for an identity $ID|_k$ and time T, then $\mathcal{C}$ gives a decryption key $DK_{ID|_k,T}$ by running* ***DeriveKey***$(SK_{ID|_k}, UK_{ID|_{k-1}}, PP)$. *There is a restriction: $\mathcal{A}$ cannot request a private key query for the challenge identity $ID^*|_k$ or its ancestors on the challenge time T^*.*
- *If it is a revocation query for an identity $ID|_k$ and time T, then $\mathcal{C}$ updates a revocation list $RL_{ID|_{k-1}}$ by running* ***Revoke***$(ID|_k, T, RL_{ID|_{k-1}}, ST_{ID|_{k-1}})$. *There is a restriction: $\mathcal{A}$ cannot request a revocation query for $ID|_k$ on time T if he already requested an update key query for $ID|_k$ on time T.*

Note that we assume that update key, decryption key, and revocation queries are requested in non-decreasing order of time.

4. ***Challenge****: $\mathcal{A}$ submits two challenge messages M_0^*, M_1^* with the same length. $\mathcal{C}$ flips a random coin $\mu \in \{0,1\}$ and gives the challenge ciphertext $CT_{ID^*|_k,T^*}$ to $\mathcal{A}$ by running* ***Encrypt***$(ID^*|_\ell, T^*, M_\mu^*, PP)$.
5. ***Phase 2****: $\mathcal{A}$ may continue to request a polynomial number of queries subject to the same restrictions as before.*
6. ***Guess****: Finally, $\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$, and wins the game if $\mu = \mu'$.*

The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}_{\mathcal{A}}^{RHIBE}(\lambda) = \left|\Pr[\mu = \mu'] - \frac{1}{2}\right|$ *where the probability is taken over all the randomness of the experiment. An RHIBE scheme is selectively secure under a chosen plaintext attack if for all PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the above experiment is negligible in the security parameter λ.*

3 Hierarchical Identity-Based Encryption

In this section, we propose an unbounded HIBE scheme from the key-policy ABE scheme of Rouselakis and Waters [19] and prove its security.

3.1 Construction

Let $\mathcal{I} = \{0,1\}^\lambda$ be the identity space where λ is a security parameter. Our unbounded HIBE scheme is described as follows:

HIBE.Setup(1^λ)**:** This algorithm takes as input a security parameter λ. It first runs the group generator $\mathcal{G}$ and obtains a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e)$. Let g be a generator of $\mathbb{G}$. Next, it selects random elements $g, u, h \in \mathbb{G}$ and random exponents $x, y \in \mathbb{Z}_p$. It sets $w = g^x, v = g^y, \alpha = xy$. It outputs a master key $MK = \alpha$ and public parameters

$$PP = \Big((p, \mathbb{G}, \mathbb{G}_T, e),\ g,\ u,\ h,\ w,\ v,\ \Omega = e(g,g)^\alpha\Big).$$

HIBE.GenKey$(ID|_k, MK, PP)$**:** This algorithm takes as input an identity $ID|_k = (I_1, \ldots, I_k) \in \mathcal{I}^k$, the master key MK, and the public parameters PP. It chooses random exponents $r_1, \ldots, r_k \in \mathbb{Z}_p$ and outputs a private key

$$SK_{ID|_k} = \Big(K_0 = g^\alpha \prod_{i=1}^{k} w^{r_i},\ \big\{K_{i,1} = (u^{I_i} h)^{-r_i},\ K_{i,2} = g^{r_i}\big\}_{i=1}^{k}\Big).$$

HIBE.RandKey$(ID|_k, \gamma, SK_{ID|_k}, PP)$**:** This algorithm takes as input an identity $ID|_k = (I_1, \ldots, I_k) \in \mathcal{I}^k$, an exponent $\gamma \in \mathbb{Z}_p$, a private key $SK_{ID|_k} = (K'_0, \{K'_{i,1}, K'_{i,2}\}_{i=1}^k)$, and the public parameters PP. It chooses random exponents $r_1, \ldots, r_k \in \mathbb{Z}_p$ and outputs a re-randomized private key

$$SK_{ID|_k} = \Big(K_0 = K'_0 \cdot g^{\gamma} \prod_{i=1}^{k} w^{r_i},\ \big\{K_{i,1} = K'_{i,1} \cdot (u^{I_i} h)^{-r_i},\ K_{i,2} = K'_{i,2} \cdot g^{r_i}\big\}_{i=1}^{k}\Big).$$

HIBE.Delegate$(ID|_k, SK_{ID|_{k-1}}, PP)$**:** This algorithm takes as input an identity $ID|_k = (I_1, \ldots, I_k) \in \mathcal{I}^k$, a private key $SK_{ID|_{k-1}} = (K'_0, \{K'_{i,1}, K'_{i,2}\}_{i=1}^{k-1})$ for $ID|_{k-1}$, and the public parameters PP. It chooses a random exponent $r_k \in \mathbb{Z}_p$ and creates a temporal delegated private key

$$TSK_{ID|_k} = \Big(K_0 = K'_0 \cdot w^{r_k},\ \big\{K_{i,1} = K'_{i,1},\ K_{i,2} = K'_{i,2}\big\}_{i=1}^{k-1},\ \big\{K_{k,1} = (u^{I_k} h)^{-r_k}, K_{k,2} = g^{r_k}\big\}\Big).$$

Next, it outputs a delegated private key $SK_{ID|_k}$ by running **HIBE.RandKey**$(ID|_k, 0, TSK_{ID|_k}, PP)$.

HIBE.Encrypt$(ID|_\ell, M, PP)$**:** This algorithm takes as input an identity $ID|_\ell = (I_1, \ldots, I_\ell) \in \mathcal{I}^\ell$, a message $M \in \mathcal{M}$, and the public parameters PP. It chooses random exponents $t, s_1, \ldots, s_k \in \mathbb{Z}_p$ and outputs a ciphertext

$$CT_{ID|_\ell} = \Big(C = \Omega^t \cdot M,\ C_0 = g^t,\ \big\{C_{i,1} = g^{s_i},\ C_{i,2} = (u^{I_i} h)^{s_i} w^{-t}\big\}_{i=1}^{\ell}\Big).$$

HIBE.Decrypt$(CT_{ID|_\ell}, SK_{ID'|_k}, PP)$**:** This algorithm takes as input a ciphertext $CT_{ID|_\ell} = (C, C_0, \{C_1, C_2\}_{i=1}^{\ell})$ for $ID|_\ell$, a private key $SK_{ID'|_k} = (K_0, \{K_{i,1}, K_{i,2}\}_{i=1}^{k})$ for $ID'|_k$, and the public parameters PP. If $ID'|_k$ is a prefix of $ID|_\ell$, it outputs an encrypted message by computing

$$M = C \cdot e(C_0, K_0)^{-1} \cdot \prod_{i=1}^{k} \big(e(C_{i,1}, K_{i,1}) \cdot e(C_{i,2}, K_{i,2})\big)^{-1}.$$

Otherwise, it outputs $\perp$.

3.2 Correctness

We have to check the correctness of the scheme. Let $CT_{ID|_\ell}$ be a ciphertext for an identity $ID|_\ell$ and $SK_{ID'|_k}$ be a private key for an identity $ID'|_k$. If $ID|_\ell = ID'|_k$, then the decryption algorithm correctly computes as follows:

$$\begin{aligned} & e(C_0, K_0) \cdot \prod_{i=1}^{k} \big(e(C_{i,1}, K_{i,1}) \cdot e(C_{i,2}, K_{i,2})\big) \\ &= e(g^t, g^{\alpha} \prod_{i=1}^{k} w^{r_i}) \cdot \prod_{i=1}^{k} \big(e(g^{s_i}, (u^{I_i} h)^{-r_i}) \cdot e((u^{I_i} h)^{s_i} w^{-t}, g^{r_i})\big) = \Omega^t. \end{aligned}$$

3.3 Security Analysis

Theorem 3.1. *The above HIBE scheme is selectively IND-CPA secure if the q-RW2 assumption holds.*

Proof. Suppose that there exists an adversary $\mathcal{A}$ that attacks the above HIBE scheme with a non-negligible advantage. A simulator $\mathcal{B}$ that solves the q-RW2 assumption using $\mathcal{A}$ is given: a challenge tuple $D = \big((p,\mathbb{G},\mathbb{G}_T,e),g,g^x,g^y,g^z,g^{(xz)^2},\{g^{b_i},g^{xzb_i},g^{xz/b_i},g^{x^2zb_i},g^{y/b_i^2},g^{y^2/b_i^2}\},\{g^{xzb_i/b_j},g^{yb_i/b_j^2},g^{xyzb_i/b_j^2},g^{(xz)^2b_i/b_j}\}\big)$ and Z where $Z = Z_0 = e(g,g)^{xyz}$ or $Z = Z_1 \in_R \mathbb{G}_T$. $\mathcal{B}$ that interacts with $\mathcal{A}$ is described as follows:

Init: $\mathcal{A}$ initially submits a challenge identity $ID^*|_\ell = (I_1^*,\ldots,I_\ell^*)$ for the selective IND-CPA experiment of HIBE where $\ell \le q$.

Setup: $\mathcal{B}$ chooses random exponents $u',h' \in \mathbb{Z}_p$. It implicitly sets $\alpha = xy$ and creates public parameters PP as

$$g,\ u = g^{u'}\prod_{i=1}^{\ell} g^{y/b_i^2},\ h = g^{h'}\prod_{i=1}^{\ell}\big(g^{xz/b_i}(g^{y/b_i^2})^{-I_i^*}\big),\ w = g^x,\ v = g^y,\ \Omega = e(w,v).$$

Phase 1: $\mathcal{A}$ adaptively requests a polynomial number of private key queries. Consider a query for the private key of $ID|_k$. There exist at least one $I_j \in ID|_k (1 \le j \le k)$ such that $I_j \notin ID^*|_\ell$ by the restriction of the private key query. If $\mathcal{B}$ can generate $SK_{ID|_j}$, then it can also generate $SK_{ID|_k}$ using **HIBE.Delegate** algorithm. Therefore, $\mathcal{B}$ generates $SK_{ID|_j}$ at first. It chooses random exponents $r_1,\ldots,r_{j-1}$, $\tilde{r}_j$ from $\mathbb{Z}_p$ and set $r_j = -y + \Sigma_{i=1}^{\ell}\frac{xzb_i}{I_j-I_i^*} + \tilde{r}_j$. It computes

$$\begin{aligned}
K_0 &= \prod_{i=1}^{\ell}\big\{(g^{x^2zb_i})^{\frac{1}{I_j-I_i^*}}\cdot w^{\tilde{r}_j}\big\}\cdot\prod_{i=1}^{j-1}w^{r_i} = \prod_{i=1}^{\ell}\big\{(g^{x^2zb_i})^{\frac{1}{I_j-I_i^*}}\cdot w^{\tilde{r}_j}\big\}\cdot g^{xy}\cdot g^{-xy}\cdot\prod_{i=1}^{j-1}w^{r_i}\\
&= g^{xy}\cdot(g^x)^{-y+\Sigma_{i=1}^{\ell}\frac{xzb_i}{I_j-I_i^*}+\tilde{r}_j}\cdot\prod_{i=1}^{j-1}w^{r_i} = g^{xy}\cdot\prod_{i=1}^{j}w^{r_i},\\
\big\{K_{i,1} &= (u^{I_i}h)^{-r_i},\ K_{i,2} = g^{r_i}\big\}_{i=1}^{j-1},\\
K_{j,1} &= (v\cdot g^{-\tilde{r}_j})^{u'I_j+h'}\cdot\prod_{i=1}^{\ell}\Big[(g^{xzb_i})^{\frac{-u'I_j+h'}{I_j-I_i^*}}\cdot\big\{(g^{y^2/b_i^2})\cdot(g^{y/b_i^2})^{-\tilde{r}_j}\cdot\prod_{\substack{\tau=1\\\tau\neq i}}^{\ell}(g^{xyzb_\tau/b_i^2})^{\frac{-1}{I_j-I_\tau^*}}\big\}^{I_j-I_i^*}\cdot\\
&\quad\prod_{\tau=1}^{\ell}(g^{(xz)^2b_\tau/b_i})^{-\frac{1}{I_j-I_\tau^*}}\cdot(g^{xz/b_i})^{-\tilde{r}_j}\Big]\\
&= \Big[g^{u'I_j+h'}\cdot\prod_{i=1}^{\ell}\big\{(g^{y/b_i^2})^{I_j-I_i^*}\cdot g^{xz/b_i}\big\}\Big]^{y-\Sigma_{i=1}^{\ell}\frac{xzb_i}{I_j-I_i^*}-\tilde{r}_j} = (u^{I_j}h)^{y-\Sigma_{i=1}^{\ell}\frac{xzb_i}{I_j-I_i^*}-\tilde{r}_j} = (u^{I_j}h)^{-r_j},\\
K_{j,2} &= v^{-1}\cdot\prod_{i=1}^{\ell}(g^{xzb_i})^{\frac{1}{I_j-I_i^*}}\cdot w^{\tilde{r}_j} = g^{-y+\Sigma_{i=1}^{\ell}\frac{xzb_i}{I_j-I_i^*}+\tilde{r}_j} = g^{r_j}.
\end{aligned}$$

Next, it sets $SK_{ID|_j} = \big(K_0,\{K_{i,1},K_{i,2}\}_{i=1}^{j}\big)$ and obtains $SK_{ID|_k}$ by running **HIBE.Delegate**$(ID|_k, SK_{ID|_j}, PP)$. It gives $SK_{ID|_k}$ to $\mathcal{A}$.

Challenge: $\mathcal{A}$ submits two challenge messages M_0^*, M_1^*. $\mathcal{B}$ chooses a random bit $\mu \in \{0,1\}$ and computes for all $i \in \{1,\ldots,\ell\}$

$$\begin{aligned}
C_{i,1} &= g^{b_i},\\
C_{i,2} &= (g^{b_i})^{u'I_i^*+h'}\cdot\prod_{\substack{\tau=1\\\tau\neq i}}^{\ell}\{(g^{yb_i/b_\tau^2})^{I_i^*-I_\tau^*}\cdot g^{xzb_i/b_\tau}\} = \Big[g^{u'I_i^*+h'}\cdot\prod_{\tau=1}^{\ell}\big\{(g^{y/b_\tau^2})^{I_i^*-I_\tau^*}\cdot g^{xz/b_\tau}\big\}\Big]^{b_i}\cdot g^{-xz}\\
&= (u^{I_i^*}h)^{b_i}\cdot w^{-z}
\end{aligned}$$

$\mathcal{B}$ gives the challenge ciphertext $CT_{ID^*|_\ell} = \left(C = Z \cdot M^*_\mu, C_0 = g^z, \{C_{i,1}, C_{i,2}\}_{i=1}^{\ell}\right)$ to $\mathcal{A}$.
Phase 2: $\mathcal{A}$ may continue to request private key queries as the same as **Phase 1**.
Guess: Finally, $\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$. If $\mu = \mu'$, then $\mathcal{B}$ outputs 0. Otherwise, it outputs 1. □

4 Revocable Hierarchical Identity-Based Encryption

In this section, we propose an unbounded RHIBE scheme by using our unbounded HIBE scheme in the previous section. To provide the revocation functionality, we basically follow the design strategy of previous RIBE (or RHIBE) schemes that use a binary tree structure [1, 22, 24].

4.1 KUNode Algorithm

We use the KUNode algorithm of Boldyreva et al. [1] for our RHIBE scheme.

Definition 4.1 (KUNode Algorithm)**.** *This algorithm takes as input a binary tree BT, a revocation list RL, and time T. It outputs a set of nodes. If η is a non-leaf node, then the left and right child node of η is denoted by η_{left} and η_{right}, respectively. Users are assigned to leaf nodes, and **Path**(η) means the set of nodes on the path from η to the root node. If a user assigned to η is revoked on time T, then $(\eta, T) \in RL$. The algorithm is given below.*

***KUNode**(BT,RL,T):*
$X, Y \leftarrow \emptyset$
$\forall(\eta_i, T_i) \in RL$
  *If $T_i \leq T$ then add **Path**(η_i) to X*
$\forall x \in X$
  If $x_{left} \notin X$ then add x_{left} to Y
  If $x_{right} \notin X$ then add x_{right} to Y
If $Y \neq \emptyset$ then add root to Y
Return Y

When a user requests a private key to a KGC, the KGC assigns a user to the leaf node η of a binary tree BT, and generates a private key. A private key is associated with the set of nodes **Path**(η). The KGC publishes the update key for a set **KUNode**(BT, RL, T) at time T, then only unrevoked users have at least one node in **Path**(η) $\cap$ **KUNode**(BT, RL, ST). Unrevoked users can derive the decryption key combining the secret key and the update key in that node.

4.2 Construction

Let $\mathcal{I} = \{0,1\}^\lambda$ be the identity space and $\mathcal{T} = \{0,1\}^\lambda$ be the time space where λ is a security parameter. Our RHIBE scheme from our HIBE scheme is described as follows:

RHIBE.Setup(1^λ): This algorithm takes as input a security parameter 1^λ. It first runs the group generator $\mathcal{G}$ and obtains a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e)$. Let g be a generator of $\mathbb{G}$. Next, it selects random elements $u, h, u_0, h_0 \in \mathbb{G}$ and random exponents $x, y \in \mathbb{Z}_p$. It sets $w = g^x, v = g^y, \alpha = xy$. It outputs a master key $MK = \alpha$ and public parameters $PP = \left((p, \mathbb{G}, \mathbb{G}_T, e), g, u, h, w, v, \Omega = e(g,g)^\alpha,\ u_0, h_0\right)$.

RHIBE.GenKey$(ID|_k, ST_{ID|_{k-1}}, PP)$**:** This algorithm takes as input an identity $ID|_k = (I_1, \ldots, I_k) \in \mathcal{I}^k$, the state $ST_{ID|_{k-1}}$, and public parameters PP. Note that the state $ST_{ID|_{k-1}}$ contains $BT_{ID|_{k-1}}$.

1. It first assigns $ID|_k$ to a random leaf node in $BT_{ID|_{k-1}}$. Let $Path$ be a path node set defined by $\mathbf{Path}(ID|_k) \in BT_{ID|_{k-1}}$.
2. For each node $\theta \in Path$, it performs the following steps: It first retrieves $\gamma_\theta \in \mathbb{Z}_p$ from $BT_{ID|_{k-1}}$ where γ_θ is associated to the node θ. Note that if γ_θ is not defined, then it chooses a random exponent $\gamma_\theta \in \mathbb{Z}_p$ and stores it to the node θ. Next, it creates a partial private key $PSK_\theta = \left(K_0, \{K_{i,1}, K_{i,2}\}_{i=1}^{k}\right)$ by running **HIBE.GenKey**$(ID|_k, \gamma_\theta, PP)$.
3. Finally, it outputs a private key $SK_{ID|_k} = \left(\{\theta, PSK_\theta\}_{\theta \in Path}\right)$.

RHIBE.UpdateKey$(T, RL_{ID|_{k-1}}, DK_{ID|_{k-1},T}, ST_{ID|_{k-1}}, PP)$**:** This algorithm takes as input time $T \in \mathcal{T}$, the revocation list $RL_{ID|_{k-1}}$, the decryption key $DK_{ID|_{k-1},T}$, the state $ST_{ID|_{k-1}}$ where it contains $BT_{ID|_{k-1}}$, and public parameters PP. Recall that $RL_{ID|_0} = RL_0$ and $ST_{ID|_0} = ST_0$. Note that $DK_{ID|_0,T} = \left(D_0 = g^\alpha (u_0^T h_0)^{-r_0}, D_1 = g^{r_0}\right)$ can be easily generated by using MK.

1. Let $KUNode$ be a covering set that is obtained by running **KUNode**$(BT_{ID|_{k-1}}, RL_{ID|_{k-1}}, T)$.
2. For each node $\theta \in KUNode$, it performs the following steps: It first retrieves $\gamma_\theta \in \mathbb{Z}_p$ from $BT_{ID|_{k-1}}$ where γ_θ is associated to the node θ. It obtains $DK'_{ID|_{k-1},T} = \left(D'_0, D'_1, \{D'_{i,1}, D'_{i,2}\}_{i=1}^{k-1}\right)$ by running **RHIBE.RandDK**$(DK_{ID|_{k-1},T}, PP)$. Next, it creates a time-constrained update key
$$TUK_\theta = \left(U_0 = g^{-\gamma_\theta} \cdot D'_0,\ U_1 = D'_1,\ \left\{U_{i,1} = D'_{i,1},\ U_{i,2} = D'_{i,2}\right\}_{i=1}^{k-1}\right).$$
3. Finally, it outputs an update key $UK_{ID|_{k-1},T} = \left(\{\theta, TUK_\theta\}_{\theta \in KUNode}\right)$.

RHIBE.DeriveKey$(SK_{ID|_k}, UK_{ID|_{k-1},T}, PP)$**:** This algorithm takes as input a private key $SK_{ID|_k}$ for an identity $ID|_k$, an update key $UK_{ID|_{k-1},T}$ for time T and the public parameters PP.

1. If $ID|_k \notin RL_{ID|_{k-1}}$, then it finds a unique node $\theta^* \in \mathbf{Path}(ID|_k) \cap \mathbf{KUNode}(BT_{ID|_{k-1}}, RL_{ID|_{k-1}}, T)$. Otherwise, it outputs $\perp$.
2. It derives $PSK_{\theta^*} = \left(K_0, \{K_{i,1}, K_{i,2}\}_{i=1}^{k}\right)$ from $SK_{ID|_k}$ and $TUK_{\theta^*} = \left(U_0, U_1, \{U_{i,1}, U_{i,2}\}_{i=1}^{k-1}\right)$ from $UK_{ID|_{k-1},T}$ for the node θ^*. Next, it creates a decryption key
$$DK_{ID|_k,T} = \Big(D_0 = K_0 \cdot U_0,\ D_1 = U_1,\ \left\{D_{i,1} = K_{i,1} \cdot U_{i,1},\ D_{i,2} = K_{i,2} \cdot U_{i,2}\right\}_{i=1}^{k-1},$$
$$\left\{D_{k,1} = K_{k,1},\ D_{k,2} = K_{k,2}\right\}\Big)$$
and re-randomizes it by running **RHIBE.RandDK**.
3. Finally, it outputs a (re-randomized) decryption key $DK_{ID|_k,T} = \left(D_0, D_1, \{D_{i,1}, D_{i,2}\}_{i=1}^{k}\right)$.

RHIBE.RandDK$(DK_{ID|_k,T}, PP)$**:** This algorithm takes as input a decryption key $DK_{ID|_k} = \left(D'_0, D'_1, \{D'_{i,1}, D'_{i,2}\}_{i=1}^{k}\right)$ for an identity $ID|_k = (I_1, I_2, \ldots, I_k) \in \mathcal{I}^k$ and time T, and the public parameters PP. It selects random exponents $r_0, r_1, \ldots, r_k \in \mathbb{Z}_p$ and outputs a re-randomized decryption key
$$DK_{ID|_k,T} = \left(D_0 = D'_0 \cdot (u_0^T h_0)^{-r_0} \cdot \prod_{i=1}^{k} w^{r_i},\ D_1 = D'_1 \cdot g^{r_0},\ \left\{D_{i,1} = D'_{i,1} \cdot (u^{I_i} h)^{-r_i},\ D_{i,2} = D'_{i,2} \cdot g^{r_i}\right\}_{i=1}^{k}\right).$$

RHIBE.Encrypt($ID|_\ell, T, M, PP$)**:** This algorithm takes as input an identity $ID|_\ell = (I_1, \ldots, I_\ell) \in \mathcal{I}^k$, time T, a message M, and the public parameters PP. It first chooses random exponents $t, s_1, \ldots, s_\ell \in \mathbb{Z}_p$ and outputs a ciphertext

$$CT_{ID|_k,T} = \Big(C = \Omega^t \cdot M,\ C_0 = g^t,\ C_1 = (u_0^T h_0)^t,\ \big\{ C_{i,1} = g^{s_i},\ C_{i,2} = w^{-t}(u^{I_i} h)^{s_i} \big\}_{i=1}^{\ell} \Big).$$

RHIBE.Decrypt($CT_{ID|_\ell,T}, DK_{ID'|_k,T'}, PP$)**:** This algorithm takes as input a ciphertext $CT_{ID|_\ell,T} = (C, C_0, C_1, \{C_{i,1}, C_{i,2}\}_{i=1}^{\ell})$, a decryption key $DK_{ID'|_k,T'} = (D_0, D_1, \{D_{i,1}, D_{i,2}\}_{i=1}^{k})$ and the public parameters PP. If $ID'|_k$ is a prefix of $ID|_\ell$ and $T = T'$, then it outputs an encrypted message

$$M = C \cdot \Big(e(C_0, D_0) \cdot e(C_1, D_1) \cdot \prod_{i=1}^{k} \big(e(C_{i,1}, D_{i,1}) \cdot e(C_{i,2}, D_{i,2}) \big) \Big)^{-1}$$

Otherwise, it outputs $\perp$.

RHIBE.Revoke($ID|_k, T, RL_{ID|_{k-1}}, ST_{ID|_{k-1}}$)**:** This algorithm takes as input an identity $ID|_k$, revocation time T, the revocation list $RL_{ID|_{k-1}}$, and the state $ST_{ID|_{k-1}}$. If $(ID|_k, -) \notin ST_{ID|_{k-1}}$, then it outputs $\perp$ since the private key of $ID|_k$ was not generated. Otherwise, it adds $(ID|_k, T)$ to $RL_{ID|_{k-1}}$ and outputs the updated revocation list $RL_{ID|_{k-1}}$.

4.3 Correctness

We have to check the correctness of the scheme. If a user's private key is not revoked in an update key, then the algorithm **RHIBE.DeriveKey** correctly derives a decryption key $DK_{ID|_k,T}$ since there exists a unique common node θ in path nodes $Path$ in a private key and a covering nodes $KUNode$ in an update key. Since the decryption key $DK_{ID|_k,T}$ for an identity $ID|_k$ at time T has the following form

$$DK_{ID|_k,T} = \Big(D_0 = g^{\alpha} (u_0{}^T h_0)^{-r_0} \prod_{i=1}^{k} w^{r_i},\ D_1 = g^{r_0},\ \big\{ D_{i,1} = (u^{I_i} h)^{-r_i},\ D_{i,2} = g^{r_i} \big\}_{i=1}^{k} \Big),$$

we can easily check that the ciphertext $CT_{ID|_k,T}$ can be correctly decrypted by using $DK_{ID|_k,T}$.

4.4 Security Analysis

Theorem 4.2. *The above RHIBE scheme is selectively IND-CPA secure if the underlying HIBE scheme is selectively IND-CPA secure.*

Proof. To prove the theorem, we will show that a polynomial time algorithm $\mathcal{B}$ that breaks selective IND-CPA security of HIBE in Section 3 can be built by using an adversary $\mathcal{A}$ that breaks the selective IND-CPA security of the proposed RHIBE scheme. $\mathcal{B}$ that interacts with $\mathcal{A}$ is described as follows:

Init: $\mathcal{A}$ initially submits a challenge identity $ID^*|_\ell = (I_1^*, \ldots, I_\ell^*)$ and challenge time T^*. $\mathcal{B}$ also submits $ID^*|_\ell$ as a challenge identity for the IND-CPA experiment of HIBE.

Setup: $\mathcal{B}$ receives $PP_{HIBE} = ((p, \mathbb{G}, \mathbb{G}_T, e), g, u, h, w, v, \Omega)$. It chooses random exponents $a, b \in \mathbb{Z}_p$ and sets $u_0 = w^b, h_0 = g^a u_0^{-T^*}$. It sets $PP_{RHIBE} = \big(PP_{HIBE}, u_0, h_0\big)$ and gives PP_{RHIBE} to $\mathcal{A}$.

For update key and private key queries, $\mathcal{B}$ classifies the type of adversaries and responds depending on his guess of the adversarial type. $\mathcal{B}$ guesses $i^* \in \{1, \ldots, \ell, \ell+1\}$. In case of $i^* = \ell + 1$, $\mathcal{B}$ supposes $\mathcal{A}$ has

no private key queries for $ID^*|_j$ for all $j \in [\ell]$. Otherwise, in case of $i^* \in [\ell]$, $\mathcal{B}$ assumes that $\mathcal{A}$ queries a private key of some $ID^*|_j$ for $i^* \leq j \leq \ell$, it means that $ID^*|_{i^*}$ is the oldest ancestor of $ID^*|_\ell$ which is queried before time T^*. If $\mathcal{B}$ discovers the guess is wrong, then $\mathcal{B}$ aborts the simulation and outputs a random bit. If $\mathcal{B}$ guesses correctly, then we can believe that $\mathcal{A}$ cannot get any information about $ST_0, ..., ST_{ID^*|_{i^*-1}}$.

Phase 1: $\mathcal{A}$ adaptively requests a polynomial number of queries. These queries are processed as follows:

If this is a private key query for $ID|_k = (I_1, \ldots, I_k)$, then $\mathcal{B}$ proceeds as follows: $\mathcal{B}$ has to output $ST_{ID|_k}$, then $\mathcal{B}$ generates $ST_{ID|_k}$ properly. Also, the private key of $ID|_k$ is related to $ST_{ID|_{k-1}}$.

- **Case** $ID|_{k-1}$ is not a prefix of $ID^*|_{i^*}$: In this case, the state information $ST_{ID|_{k-1}}$ is normally generated. It easily creates $SK_{ID|_k}$ by running **RHIBE.GenKey**$(ID|_k, ST_{ID|_{k-1}}, PP)$ since it knows $ST_{ID|_{k-1}}$. It also normally generates $ST_{ID|_k}$. Note that it sets γ_θ as the master key of PSK where γ_θ is associated to a node θ. Finally, it gives $SK_{ID|_k}$ and $ST_{ID|_k}$ to $\mathcal{A}$.

- **Case** $ID|_{k-1}$ is a prefix of $ID^*|_{i^*}$: In this case, $\mathcal{A}$ does not know $ST_{ID^*|_{k-1}}$, so it does not need to generate $ST_{ID^*|_{k-1}}$ properly. If $BT_{ID^*|_{k-1}}$ is already generated, then it use this. Otherwise, it has to generate $BT_{ID^*|_{k-1}}$. Next, it assigns a random leaf node of $BT_{ID^*|_{k-1}}$ to $ID^*|_k$. Let $Path = \mathbf{Path}(ID|_k) \subset BT_{ID^*|_{k-1}}$. Note that it can retrieve $\gamma_\theta \in \mathbb{Z}_p$ by loading from $BT_{ID|_{k-1}}$ if it is already selected or selecting $\gamma_\theta \in \mathbb{Z}_p$ otherwise. It generates $SK_{ID|_k}$ as following:

  - Case $k < i^*$: In this case, we have $ID|_k \neq ID^*|_k$ since $\mathcal{A}$ is a type i^* adversary. It first request an HIBE private key query for $ID|_k$ and receives $SK_{HIBE} = \left(K'_0, \{K'_{i,1}, K'_{i,2}\}_{i=1}^k\right)$. For each $\theta \in Path$, it retrieves γ_θ and creates PSK_θ by running **HIBE.RandKey**$(ID|_k, \gamma_\theta, SK_{HIBE}, PP)$. It creates $SK_{ID|_k} = \left(\{\theta, PSK_\theta\}\right)_{\theta \in Path}$. Note that it sets $\alpha + \gamma_\theta$ as the master key of PSK.
  - Case $k = i^*$ and $ID|_k \neq ID^*|_{i^*}$: It first requests an HIBE private key for $ID|_k$ and receives $SK_{HIBE} = \left(K'_0, \{K'_{i,1}, K'_{i,2}\}_{i=1}^k\right)$. For each $\theta \in Path$, it retrieves γ_θ and proceeds as follows: If $\theta \in \mathbf{Path}(ID^*|_k)$, it creates PSK_θ by running **HIBE.GenKey**$(ID|_k, \gamma_\theta, PP)$. Otherwise, it creates PSK_θ by running **HIBE.RandKey**$(ID|_k, \gamma_\theta, SK_{HIBE}, PP)$. It creates $SK_{ID|_k} = (\{\theta, PSK_\theta\}_{\theta \in Path})$. Note that it sets γ_θ as the master key of PSK if $\theta \in \mathbf{Path}(ID^*|_{i^*})$ or $\alpha + \gamma_\theta$ as the master key otherwise.
  - Case $k = i^*$ and $ID|_k = ID^*|_{i^*}$: For each $\theta \in Path$, it retrieves γ_θ and creates PSK_θ by running **HIBE.GenKey**$(ID|_k, \gamma_\theta, PP)$. It creates $SK_{ID|_k} = \left(\{\theta, PSK_\theta\}_{\theta \in Path}\right)$. Note that it simply sets γ_θ as the master key of PSK.

  Finally, it gives $SK_{ID|_k}$ and $ST_{ID|_k}$ to $\mathcal{A}$.

If this is an update key query for $ID|_{k-1} = (I_1, \ldots, I_{k-1})$ and T, then $\mathcal{B}$ proceeds as follows:

- **Case** $ID|_{k-1}$ is not a prefix of $ID^*|_{i^*}$: In this case, the decryption key $DK_{ID|_{k-1},T}$ can be obtained and the state information $ST_{ID|_{k-1}}$ is normally generated. It first obtains $DK_{ID|_{k-1},T}$ by requesting a decryption key query for $ID|_{k-1}$ and T to himself. Next, it retrieves $ST_{ID|_{k-1}}$ if it is already generated or normally generates $ST_{ID|_{k-1}}$ otherwise. It creates $UK_{ID|_{k-1},T}$ by running **RHIBE.UpdateKey**$(T, RL_{ID|_{k-1}}, DK_{ID_{k-1},T}, ST_{ID|_{k-1}}, PP)$. Note that it sets $\alpha - \gamma_\theta$ as the master key of TUK where γ_θ is associated to a node θ. Finally, it gives $UK_{ID|_{k-1},T}$ to $\mathcal{A}$.

- **Case** $ID|_{k-1}$ is a prefix of $ID^*|_{i^*}$: Let $KUNode = \mathbf{KUNode}(BT_{ID|_{k-1}}, RL_{ID|_{k-1}}, T)$. In this case, it generates $UK_{ID|_{k-1},T}$ as follows:

– Case $ID|_{k-1}$ is a prefix of $ID^*|_{i^*-1}$: For each $\theta \in KUNode$, it retrieves γ_θ and proceeds as follows: It first obtains $SK_{HIBE} = \left(K'_0, \{K'_{i,1}, K'_{i,2}\}_{i=1}^{k-1}\right)$ by running **HIBE.GenKey**$(ID|_{k-1}, -\gamma_\theta, PP)$. Next, it creates TUK_θ by selecting a random exponent $r_0 \in \mathbb{Z}_p$ as $TUK_\theta = \left(U_0 = K'_0 \cdot (u_0^T h_0)^{-r_0},\ U_1 = g^{r_0},\ \{U_{i,1} = K'_{i,1},\ U_{i,2} = K'_{i,2}\}_{i=1}^{k-1}\right)$. It creates $UK_{ID|_{k-1},T} = \left(\{\theta, TUK_\theta\}_{\theta \in KUNode}\right)$. Note that it sets $-\gamma_\theta$ as the master key of TUK.

– Case $ID|_{k-1} = ID^*|_{i^*-1}$: In this case, we have $T \neq T^*$ by the restriction that $ID^*|_k$ should be revoked before the time T^*.

1. If $k = 1$, it creates $DK_{ID|_0,T} = \left(D_0 = v^{-\frac{a}{b(T-T^*)}} \cdot g^{-ar_0} \cdot u_0^{-r_0(T-T^*)}, D_1 = v^{\frac{1}{b(T-T^*)}} \cdot g^{ar_0}\right)$. Otherwise, it creates $DK_{ID|_{k-1},T} = \left(D_0, D_1, \{D_{i,1}, D_{i,2}\}_{i=1}^{k-1}\right)$ by requesting a decryption key query to himself.
2. For each $\theta \in KUNode$, it retrieves γ_θ and proceeds as follows: If $\theta \in$ **Path**$(ID^*|_k)$, it obtains a re-randomized decryption key $DK'_{ID|_{k-1},T} = \left(D'_0, D'_1, \{D'_{i,1}, D'_{i,2}\}_{i=1}^{k-1}\right)$ by running **RHIBE.RandDK**$(DK_{ID|_{k-1},T}, PP)$ and creates $TUK_\theta = (U_0 = g^{-\gamma_\theta} \cdot D'_0,\ U_1 = D'_1,\ \{U_{i,1} = D'_{i,1},\ U_{i,2} = D'_{i,2}\}_{i=1}^{k-1})$. Otherwise, it obtains $SK_{HIBE} = \left(K'_0, \{K'_{i,1}, K'_{i,2}\}_{i=1}^{k-1}\right)$ by running **HIBE.GenKey**$(ID|_{k-1}, -\gamma_\theta, PP_{HIBE})$ and creates $TUK_\theta = \left(U_0 = K'_0 \cdot (u_0^T h_0)^{-r_0},\ U_1 = g^{r_0},\ \{U_{i,1} = K'_{i,1},\ U_{i,2} = K'_{i,2}\}_{i=1}^{k-1}\right)$.
3. It creates $UK_{ID|_{k-1},T} = \left(\{\theta, TUK_\theta\}_{\theta \in KUNode}\right)$. Note that it sets $\alpha - \gamma_\theta$ as the master key of TUK if $\theta \in$ **Path**$(ID^*|_k)$ or $-\gamma_\theta$ as the master key otherwise.

Finally, it gives $UK_{ID|_{k-1},T}$ to $\mathcal{A}$.

If this is a decryption key query for $ID|_k = (I_1, \ldots, I_k)$ and T, then $\mathcal{B}$ proceeds as follows:

- **Case** $T \neq T^*$: In this case, it can easily generate $DK_{ID|_0,T}$ since $T \neq T^*$ and then delegates it to generate $DK_{ID|_k,T}$. It first selects a random exponent $r_0 \in \mathbb{Z}_p$ and creates $DK_{ID|_0,T} = \left(D'_0 = v^{-\frac{a}{b(T-T^*)}} g^{-ar_0} u_0^{-r_0(T-T^*)}, D'_1 = v^{\frac{1}{b(T-T^*)}} g^{r_0}\right)$. Next, it selects random exponents $r_1, \ldots, r_k \in \mathbb{Z}_p$ and creates $DK_{ID|_k,T} = \left(D_0 = D'_0 \cdot \prod_{i=1}^{k} w^{r_i},\ D_1 = D'_1,\ \{D_{i,1} = (u^{I_i} h)^{-r_i},\ D_{i,2} = g^{r_i}\}_{i=1}^{k}\right)$.
- **Case** $T = T^*$: In this case, there exists at least one $I_j \in ID|_k$ such that $I_j \notin ID^*|_k$ by the restriction of the decryption key query. It first queries the private key query for $ID|_k$ since $I_j \notin ID^*|_k$ and receives $SK_{HIBE} = \left(K'_0, \{K'_{i,1}, K'_{i,2}\}_{i=1}^{k}\right)$. Next, it selects a random exponent $r_0 \in \mathbb{Z}_p$ and creates $DK_{ID|_k,T} = \left(D_0 = K'_0 \cdot g^{-ar_0}, D_1 = g^{r_0}, \left\{D_{i,2} = K'_{i,1}, D_{i,2} = K'_{i,1}\right\}_{i=1}^{k}\right)$.

If this is a revocation query for $ID|_k = (I_1, \ldots, I_k)$ and T, then $\mathcal{B}$ proceeds as follows: It adds a pair $(ID|_k, T)$ into a revocation list $RL_{ID|_{k-1}}$ by running **RHIBE.Revoke** algorithm. Note that $\mathcal{A}$ cannot query to revoke $ID|_k$ on time T if he already requested an update key query for $ID|_k$ on time T.

Challenge: In the challenge step, $\mathcal{A}$ submits two challenge messages M_0^*, M_1^* with the same length. $\mathcal{B}$ also submits M_0^*, M_1^* as challenge messages and receives $CT_{HIBE} = \left(C', C'_0, \{C'_{i,1}, C'_{i,2}\}_{i=1}^{\ell}\right)$. Next, $\mathcal{B}$ sets the challenge ciphertext $CT_{ID^*|_\ell,T^*} = \left(C = C',\ C_0 = C'_0,\ C_1 = (C'_0)^a,\ \{C_{i,1} = C'_{i,1},\ C_{i,2} = C'_{i,2}\}_{i=1}^{\ell}\right)$ and gives it to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ may continue to request a polynomial number of queries as the same as **Phase 1**.

Guess: Finally, $\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$. $\mathcal{B}$ also outputs the same guess μ'. □

5 Conclusion

In this paper, we proposed the first unbounded RHIBE scheme using proposed HIBE and the history-free approach of Seo and Emura [24]. To achieve our scheme, we first proposed an unbounded HIBE scheme from the KP-ABE scheme of Rouselakis and Waters [19]. Our proposed RHIBE scheme makes it efficient to generate private keys in IBE for a large number of users since it allows the delegation of the key generation using a hierarchical structure among users and provides the revocation functionality. Furthermore it solves the open problem of removing the limitation on maximum hierarchy.

The security of our RHIBE scheme was proved in the selective model. It will be interesting to construct a fully secure RHIBE scheme with no limitations on maximum hierarchy. Our RHIBE scheme essentially uses the complete subtree (CS) method for revocation. We expect that the subset difference (SD) method also can be applied to our RHIBE scheme since Seo and Emura [24] also proposed an RHIBE scheme that uses the SD method by following the methodology of Lee et al. [12].

Acknowledgements

The first two authors (Geumsook Ryu and Kwangsu Lee) contributed equally to this work.

References

[1] Alexandra Boldyreva, Vipul Goyal, and Virendra Kumar. Identity-based encryption with efficient revocation. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM Conference on Computer and Communications Security*, pages 417–426. ACM, 2008.

[2] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.

[3] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005.

[4] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[5] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307. Springer, 2006.

[6] Craig Gentry. Practical identity-based encryption without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 445–464. Springer, 2006.

[7] Craig Gentry and Shai Halevi. Hierarchical identity based encryption with polynomially many levels. In Omer Reingold, editor, *Theory of Cryptography - TCC 2009*, volume 5444 of *Lecture Notes in Computer Science*, pages 437–456. Springer, 2009.

[8] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, 2002.

[9] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer, 2002.

[10] Kwangsu Lee. Self-updatable encryption with short public parameters and its extensions. *Designs Codes Cryptogr.*, 2015. `http://dx.doi.org/10.1007/s10623-015-0039-9`.

[11] Kwangsu Lee, Seung Geol Choi, Dong Hoon Lee, Jong Hwan Park, and Moti Yung. Self-updatable encryption: Time constrained access control with hidden attributes and better efficiency. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*, volume 8269 of *Lecture Notes in Computer Science*, pages 235–254. Springer, 2013.

[12] Kwangsu Lee, Dong Hoon Lee, and Jong Hwan Park. Efficient revocable identity-based encryption via subset difference methods. Cryptology ePrint Archive, Report 2014/132, 2014. `http://eprint.iacr.org/2014/132`.

[13] Kwangsu Lee, Jong Hwan Park, and Dong Hoon Lee. Anonymous hibe with short ciphertexts: full security in prime order groups. *Designs Codes Cryptogr.*, 74(2):395–425, 2015.

[14] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In Daniele Micciancio, editor, *Theory of Cryptography - TCC 2010*, volume 5978 of *Lecture Notes in Computer Science*, pages 455–479. Springer, 2010.

[15] Allison B. Lewko and Brent Waters. Unbounded hibe and attribute-based encryption. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 547–567. Springer, 2011.

[16] Benoît Libert and Damien Vergnaud. Adaptive-id secure revocable identity-based encryption. In Marc Fischlin, editor, *Topics in Cryptology - CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2009.

[17] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer, 2001.

[18] Seunghwan Park, Kwangsu Lee, and Dong Hoon Lee. New constructions of revocable identity-based encryption from multilinear maps. *IEEE Trans. Inf. Forensic Secur.*, 10(8):1564–1577, 2015.

[19] Yannis Rouselakis and Brent Waters. Practical constructions and new proof methods for large universe attribute-based encryption. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM Conference on Computer and Communications Security*, pages 463–474. ACM, 2013.

[20] Amit Sahai, Hakan Seyalioglu, and Brent Waters. Dynamic credentials and ciphertext delegation for attribute-based encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 199–217. Springer, 2012.

[21] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2005.

[22] Jae Hong Seo and Keita Emura. Efficient delegation of key generation and revocation functionalities in identity-based encryption. In Ed Dawson, editor, *Topics in Cryptology - CT-RSA 2013*, volume 7779 of *Lecture Notes in Computer Science*, pages 343–358. Springer, 2013.

[23] Jae Hong Seo and Keita Emura. Revocable identity-based encryption revisited: Security model and construction. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography - PKC 2013*, volume 7778 of *Lecture Notes in Computer Science*, pages 216–234. Springer, 2013.

[24] Jae Hong Seo and Keita Emura. Revocable hierarchical identity-based encryption: History-free update, security against insiders, and short ciphertexts. In Kaisa Nyberg, editor, *Topics in Cryptology - CT-RSA 2015*, volume 9048 of *Lecture Notes in Computer Science*, pages 106–123. Springer, 2015.

[25] Jae Hong Seo, Tetsutaro Kobayashi, Miyako Ohkubo, and Koutarou Suzuki. Anonymous hierarchical identity-based encryption with constant size ciphertexts. In Stanislaw Jarecki and Gene Tsudik, editors, *Public-Key Cryptography - PKC 2009*, volume 5443 of *Lecture Notes in Computer Science*, pages 215–234. Springer, 2009.

[26] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology - CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

[27] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005.

[28] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer, 2009.