Efficient Delegation of Key Generation and Revocation Functionalities in Identity-Based Encryption*

Jae Hong Seo† and Keita Emura†

January 11, 2013

Abstract

In the public key cryptosystems, revocation functionality is required when a secret key is corrupted by hacking or the period of a contract expires. In the public key infrastructure setting, numerous solutions have been proposed, and in the Identity Based Encryption (IBE) setting, a recent series of papers proposed revocable IBE schemes. Delegation of key generation is also an important functionality in cryptography from a practical standpoint since it allows reduction of excessive workload for a single key generation authority. Although efficient solutions for either revocation or delegation of key generation in IBE systems have been proposed, an important open problem is efficiently delegating both the key generation and revocation functionalities in IBE systems. Libert and Vergnaud, for instance, left this as an open problem in their CT-RSA 2009 paper. In this paper, we propose the first solution for this problem. We prove the selective-ID security of our proposal under the Decisional Bilinear Diffie-Hellman assumption in the standard model.

1 Introduction

Shamir introduced the concept of the Identity-based Encryption (IBE) scheme, a public key encryption scheme allowing any bit-string (e.g., e-mail address) to be a public key of a user that chooses such a bit-string [24]. Since Boneh and Franklin's first realization of IBE using bilinear pairings over elliptic curves, IBE systems have been applied in numerous applications. Several variations of IBE systems have also been proposed for adding other functionalities. In particular, the hierarchical identity-based encryption (HIBE) scheme allows the key generation center (KGC) to delegate the key generation functionality to users [11] and the revocable IBE (RIBE) scheme allows the KGC to efficiently revoke users for each time period [2].

Revocation Functionality in IBE. In public key cryptosystems, we need revocation functionality when a secret key is corrupted by hacking or the period of a contract expires. In the public key infrastructure setting, numerous solutions have been proposed, and in the IBE setting, a series of recent papers has proposed *scalable* RIBE schemes since Boldyreva et al. [2]. In fact, Boneh and Franklin [5] already proposed a trivial solution for revocation functionality, wherein new decryption keys are issued for each time period. However, their solution introduces huge overheads for the KGC that are linearly increased in the number of users. Boldyreva et al. and all subsequent works were aimed at constructing a scalable RIBE scheme, that is, the KGC's overhead increases logarithmically in the number of users. All proposed scalable RIBE schemes used the same methodology for revocation by using a binary tree structure. Each user ID is assigned to a leaf node ζ_{ID} of the binary tree structure and has keys corresponding to the nodes on the path between the assigned leaf node and the root node. By using the technique called the Complete Subtree (CS) method [20], which is widely accepted for broadcast encryption, the KGC broadcasts the key update for each time period (i.e., no secure channel is required in this phase) such that only non-revoked users can generate the decryption key

*An extended abstract appears in CT-RSA 2013 [22]. This is the full version.

†National Institute of Information and Communications Technology (NICT), 4-2-1, Nukui-kitamachi, Koganei, Tokyo, 184-8795, Japan. {jaehong, k-emura}@nict.go.jp

for that time period from their secret key and the key update. For a non-revoked user, there is at least one subkey among the $\log N$ size key, where N is the maximum number of users. Since the CS method is secure against colluding and allows short key updates, the resulting RIBE scheme is well scalable and secure.[1]

Delegation Functionality in IBE. For a large network, a single KGC has an excessive workload for performing computationally expensive key generation and establishing secure channels to transmit each user's secret key. To mitigate this problem, Horwitz and Lynn [14] introduced the concept of HIBE such that the responsibility for key generation is distributed to the lower-level KGC by delegating key generation functionality. Numerous constructions for HIBE schemes and variants with additional properties have subsequently been proposed [11, 3, 4, 6, 10, 23, 25, 16].

Delegation of Both Key Generation and Revocation Functionalities in IBE. Although IBE schemes with either efficient revocation or efficient delegation for key generation functionality have been proposed, it is non-trivial to achieve both functionalities at the same time, and in fact Libert and Vergnaud left this as an open problem at CT-RSA 2009 [18]. We simply call such a scheme having both functionalities a Revocable HIBE (RHIBE) scheme. There are some difficulties in achieving RHIBE.

1. Trivial approaches will lead to exponentially large secret keys in the corresponding hierarchical level.
2. Key generations and key updates are recursively defined: this leads some difficulty in the security proof.

All existing scalable RIBE schemes utilize binary tree structures, that is the CS method, for revocation. In the scalable RIBE scheme using the CS method, a secret key of each user consists of $\log N$ subkeys, where N is the number of all users and at least one subkey of a non-revoked user ID can be used to generate a decryption key $\mathsf{dk}_{\mathsf{ID},\mathsf{T}}$ from the key update ku_{T} on a time period T. If we extend the RIBE scheme for the RHIBE scheme in a natural way, the second-level user has to have $(\log N)^2$ subkeys since one of the subkeys of the paraent's key can be used in each time period so that a child should have a $\log N$ subkey for each parent's subkey. In general, ℓ-level users have $(\log N)^\ell$ subkeys, so the size of the secret key exponentially grows in the corresponding hierarchical depth.

For constructing RHIBE, if we follow the same strategy used by all scalable RIBE schemes, KGC may not be able to directly generate secret keys of descendants (except for the first-level user). Each intermediate-level user's secret key is generated according to the shape of the binary tree structure, which is managed by its parent. However, the KGC does not know such a binary tree, so the KGC cannot create secret keys of intermediate-level users. (Note that the KGC can generate decryption keys for all descendants.) Therefore, the secret key and key updates have to be *recursively* defined. This makes the situation more complicated. In particular, in the security model the adversary can query secret keys of descendants of the challenge identity, but it is non-trivial to recursively generate such secret keys without knowing the challenge identity's secret key. We explain the detailed difficulty in Section 4.

Our Contribution. In this paper, we study a way to *efficiently* delegate both key generation and revocation functionalities in the IBE system. In particular, we propose the first realization of RHIBE. Our RHIBE construction is based on the Boneh-Boyen HIBE (BB-HIBE) scheme [3]. We carefully deal with the difficulties mentioned above. The ciphertext size of our RHIBE is almost the same as that of the BB-HIBE (only one additional group element is required) and the revocation cost of each user is the same as that of the Boldyreva et al. one. Nevertheless, our scheme enables hierarchical structures of identities. Moreover, a user (of a level ℓ) needs to manage $O(\ell^2 \log N)$-size secret key (polynomial in the hierarchical depth). Our RHIBE is selective-ID secure under the Decisional Bilinear Diffie-Hellman (DBDH) assumption in the standard model.

Related Work. Boldyreva et al. [2] proposed the first RIBE by applying the fuzzy IBE scheme [21]. Thanks to the collusion resistance of the underlying fuzzy IBE, no revoked users can compute a decryption key from their own secret key and publicly available key update information. Moreover, by applying the CS

[1] The security model for the RIBE scheme is almost equal to that of the conventional IBE scheme. The only difference is that the adversary of the RIBE scheme is allowed to query for the challenge identity ID^*, but in this case the challenge identity should be revoked on the challenge time T^*.

method [20], the Boldyreva et al. scheme is scalable in the sense that the costs of the KGC logarithmically depend on the number of users. Although their scheme is secure under a relatively weaker notion, called selective-ID security, Libert and Vergnaud [18] proposed adaptive-ID secure RIBE by applying a variant of the Waters IBE scheme [19]. In the proof of the Libert-Vergnaud RIBE scheme, the simulator can construct all the secret keys (as in the Gentry IBE [9]) to answer the secret key query for the challenge user. Recently, a RIBE scheme from lattices [7] was proposed.

In R(H)IBE, each decryption key is computed from the long-term secret key and key update information. Dodis et al. considered a similar functionality which we call key-insulated PKE [8, 1, 17] and IBE [13, 12, 26]. A user computes its decryption key from the long-term secret key and a helper key served in physically insulated storage. A difference between key-insulated PKE and RIBE is that the former requires a secure channel between a user and the storage for every key update, but the latter needs a secure channel only once for transmitting each user's secret key from its ancestor. Moreover, since each helper key is generated for each user, the total size of the helper key linearly depends on the number of total users, whereas it logarithmically depends on the number of total (i.e., non-revoked) users in RIBE.

Outline. This paper is organized as follows. The next section gives preliminaries. In Section 3, we define a syntax and a security model for the RHIBE scheme and we propose our RHIBE construction in Section 4. Lastly, we analyze the security of the proposed scheme and give a conclusion with interesting open problems.

2 Preliminaries

This section gives the definition of the bilinear groups, the DBDH assumption, the KUNode algorithm [2], and the definition of HIBE, and introduce the BB-HIBE scheme.

Definition 2.1 (Bilinear Groups)**.** *The bilinear group generator $\mathcal{G}(\cdot)$ is an algorithm that 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 size 2λ, $\mathbb{G}$ and $\mathbb{G}_t$ are cyclic groups of order p, and e is an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with*

- *Bilinearity : for all $u, u', v, v' \in \mathbb{G}$, $e(uu', v) = e(u, v)e(u', v)$ and $e(u, vv') = e(u, v)e(u, v')$,*
- *Non-degeneracy : for a generator g of $\mathbb{G}$, $e(g, g) \neq 1_{\mathbb{G}_t}$, where $1_{\mathbb{G}_t}$ is identity element in $\mathbb{G}_t$.*

Definition 2.2 (Decision Bilinear Diffie-Hellman (DBDH) Assumption)**.** *Given a bilinear group $(p, \mathbb{G}, \mathbb{G}_t, e)$ generated by $\mathcal{G}(\lambda)$, define two distributions $\mathcal{D}_0(\lambda) = (g, g^a, g^b, g^c, e(g, g)^{abc})$ and $\mathcal{D}_1(\lambda) = (g, g^a, g^b, g^c, e(g, g)^z)$, where $g \xleftarrow{\$} \mathbb{G}$ and $a, b, c, z \xleftarrow{\$} \mathbb{Z}_p$. The DBDH problem in the bilinear group $(p, \mathbb{G}, \mathbb{G}_t, e)$ is to decide a bit b from given $\mathcal{D}_b$, where $b \xleftarrow{\$} \{0, 1\}$. The advantage of $\mathcal{A}$ in solving the DBDH problem in the bilinear group $(p, \mathbb{G}, \mathbb{G}_t, e)$ is defined by*

$$Adv^{DBDH}_{\mathcal{G},\mathcal{A}}(\lambda) = \Big|\Pr[\mathcal{A}(\mathcal{D}_0(\lambda)) \to 1] - \Pr[\mathcal{A}(\mathcal{D}_1(\lambda)) \to 1]\Big|.$$

We say that the DBDH assumption holds in the bilinear group $(p, \mathbb{G}, \mathbb{G}_t, e)$ if no Probabilistic Polynomial Time (PPT) algorithm has a non-negligible advantage in solving the DBDH problem in the bilinear group $(p, \mathbb{G}, \mathbb{G}_t, e)$.

Our RHIBE scheme is based on the BB-HIBE scheme [3], which is IND-sID-CPA secure under the DBDH assumption. The revocation method in each level of our construction follows Boldyreva et al.'s way using binary tree structure. The following KUNode algorithm is essentially used for revoking users in our construction.

Definition 2.3 (The KUNode Algorithm [2])**.** *This algorithm takes as input a binary tree* BT*, revocation list RL, and time T, and outputs a set of nodes. A formal description of this algorithm is as follows: If x is a non-leaf node, then x_{left} and x_{right} denote the left and right child of x, respectively. Each user is assigned*

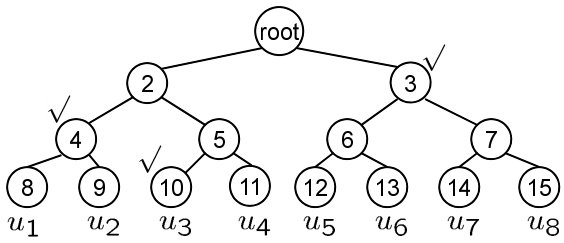

Figure 1: Example of $\mathsf{KUNode}(\mathtt{BT}, RL, T)$: We revoke the user u_4 (that is assigned to x_{11}). Then, all users, except u_4, have a node $x \in Y = \{x_3, x_4, x_{10}\}$ (we checked these nodes) that is contained in the set of nodes on the path from each user's assigned node to `root`.

to a leaf node. If a user (that is assigned to ζ) is revoked on time T, then $(\zeta, T) \in RL$. $\mathsf{Path}(\zeta)$ *denotes the set of nodes on the path from ζ to* `root`*. The description of* KUNode *is given below.*

$\mathsf{KUNode}(\mathtt{BT}, RL, T)$:
$\mathtt{X}, \mathtt{Y} \leftarrow \emptyset$;
$\forall(\zeta, T_i) \in RL$
If $T_i \leq T$ then add $\mathsf{Path}(\zeta_i)$ *to* $\mathtt{X}$
$\forall x \in \mathtt{X}$
If $x_{left} \notin \mathtt{X}$ then add x_{left} to $\mathtt{Y}$
If $x_{right} \notin \mathtt{X}$ then add x_{right} to $\mathtt{Y}$
If $\mathtt{Y} = \emptyset$ *then add* `root` *to* $\mathtt{Y}$
Return $\mathtt{Y}$

We give a simple example of KUNode in the fugure 1.

Next, we define HIBE and introduce the BB-HIBE scheme as follows.

Definition 2.4. *A hierarchical identity-based encryption (HIBE) consists of four algorithms* Setup, KeyGen[2], Enc, *and* Dec. *The specification of each algorithm is as follows.*

$\mathsf{Setup}(\lambda, N, L)$*: It takes a security parameter λ, the maximum number of users N, and the maximum length of the hierarchy of identity L as input. Then, it outputs a master public key* mpk *and a master secret key* msk*. We assume that the message space $\mathcal{M}$ and the identity space $\mathcal{I}$ (which is an union of all length of identity set), the time space $\mathcal{T}$, and the ciphertext space $\mathcal{CT}$ are contained in* mpk*.*

$\mathsf{KeyGen}(\mathsf{sk}_{\mathsf{ID}_{|\ell}}, \mathsf{ID}_{|\ell+1}, \mathsf{mpk})$*: On input of a private key $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$, a children's identity $\mathsf{ID}_{|\ell+1}$ and the master public key* mpk*, it outputs a private key $\mathsf{sk}_{\mathsf{ID}_{|\ell+1}}$.*

$\mathsf{Enc}(\mathsf{ID}_{|\ell}, M, \mathsf{mpk})$*: This algorithm takes an identity $\mathsf{ID}_{|\ell} \in \mathcal{I}$, a message $M \in \mathcal{M}$, and the master public key* mpk *as input, outputs a ciphertexts $\mathsf{CT} \in \mathcal{CT}$.*

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}_{|\ell}}, \mathsf{CT}, \mathsf{mpk})$*: Given a private key $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$, a ciphertext $\mathsf{CT} \in \mathcal{CT}$, and the master public key* mpk*, it outputs a message $M \in \mathcal{M}$ or $\perp$ (invalid ciphertext).*

[2]We equate the key derivation algorithm and the KeyGen algorithm in this paper.

Definition 2.5 (IND-sID-CPA). *Let $\mathcal{HIBE}=($*Setup, KeyGen, Enc, Dec*$)$ be a HIBE scheme. The* $\mathsf{KeyGen}(\cdot)$ *oracle is defined as follows.*

$\mathsf{KeyGen}(\cdot)$ *is the private key generation oracle. It takes an identity* $\mathsf{ID}_{|\ell}$ *of length* ℓ *as input, runs the* KeyGen *algorithm to get the private key* $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$ *and returns* $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$.

Next, we define the security of HIBE, called the IND-sID-CPA security.

$$\boxed{\mathbf{Exp}^{IND\text{-}sID\text{-}CPA}_{\mathcal{HIBE},\mathcal{A}}(\lambda)}$$
$$(\mathsf{ID}^*_{|\ell^*}) \leftarrow \mathcal{A}(state)$$
$$(\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{Setup}(\lambda, N, L)$$
$$(\mathsf{M}^*_0, \mathsf{M}^*_1, state) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\cdot)}(state, \mathsf{mpk})$$
$$b \xleftarrow{\$} \{0,1\}$$
$$\mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{ID}^*_{|\ell^*}, \mathsf{M}^*_b, \mathsf{mpk})$$
$$b' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\cdot)}(state, \mathsf{mpk}, \mathsf{CT}^*)$$
$$\textit{Return} \begin{cases} 1 \textit{ if } b = b' \\ 0 \textit{ otherwise} \end{cases}.$$

There are conditions that $\mathcal{A}$ should follow.

1. *The challenge messages* M^*_0 *and* M^*_1 *should have the same length.*
2. *No* $\mathsf{KeyGen}(\cdot)$ *is queried on the challenge identity* $\mathsf{ID}^*_{|\ell^*}$ *or its ancestors.*

The advantage of the adversary $\mathcal{A}$ is defined as

$$Adv^{IND\text{-}sID\text{-}CPA}_{\mathcal{HIBE},\mathcal{A}}(\lambda) = \left| \Pr[\mathbf{Exp}^{IND\text{-}sID\text{-}CPA}_{\mathcal{HIBE},\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right|.$$

If the function $Adv^{IND\text{-}sID\text{-}CPA}_{\mathcal{HIBE},\mathcal{A}}$ is negligible in the security parameter λ, the we say that the scheme $\mathcal{HIBE}$ is IND-sID-CPA secure.

The BB-HIBE scheme is described as follows.

$\mathsf{Setup}(\lambda, N, L)$: WLOG, we assume that $N = 2^n$ for some n. Randomly choose $g, g_2, h_1, \ldots, h_L \xleftarrow{\$} \mathbb{G}$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p$. Set $\mathsf{mpk} = \{g, g_1 = g^\alpha, g_2, h_1 \ldots, h_L\}$, $\mathsf{msk} = \{g_2^\alpha\}$.

$\mathsf{KeyGen}(\mathsf{sk}_{\mathsf{ID}_{|\ell}}, \mathsf{ID}_{|\ell+1}, \mathsf{mpk})$: Let $F_\ell(x) := g_1^x h_\ell$ for $\ell \in [1, L]$.

- $\ell = 0$: Note that $\mathsf{sk}_{\mathsf{ID}_{|1}} = \mathsf{msk}$. Choose a random value $r_1 \xleftarrow{\$} \mathbb{Z}_p$, and return $\mathsf{sk}_{\mathsf{ID}_{|1}} = (\mathsf{msk} \cdot F_1(\mathsf{I}_1)^{r_1}, g^{r_1})$, where $\mathsf{ID}_{|1} = \mathsf{I}_1$.
- $\ell > 0$: Parse $\mathsf{sk}_{\mathsf{ID}_{|\ell}} = (d', d_1, \ldots, d_\ell)$. Choose a random value $r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_p$, and return $\mathsf{sk}_{\mathsf{ID}_{|\ell+1}} = (d' \cdot F_{\ell+1}(\mathsf{I}_{\ell+1})^{r_{\ell+1}}, d_1, \ldots, d_\ell, g^{r_{\ell+1}})$, where $\mathsf{ID}_{|\ell+1} = (\mathsf{I}_1, \ldots, \mathsf{I}_{\ell+1})$.

$\mathsf{Enc}(\mathsf{ID}_{|\ell}, \mathsf{M}, \mathsf{mpk})$: Choose a random value $t \xleftarrow{\$} \mathbb{Z}_p$ and return

$$\mathsf{CT} = (\mathsf{M} \cdot e(g_1, g_2)^t, \quad g^t, \quad F_1(\mathsf{I}_1)^t, \ldots, F_\ell(\mathsf{I}_\ell)^t).$$

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}_{|\ell}}, \mathsf{CT}, \mathsf{mpk})$: Parse $\mathsf{CT} = (A, B, C_1, \ldots, C_\ell)$ and $\mathsf{sk}_{\mathsf{ID}_{|\ell}} = (d', d_1, \ldots, d_\ell)$ and return

$$A \cdot \frac{\prod_{i=1}^{\ell} e(C_i, d_i)}{e(B, d')} = \mathsf{M}.$$

Theorem 2.1 ([3]). *If the DBDH assumption holds in the bilinear group $(p, \mathbb{G}, \mathbb{G}_t, e)$, then the BB-HIBE scheme over $(p, \mathbb{G}, \mathbb{G}_t, e)$ is IND-sID-CPA secure.*

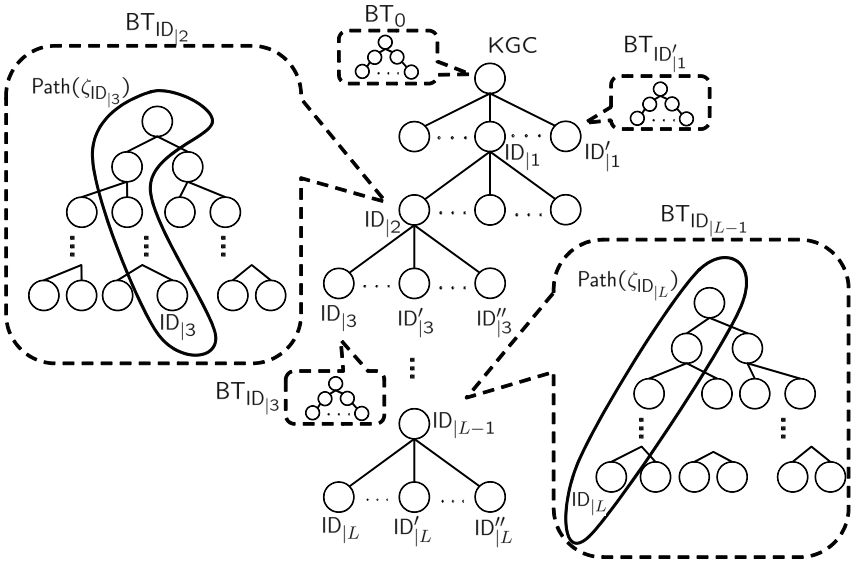

Figure 2: The Hierarchical Structures: Each user (except the last level user) in the hierarchy of IBE system has his own binary tree for revocation functionality.

3 Syntax and Security Model of RHIBE

We give formal definitions of the *hierarchical identity-based encryption with efficient revocation* scheme, which is simply called the Revocable Hierarchical Identity-Based Encryption (RHIBE) scheme, and its security by extending those of the *revocable IBE* in [2]. An RHIBE scheme consists of seven algorithms: Setup, KeyGen, KeyUp, DKG, Enc, Dec, and Revk. Roughly speaking, the Setup algorithm is run by the trusted authority called the Key Generation Center (KGC) for system parameters and a corresponding master secret key. When a user (possibly the KGC) generates a child's key, it can run the KeyGen algorithm with its secret key (the master secret key for the KGC) and a child's identity. If some users are revoked, the identities of revoked users should be updated along with each user's revocation time period. A $(\ell-1)$-th level user $\mathsf{ID}_{|\ell-1}$ issues key update information $\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}$ at every time period T by running the KeyUp algorithm. This information is managed by the binary tree $\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}$ which is served in the state $\mathsf{st}_{\mathsf{ID}_{|\ell-1}}$. While performing revocation, senders do not need to be updated and can encrypt a message using the receiver's identity, a time period that the receiver can decrypt, and the Enc algorithm. Every user $\mathsf{ID}_{|\ell}$ that is not revoked on time T can create a decryption key $\mathsf{dk}_{\mathsf{ID}_{\ell},\mathsf{T}}$ from $\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}$ by running the DKG algorithm, so that only a non-revoked user $\mathsf{ID}_{|\ell}$ (on time T) can decrypt an encrypted message for the identity $\mathsf{ID}_{|\ell}$ and the time period T by using the decryption key $\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}}$ and the Dec algorithm. Figure 2 explains the hierarchical structure with binary tree structures and we also provide the formal syntax of the RHIBE scheme below.

Definition 3.1. *An RHIBE scheme consists of seven algorithms:* Setup, KeyGen, KeyUp, DKG, Enc, Dec, *and* Revk. *The specification of each algorithm is as follows:*

Setup(λ, N, L): *It takes a security parameter λ, maximum number of users in each level N, and maximum length of the hierarchy of identity L as input. It then outputs a master public key* mpk, *a master secret key* msk, *initial state* st_0, *and an empty revocation list* RL. *We assume that the message space $\mathcal{M}$ and the identity space $\mathcal{I}$ (which is a union of all levels of identities), time space $\mathcal{T}$, and ciphertext space $\mathcal{CT}$ are contained in* mpk.

$\mathsf{KeyGen}(\mathsf{sk}_{\mathsf{ID}_{|\ell}}, \mathsf{st}_{\mathsf{ID}_{|\ell}}, \mathsf{ID}_{|\ell+1}, \mathsf{mpk})$*: On input of a private key* $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$ *and state* $\mathsf{st}_{\mathsf{ID}_{|\ell}}$*, children's identity* $\mathsf{ID}_{|\ell+1}$ *and the master public key* mpk*, it outputs a private key* $\mathsf{sk}_{\mathsf{ID}_{|\ell+1}}$ *and updates state* $\mathsf{st}_{\mathsf{ID}_{|\ell}}$*.*

$\mathsf{KeyUp}(\mathsf{sk}_{\mathsf{ID}_{|\ell}}, \mathsf{T}, \mathsf{RL}_{\mathsf{ID}_{|\ell}}, \mathsf{st}_{\mathsf{ID}_{|\ell}}, \mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}, \mathsf{mpk})$*: It takes a private key* $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$ *of* $\mathsf{ID}_{|\ell} \in \mathcal{I}$*, key update time* $\mathsf{T} \in \mathcal{T}$*, a revocation list* $\mathsf{RL}_{\mathsf{ID}_{|\ell}}$*, state* $\mathsf{st}_{\mathsf{ID}_{|\ell}}$ *of* $\mathsf{ID}_{|\ell}$*, key update* $\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}$ *published by* $\mathsf{ID}_{|\ell-1}$*, and the master public key* mpk *as input, and outputs a key update* $\mathsf{ku}_{\mathsf{ID}_{|\ell},\mathsf{T}}$ *or* $\perp$ *(when* $\mathsf{ID}_{|\ell}$ *is revoked).*

$\mathsf{DKG}(\mathsf{sk}_{\mathsf{ID}_{|\ell}}, \mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}, \mathsf{T}, \mathsf{mpk})$*: Given private key* $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$*, key update* $\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}$*, and the master public key* mpk*, it outputs a decryption key* $\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}}$ *that can be used during a time period* T *or* $\perp$ *that means the identity* $\mathsf{ID}_{|\ell}$ *is revoked for some time period* $\mathsf{T}' \leq \mathsf{T}$*.*

$\mathsf{Enc}(\mathsf{ID}_{|\ell}, \mathsf{T}, \mathsf{M}, \mathsf{mpk})$*: This algorithm takes an identity* $\mathsf{ID}_{|\ell} \in \mathcal{I}$*, time* $\mathsf{T} \in \mathcal{T}$*, a message* $\mathsf{M} \in \mathcal{M}$*, and the master public key* mpk *as input, and outputs a ciphertext* $\mathsf{CT} \in \mathcal{CT}$*.*

$\mathsf{Dec}(\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}}, \mathsf{CT}, \mathsf{mpk})$*: Given a decryption key* $\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}}$*, a ciphertext* $\mathsf{CT} \in \mathcal{CT}$*, and the master public key* mpk*, it outputs a message* $\mathsf{M} \in \mathcal{M}$ *or* $\perp$ *(invalid ciphertext).*

$\mathsf{Revk}(\mathsf{ID}_{|\ell}, \mathsf{T}, \mathsf{RL}_{\mathsf{ID}_{|\ell-1}}, \mathsf{st}_{\mathsf{ID}_{|\ell-1}})$*: It takes an identity* $\mathsf{ID}_{|\ell} \in \mathcal{I}$*, time* $\mathsf{T} \in \mathcal{T}$*, the revocation list* $\mathsf{RL}_{\mathsf{ID}_{|\ell-1}}$ *managed by* $\mathsf{ID}_{|\ell-1}$ *and state* $\mathsf{st}_{\mathsf{ID}_{|\ell-1}}$*, and updates the revocation list* $\mathsf{RL}_{\mathsf{ID}_{|\ell-1}}$ *by adding* $\mathsf{ID}_{|\ell}$ *as a revoked user at time* T*.*

Note that two algorithms KeyGen *and* Revk *are stateful, and if* $\mathsf{ID}_{|\ell}$ *is revoked at time period* T*, then all descendants should be revoked at the time* T*. For rigorous definition, if* $\ell = 0$*, then let* $\mathsf{sk}_{\mathsf{ID}_{|0}} = \mathsf{msk}$*,* $\mathsf{st}_{\mathsf{ID}_{|0}} = \mathsf{st}_0$*,* $\mathsf{RL}_{\mathsf{ID}_{|0}} = \mathsf{RL}_0$*,* $\mathsf{ku}_{\mathsf{ID}_{|-1},t}$ *be* msk*, and* $\mathsf{ku}_{\mathsf{ID}_{|0},\mathsf{T}} = \mathsf{ku}_{0,\mathsf{T}}$*.*

*We require the following correctness condition: For any output (*mpk*,*msk*) of* Setup*, any message* $\mathsf{M} \in \mathcal{M}$*, any identity* $\mathsf{ID}_{|\ell}$ *(of length* $\ell \leq L$*), any time* $\mathsf{T} \in \mathcal{T}$*, all possible states* $\{\mathsf{st}_{\mathsf{ID}_{|i}}\}_{i\in[0,\ell-1]}$ *and revocation lists* $\{\mathsf{RL}_{\mathsf{ID}_{|i}}\}_{i\in[0,\ell-1]}$*, if* $\mathsf{ID}_{|\ell}$ *is not revoked (and so all ancestors of* $\mathsf{ID}_{|\ell}$ *are not revoked) by time* T*, the following probability should be* 1*.*

$$\Pr\Big[\mathsf{Dec}(\mathsf{dk}_{\mathsf{ID}_{|\ell}}, \mathsf{Enc}(\mathsf{ID}_{|\ell}, \mathsf{T}, \mathsf{M}, \mathsf{mpk}), \mathsf{mpk}) = \mathsf{M} \,\Big|\, \begin{array}{ll} \textit{for } i \in [1,\ell], & \mathsf{KeyGen}(\mathsf{sk}_{\mathsf{ID}_{|i-1}}, \mathsf{st}_{\mathsf{ID}_{|i-1}}, \mathsf{ID}_{|i}, \mathsf{mpk}) \to \mathsf{sk}_{\mathsf{ID}_{|i}}, \\ & \mathsf{KeyUp}(\mathsf{sk}_{\mathsf{ID}_{|i-1}}, \mathsf{T}, \mathsf{RL}_{\mathsf{ID}_{|i-1}}, \mathsf{st}_{\mathsf{ID}_{|i-1}}, \mathsf{ku}_{\mathsf{ID}_{|i-2},\mathsf{T}}, \mathsf{mpk}) \to \mathsf{ku}_{\mathsf{ID}_{|i-1},\mathsf{T}}; \\ & \mathsf{DKG}(\mathsf{sk}_{\mathsf{ID}_{|\ell}}, \mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}, \mathsf{mpk}) \to \mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}}. \end{array} \Big]$$

We provide the security definition for a revocable HIBE scheme by extending Boldyreva et al.'s security definition for a revocable IBE scheme [2]. Our security definition allows the adversary to access several oracles: KeyGen, KeyUp, and Revk. We provide the precise definition of the oracles and new security notion for RHIBE using such oracles.

Definition 3.2 (IND-sRID-CPA)**.** *Let* $\mathcal{RHIBE}$*=(*Setup*,* KeyGen*,* KeyUp*,* DKG*,* Enc*,* Dec*,* Revk*) be an RHIBE scheme. First, we define three oracles.*

$\mathsf{KeyGen}(\cdot)$ *is the private key generation oracle. It takes an identity* $\mathsf{ID}_{|\ell}$ *of length* ℓ *as input, runs the* KeyGen *algorithm to get the private key* $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$*, and returns* $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$*.*

$\mathsf{KeyUp}(\cdot,\cdot)$ *is the key update oracle that takes time* T *and an identity* $\mathsf{ID}_{|\ell}$ *of length* ℓ *as input, runs the* KeyUp *algorithm to obtain the key update* $\mathsf{ku}_{\mathsf{ID}_{|\ell},\mathsf{T}}$*, and returns it.*

$\mathsf{Revk}(\cdot,\cdot,\cdot)$ *is the revocation oracle. It takes an identity* $\mathsf{ID}_{|\ell}$ *of length* ℓ*, its child identity* $\mathsf{ID}_{|\ell+1}$*, and time* T *as input, runs the the* Revk *algorithm to revoke* $\mathsf{ID}_{|\ell+1}$*, and updates* $\mathsf{RL}_{\mathsf{ID}_{|\ell}}$*.*

We assume that all oracles share a state. Next, we define the security of RHIBE, called IND-sRID-CPA security.

$\boxed{\mathbf{Exp}^{IND\text{-}sRID\text{-}CPA}_{\mathcal{RHIBE},\mathcal{A}}(\lambda)}$
$(\mathsf{ID}^*_{|\ell^*}, \mathsf{T}^*, state) \leftarrow \mathcal{A}(state)$
$(\mathsf{mpk}, \mathsf{msk}, \mathsf{RL}_0, \mathsf{st}_0) \leftarrow \mathsf{Setup}(\lambda, N, L)$
$(\mathsf{M}^*_0, \mathsf{M}^*_1, state) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\cdot),\mathsf{KeyUp}(\cdot,\cdot),\mathsf{Revk}(\cdot,\cdot,\cdot)}(state, \mathsf{mpk})$
$b \xleftarrow{\$} \{0,1\}$
$\mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{ID}^*_{|\ell^*}, \mathsf{T}^*, \mathsf{M}^*_b, \mathsf{mpk})$
$b' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\cdot),\mathsf{KeyUp}(\cdot,\cdot),\mathsf{Revk}(\cdot,\cdot,\cdot)}(state, \mathsf{mpk}, \mathsf{CT}^*)$
$Return \begin{cases} 1 \ if \ b = b' \\ 0 \ otherwise \end{cases}.$

There are three conditions that $\mathcal{A}$ should follow.

1. *The challenge messages M^*_0 and M^*_1 should have the same length.*
2. *$\mathsf{KeyUp}(\cdot,\cdot)$ and $\mathsf{Revk}(\cdot,\cdot,\cdot)$ can be queried at a time that is greater than or equal to the time of all previous queries; i.e. the adversary is allowed to query only in non-decreasing order of time. Also, $\mathsf{Revk}(\cdot,\cdot,\cdot)$ cannot be queried on a time T if $\mathsf{KeyUp}(\cdot,\cdot)$ was queried on T.*
3. *If $\mathsf{KeyGen}(\cdot)$ is queried on $\mathsf{ID}^*_{|\ell}$, which is an ancestor's identity of the challenge identity (that is, $\ell < \ell^*$), then $\mathsf{Revk}(\cdot,\cdot,\cdot)$ must be queried to revoke $\mathsf{ID}^*_{|\ell}$ at a time T for some $\mathsf{T} \leq \mathsf{T}^*$; hence, $\mathsf{ID}^*_{|\ell^*}$ is also directly revoked on the time $\mathsf{T} < \mathsf{T}^*$.*

The advantage of the adversary $\mathcal{A}$ is defined as

$$Adv^{IND\text{-}sRID\text{-}CPA}_{\mathcal{RHIBE},\mathcal{A}}(\lambda) = \left| \Pr[\mathbf{Exp}^{IND\text{-}sRID\text{-}CPA}_{\mathcal{RHIBE},\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right|.$$

If the function $Adv^{IND\text{-}sRID\text{-}CPA}_{\mathcal{RHIBE},\mathcal{A}}$ is negligible in the security parameter λ, we say that the scheme $\mathcal{RHIBE}$ is IND-sRID-CPA secure.

4 Our Construction

In this section, we propose our main construction for the RHIBE scheme. For revocation, we use the same methodology using binary structures as that used in all prior scalable RIBE schemes. In the RHIBE scheme, each intermediate level user $\mathsf{ID}_{|\ell-1}$ has its own binary tree $\mathtt{BT}_{\mathsf{ID}_{|\ell-1}}$ for revoking capabilities and issues the key update $\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}$ at each time period T. Then, a non-revoked child user $\mathsf{ID}_{|\ell}$ can generate $\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}}$ from the key update $\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}$ and its secret key $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$.

Before providing our construction, we first define several notations for simple description of the proposed construction, which is given in Table 1. Our RHIBE scheme is based on the BB-HIBE scheme [3]. As we mentioned before, a trivial approach for delegation of revocation functionality will end up with exponential secret key size in the hierarchical level. Therefore, our main contribution is to propose an efficient way of dealing with delegation of revoking capabilities and show that the proposed methodology does not harm the semantic security of the underlying BB-HIBE scheme; that is, in the security proof, we give a reduction to the IND-sID-CPA security of the BB-HIBE scheme.

For a decryption key of a user $\mathsf{ID}_{|\ell}$ at a time period T, we use a hierarchical extension as follows:

$$\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}} = \left(g_2^{\alpha} \cdot (g_1^{\mathsf{T}} h_0)^{s_0} \cdot \prod_{i \in [1,\ell]} (g_1^{\mathsf{I}_i} h_i)^{s_i}, g^{s_0}, g^{s_1} \ldots, g^{s_\ell} \right) \in \mathbb{G}^{\ell+2},$$

where $g, g_1, g_2, h_0, \ldots, h_\ell$ are public parameters, g_2^{α} is a master key, $\mathsf{ID}_{|\ell} = (\mathsf{I}_1, \ldots, \mathsf{I}_\ell)$, and $s_0, \ldots, s_\ell$ are random integers chosen from $\mathbb{Z}_p$. If we use the notation in Table 1, then

$$\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}} = \left(g_2^{\alpha} \cdot \overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{|\ell})^{\overrightarrow{s}}, g^{\overrightarrow{s}} \right) \text{ where } \overrightarrow{s} = (s_0, \ldots, s_\ell).$$

Preparation	Notation	Meaning
For $g \in \mathbb{G}$ and $\overrightarrow{r} = (r_1, \ldots, r_m) \in \mathbb{Z}_p^m$,	$g^{\overrightarrow{r}}$	$(g^{r_1}, \ldots, g^{r_m}) \in \mathbb{G}^m$
For $(g_1, \ldots, g_m) \in \mathbb{G}^m$ and $\overrightarrow{r} = (r_1, \ldots, r_m) \in \mathbb{Z}_p^m$,	$(g_1, \ldots, g_m)^{\overrightarrow{r}}$	$\prod_{i=1}^m g_i^{r_i} \in \mathbb{G}$
For $g^{\overrightarrow{r}} = (g^{r_1}, \ldots, g^{r_m})$ and $g^{\overrightarrow{s}} = (g^{s_1}, \ldots, g^{s_m}) \in \mathbb{G}^m$,	$g^{\overrightarrow{r}} \circ g^{\overrightarrow{s}}$	$(g^{r_1} \cdot g^{s_1}, \ldots, g^{r_m} \cdot g^{s_m}) \in \mathbb{G}^m$
For $g^{\overrightarrow{r}_1}, \ldots, g^{\overrightarrow{r}_m} \in \mathbb{G}^m$,	$\bigotimes_{i=1}^m g^{\overrightarrow{r}_i}$	$g^{\overrightarrow{r}_1} \circ \cdots \circ g^{\overrightarrow{r}_m} \in \mathbb{G}^m$
For $g_1, h_0, \ldots, h_L \in \mathbb{G}$ and $\forall i \in [0, L]$,	$F_i(x)$	$g_1^x h_i \in \mathbb{G}$
For $\mathsf{T} \in \mathcal{T}$ and $\mathsf{ID}_{\|\ell} = (\mathsf{I}_1, \ldots, \mathsf{I}_\ell) \in \mathcal{I}$,	$\overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{\|\ell})$	$(F_0(\mathsf{T}), F_1(\mathsf{I}_1), \ldots, F_\ell(\mathsf{I}_\ell)) \in \mathbb{G}^{\ell+1}$
For $\mathsf{ID}_{\|\ell} = (\mathsf{I}_1, \ldots, \mathsf{I}_\ell) \in \mathcal{I}$,	$\overrightarrow{F}(*, \mathsf{ID}_{\|\ell})$	$(1_{\mathbb{G}}, F_1(\mathsf{I}_1), \ldots, F_\ell(\mathsf{I}_\ell)) \in \mathbb{G}^{\ell+1}$

Table 1: Quick notations

Note that $\overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{|\ell})$ is a vector in $\mathbb{G}^{\ell+1}$ so that $\overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{|\ell})^{\overrightarrow{s}}$ is a group element in $\mathbb{G}$. Information about decryption keys are divided into secret keys and key updates. In particular, the master key g_2^α is randomly divided into two parts R_θ and g_2^α / R_θ, where we will explain θ later. The secret key of $\mathsf{ID}_{|\ell}$ contains information about

$$\left(R_\theta \cdot \prod_{i \in [1,\ell]} (g_1^{\mathsf{I}_i} h_i)^{s_i}, 1_{\mathbb{G}}, g^{s_1} \ldots, g^{s_\ell}\right) = \left(R_\theta \overrightarrow{F}(*, \mathsf{ID}_{|\ell})^{\overrightarrow{s}'}, g^{\overrightarrow{s}'}\right),$$

where $\overrightarrow{s}' = (0, s_1, \ldots, s_\ell) \in \mathbb{Z}_p^{\ell+1}$, and the key update for a time T, which is managed by $\mathsf{ID}_{|\ell-1}$, contains information about

$$\left((g_2^\alpha / R_\theta) \cdot (g_1^{\mathsf{T}} h_0)^{s_0} \cdot \prod_{i \in [1,\ell-1]} (g_1^{\mathsf{I}_i} h_i)^{s_i}, g^{s_0}, \ldots, g^{s_{\ell-1}}, 1_{\mathbb{G}}\right)$$

$$= \left((g_2^\alpha / R_\theta) \cdot \overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{|\ell-1})^{\overrightarrow{s}''}, g^{\overrightarrow{s}''}\right),$$

where $\overrightarrow{s}'' = (s_0, s_1, \ldots, s_{\ell-1}, 0) \in \mathbb{Z}_p^{\ell+1}$. Let R_θ be assigned in the node θ in $\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}$, $\zeta_{\mathsf{ID}_{|\ell}}$ be the leaf node assigned for $\mathsf{ID}_{|\ell}$ in $\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}$, and $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}})$ be the path from $\zeta_{\mathsf{ID}_{|\ell}}$ to the root node in $\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}$. If $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$ contains the above form for all θ on $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}})$ and key update on time T contains the above form for all θ in $\mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}, \mathsf{RL}_{\mathsf{ID}_{|\ell-1}}, \mathsf{T})$, then a non-revoked user $\mathsf{ID}_{|\ell}$ can generate the corresponding decryption key by a simple product of the above two forms since there exists at least one θ in $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}}) \cap \mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}, \mathsf{RL}_{\mathsf{ID}_{|\ell-1}}, \mathsf{T})$.

If we consider revocable IBE schemes, the above method is sufficient. However, when constructing the RHIBE scheme, we should consider the fact that users can generate valid key updates for children *only* during the time period in which they are not revoked. This implies that both secret keys and key updates should contain information about all ancestor's secret keys and key updates. To this end, we recursively define children's secret keys and key updates from parents' secret key and key update. Note that if we use binary tree structures for revocation, the KGC cannot directly generate secret keys of descendants (except for the first-level user) since the master key parts of decryption keys of descendants are randomly divided into two parts (each for the secret key and key update) according to the binary tree structure managed by their parents, which are also intermediate-level users, but the KGC does not know of such a binary tree structure. Therefore, the secret key and key updates have to be recursively defined, which makes the situation more complicated.

A simple example helps to explain our construction for delegating the revocation functionality. Assume that a user $\mathsf{ID}_{|1} = \mathsf{I}_1$ is not revoked on time T. The secret key of $\mathsf{ID}_{|1}$ and the key update $\mathsf{ku}_{0,\mathsf{T}}$ generated by the KGC are

$$\{(R_\theta \cdot (g_1^{\mathsf{I}_1} h_1)^{\gamma_1}, g^{\gamma_1})\}_{\theta \in \mathsf{Path}(\zeta_{\mathsf{ID}_{|1}})} \text{ and } \{((g_2^\alpha / R_\theta) \cdot (g_1^{\mathsf{T}} h_0)^{\gamma_0}, g^{\gamma_0})\}_{\theta \in \mathsf{KUNode}(\mathsf{BT}_0, \mathsf{RL}_0, \mathsf{T})},$$

respectively, where $\zeta_{\mathsf{ID}_{|1}}$ is a leaf node assigned for $\mathsf{ID}_{|1}$ by the KGC. The user $\mathsf{ID}_{|1}$ has its own binary tree structure $\mathsf{BT}_{\mathsf{ID}_{|1}}$ for revocation functionality. The revocation methodology used by $\mathsf{ID}_{|1}$ is the same as

for the KGC. Whenever a child $\mathsf{ID}_{|2}$ registers in the system, it randomly assigns a leaf node $\zeta_{\mathsf{ID}_{|2}}$ of $\mathtt{BT}_{\mathsf{ID}_{|1}}$ for $\mathsf{ID}_{|2}$. For all nodes θ' in $\mathsf{Path}(\zeta_{\mathsf{ID}_{|2}})$, assign random values $R_{\theta'}$. Assume that the secret key of $\mathsf{ID}_{|1}$ is $\{(d'_\theta, d_\theta) := (R_\theta \cdot (g_1^{\mathsf{I}_1} h_1)^{\gamma_1}, g^{\gamma_1})\}_{\theta \in \mathsf{Path}(\zeta_{\mathsf{ID}_{|1}})}$. The secret key $\mathsf{sk}_{\mathsf{ID}_{|2}}$ of a child $\mathsf{ID}_{|2} := (\mathsf{I}_1, \mathsf{I}_2)$ is generated as

$$\begin{array}{ll} & \{(d'_\theta \cdot (g_1^{\mathsf{I}_2} h_2)^{\gamma_2},\ 1_{\mathbb{G}},\ d_\theta,\ g^{\gamma_2})\}_{\theta \in \mathsf{Path}(\zeta_{\mathsf{ID}_{|1}})} \\ \text{and} & \{(R_{\theta'} \cdot \prod_{i\in[1,2]} (g_1^{\mathsf{I}_i} h_i)^{\gamma'_i},\ 1_{\mathbb{G}},\ g^{\gamma'_1},\ g^{\gamma'_2})\}_{\theta' \in \mathsf{Path}(\zeta_{\mathsf{ID}_{|2}})}. \end{array}$$

We use notation $\overrightarrow{d}^{(1,j)}_{\mathsf{ID}_{|2}}$ to denote a vector $(d'_\theta \cdot (g_1^{\mathsf{I}_2} h_2)^{s_2},\ 1_{\mathbb{G}},\ d_\theta,\ g^{s_2}) \in \mathbb{G}^4$, where θ is the j-th level node, and notation $\overrightarrow{d}^{(2,j)}_{\mathsf{ID}_{|2}}$ to denote a vector $(R_{\theta'} \cdot \prod_{i\in[1,2]} (g_1^{\mathsf{I}_i} h_i)^{\gamma_i},\ 1_{\mathbb{G}},\ g^{\gamma_1},\ g^{\gamma_2}) \in \mathbb{G}^4$, where θ' is the j-th level node.

The key update on T is computed as follows. Assume that $\mathsf{ID}_{|1}$ is not revoked on time T. Then, $\mathsf{ID}_{|1}$ can choose a node θ in $\mathsf{Path}(\zeta_{\mathsf{ID}_{|1}}) \cap \mathsf{KUNode}(\mathtt{BT}_0, \mathsf{RL}_0, \mathsf{T})$. Let Lv_1 be the level of θ in $\mathtt{BT}_0$. Note that all nodes in $\mathsf{Path}(\zeta_{\mathsf{ID}_{|1}})$ have different levels, so we can identify nodes from their corresponding levels. Then, $(f_1, f_2) := ((g_2^\alpha / R_\theta) \cdot (g_1^{\mathsf{T}} h_0)^{s_0}, g^{s_0})$ is a valid key update for the first level users including $\mathsf{ID}_{|1}$, and the key update for children of $\mathsf{ID}_{|1}$ is generated as

$$\{(\{\mathsf{Lv}_1\},\ (f_1 / R_{\theta'})(g_1^{\mathsf{I}_1} h_1)^\delta,\ f_2,\ g^\delta,\ 1_{\mathbb{G}})\}_{\theta' \in \mathsf{KUNode}(\mathtt{BT}_{\mathsf{ID}_{|1}}, \mathsf{RL}_{\mathsf{ID}_{|1}}, \mathsf{T})}.$$

We use notation $\overrightarrow{f}_{\mathsf{ID}_{|1},\theta'}$ to denote a vector $((f_1 / R_{\theta'})(g_1^{\mathsf{I}_1} h_1)^\delta,\ f_2,\ g^\delta,\ 1_{\mathbb{G}}) \in \mathbb{G}^4$.

The decryption key of $\mathsf{ID}_{|2}$ on time T is generated as follows. First, it identifies the subkey part of $\mathsf{sk}_{\mathsf{ID}_{|1}}$ that is used by its parent for delegation on time T. (It can see from Lv_1 in $\mathsf{ku}_{\mathsf{ID}_{|1},\mathsf{T}}$.) Let θ be the level Lv_1 node on $\mathsf{Path}(\zeta_{\mathsf{ID}_{|1}})$. Next, if $\mathsf{ID}_{|2} = (\mathsf{I}_1, \mathsf{I}_2)$ is not revoked, it can choose a node θ' on $\mathsf{Path}(\zeta_{\mathsf{ID}_{|2}}) \cap \mathsf{KUNode}(\mathtt{BT}_{\mathsf{ID}_{|1}}, \mathsf{RL}_{\mathsf{ID}_{|1}}, \mathsf{T})$. Let Lv_2 be the level of θ'. It then generates the decryption key as

$$\begin{array}{ll} & \bigotimes_{i=1}^{2} (\overrightarrow{d}^{(i,\mathsf{Lv}_i)}_{\mathsf{ID}_{|2}}) \ \circ\ \overrightarrow{f}_{\mathsf{ID}_{|1},\theta'} \\ = & \Big(\Big(d'_\theta (g_1^{\mathsf{I}_2} h_2)^{\gamma_2}\Big)\Big(R_{\theta'} \prod_{i\in[1,2]} (g_1^{\mathsf{I}_i} h_i)^{\gamma'_i}\Big)\Big(f_1 / R_{\theta'} (g_1^{\mathsf{I}_1} h_1)^\delta\Big),\ f_2,\ d_\theta g^{\gamma'_1} g^\delta,\ g^{\gamma_2} g^{\gamma'_2}\Big). \end{array}$$

A simple calculation shows that the above decryption key has the desired form

$$\Big(g_2^\alpha \cdot (g_1^{\mathsf{T}} h_0)^{s_0} (g_1^{\mathsf{I}_i} h_1)^{\gamma_1 + \gamma'_1 + \delta} (g_1^{\mathsf{I}_i} h_2)^{\gamma_2 + \gamma'_2},\ g^{s_0},\ g^{\gamma_1 + \gamma'_1 + \delta},\ g^{\gamma_2 + \gamma'_2}\Big).$$

In a similar way, we can define the secret keys and key updates for users from other levels. However, the security of the above construction is not easy to prove since descendants have a great deal of information about the ancestors' secret keys. In particular, in the security model the adversary can query $\mathsf{sk}_{\mathsf{ID}^*_{|\ell^*+1}}$, but the simulator may not generate such a secret key without knowing $\mathsf{sk}_{\mathsf{ID}^*_{|\ell^*}}$ since secret key generation algorithm is recursively defined. (When the challenge identity $\mathsf{ID}^*_{|\ell^*}$ is not revoked on the challenge time T^*, the simulator cannot generate $\mathsf{sk}_{\mathsf{ID}^*_{|\ell^*}}$. If not, the simulator can generate the $\mathsf{dk}_{\mathsf{ID}^*_{|\ell^*},\mathsf{T}^*}$ by itself from $\mathsf{sk}_{\mathsf{ID}^*_{\ell^*}}$ and $\mathsf{ku}_{\mathsf{ID}_{\ell^*-1},\mathsf{T}}$ so that the simulator can solve the underlying problem without a help of the adversary.) To circumvent this obstacle, we slightly modify the above construction by adding re-randomization processes in the KeyGen and KeyUp algorithms. Hence, the simulator can generate $\mathsf{sk}_{\mathsf{ID}^*_{|\ell^*+1}}$ even when it does not know $\mathsf{sk}_{\mathsf{ID}^*_{|\ell^*}}$ since all randomness used in $\mathsf{sk}_{\mathsf{ID}^*_{|\ell^*+1}}$ is independent from $\mathsf{sk}_{\mathsf{ID}^*_{|\ell^*}}$.

Now we describe our RHIBE construction. In our construction, each user $\mathsf{ID}_{|\ell}$ keeps state information $\mathsf{st}_{\mathsf{ID}_{|\ell}}$ including a binary tree $\mathtt{BT}_{\mathsf{ID}_{|\ell}}$. We sometimes use $\mathtt{BT}_{\mathsf{ID}_{|\ell}}$ to precisely indicate the state information associated with the binary tree. $\mathsf{st}_{\mathsf{ID}_{|\ell}}$ contains the randomness $\{\tilde{R}_{(i,j)}\}_{(i,j)\in[1,\ell-1]\times[1,n]}$ used for the re-randomization process as well. All states are initialized as empty sets.

$\mathsf{Setup}(\lambda, N, L)$: We assume, without loss of generality, that $N = 2^n$ for some n. Randomly choose group elements $g, g_2, h_0, \ldots, h_L \xleftarrow{\$} \mathbb{G}$ and an integer $\alpha \xleftarrow{\$} \mathbb{Z}_p$. Set $\mathsf{mpk} = \{g, g_1 = g^\alpha, g_2, h_0 \ldots, h_L\}$ and $\mathsf{msk} = \{g_2^\alpha\}$.

$\mathsf{KeyGen}(\mathsf{sk}_{\mathsf{ID}_{|\ell}}, \mathsf{st}_{\mathsf{ID}_{|\ell}}, \mathsf{ID}_{|\ell+1}, \mathsf{mpk})$: According to the value ℓ, this algorithm is differently defined.

$\ell = 0$: Note that $\mathsf{sk}_{\mathsf{ID}_{|0}} = \mathsf{msk}$ and $\mathsf{st}_{\mathsf{ID}_{|0}} = \mathsf{st}_0$. Randomly choose an unassigned leaf ζ from $\mathtt{BT}_0$, and store $\mathsf{ID}_{|1}$ in the node ζ. We sometimes use a notation ζ_{ID} to precisely indicate that the node ζ is associated with ID.

For all $\theta \in \mathsf{Path}(\zeta_{\mathsf{ID}_{|1}}) \subset \mathtt{BT}_0$,

1. Recall R_θ from θ in $\mathtt{BT}_0$ if it is defined. Otherwise, $R_\theta \xleftarrow{\$} \mathbb{G}$ and store it in the node $\theta \in \mathtt{BT}_0$.
2. Choose $\gamma_\theta \xleftarrow{\$} \mathbb{Z}_p$ and compute $\overrightarrow{d}^{(1,j)}_{\mathsf{ID}_{|1}} := (R_\theta \cdot F_1(\mathsf{ID}_{|1})^{\gamma_\theta}, 1_{\mathbb{G}},\ g^{\gamma_\theta}) \in \mathbb{G}^3$, where j is the level of θ on the path $\mathsf{Path}(\zeta_{\mathsf{ID}_{|1}})$.

Return $\mathsf{sk}_{\mathsf{ID}_{|1}} = \{\overrightarrow{d}^{(1,j)}_{\mathsf{ID}_{|1}} \in \mathbb{G}^3\}_{j\in[1,n]}$.

$\ell > 0$: Randomly choose an unassigned leaf ζ from $\mathtt{BT}_{\mathsf{ID}_{|\ell}}$, and store $\mathsf{ID}_{|\ell+1}$ in the node $\zeta_{\mathsf{ID}_{|\ell+1}}$. Parse $\mathsf{sk}_{\mathsf{ID}_{|\ell}} = \{\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell}} \in \mathbb{G}^{\ell+2}\}_{(i,j)\in[1,\ell]\times[1,n]}$.

For all $(i,j) \in [1,\ell] \times [1,n]$,

1. Recall $\tilde{R}_{(i,j)}$ from $\mathsf{st}_{\mathsf{ID}_{|\ell}}$ if it is defined. Otherwise, $\tilde{R}_{(i,j)} \xleftarrow{\$} \mathbb{G}$ and store it in $\mathsf{st}_{\mathsf{ID}_{|\ell}}$.
2. Choose $\overrightarrow{\gamma}_{(i,j)} \xleftarrow{\$} \{0\} \times \mathbb{Z}_p^{\ell+1}$ and compute $\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell+1}} := (\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell}},\ 1_{\mathbb{G}}) \circ (\tilde{R}_{(i,j)} \cdot \overrightarrow{F}(*, \mathsf{ID}_{|\ell+1})^{\overrightarrow{\gamma}_{(i,j)}},\ g^{\overrightarrow{\gamma}_{(i,j)}}) \in \mathbb{G}^{\ell+3}$.

For all $\theta \in \mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell+1}}) \subset \mathtt{BT}_{\mathsf{ID}_{|\ell}}$,

1. Recall R_θ from the corresponding node θ in $\mathtt{BT}_{\mathsf{ID}_{|\ell}}$ if it is defined. Otherwise, $R_\theta \xleftarrow{\$} \mathbb{G}$ and store it in the node θ.
2. Choose $\overrightarrow{\gamma}_\theta \xleftarrow{\$} \{0\} \times \mathbb{Z}_p^{\ell+1}$ and compute $\overrightarrow{d}^{(\ell+1,j)}_{\mathsf{ID}_{|\ell+1}} := (R_\theta \cdot \overrightarrow{F}(*, \mathsf{ID}_{|\ell+1})^{\overrightarrow{\gamma}_\theta},\ g^{\overrightarrow{\gamma}_\theta}) \in \mathbb{G}^{\ell+3}$, where j is the level of θ in the tree $\mathtt{BT}_{\mathsf{ID}_{|\ell}}$.

Return $\mathsf{sk}_{\mathsf{ID}_{|\ell+1}} = \{\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell+1}} \in \mathbb{G}^{\ell+3}\}_{(i,j)\in[1,\ell+1]\times[1,n]}$.

$\mathsf{KeyUp}(\mathsf{sk}_{\mathsf{ID}_{|\ell}}, \mathsf{T}, \mathsf{RL}_{\mathsf{ID}_{|\ell}}, \mathsf{st}_{\mathsf{ID}_{|\ell}}, \mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}, \mathsf{mpk})$: According to the value ℓ, this algorithm is differently defined.

$\ell = 0$: Note that $\mathsf{sk}_{\mathsf{ID}_{|0}} = \mathsf{msk}$, $\mathsf{RL}_{\mathsf{ID}_{|0}} = \mathsf{RL}_0$, $\mathsf{st}_{\mathsf{ID}_{|0}} = \mathsf{st}_0$, and $\mathsf{ku}_{\mathsf{ID}_{|-1},\mathsf{T}} = \mathsf{msk}$. For all nodes $\theta \in \mathsf{KUNode}(\mathtt{BT}_0, \mathsf{RL}_0, \mathsf{T})$,

1. Recall R_θ from the node $\theta \in \mathtt{BT}_0$. Note that R_θ is already defined during the key generation process.
2. Choose $\delta_\theta \xleftarrow{\$} \mathbb{Z}_p$ and compute $\overrightarrow{f}_{0,\theta}$ by $((g_2^\alpha / R_\theta) F_0(\mathsf{T})^{\delta_\theta},\ g^{\delta_\theta},\ 1_{\mathbb{G}}) \in \mathbb{G}^3$.

Return $\mathsf{ku}_{0,\mathsf{T}} = \{\emptyset,\ \overrightarrow{f}_{0,\theta} \in \mathbb{G}^3\}_{\theta\in\mathsf{KUNode}(\mathtt{BT}_0,\mathsf{RL}_0,\mathsf{T})}$.

$\ell > 0$:

1. Parse $\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}$ as $\{\{\mathsf{Lv}_i\}_{i\in[1,\ell-1]},\ \overrightarrow{f}_{\mathsf{ID}_{|\ell-1},\theta} \in \mathbb{G}^{\ell+2}\}_{\theta\in\mathsf{KUNode}(\mathtt{BT}_{\mathsf{ID}_{|\ell-1}},\mathsf{RL}_{\mathsf{ID}_{|\ell-1}},\mathsf{T})}$. Note that if $\ell = 1$, then $\{\mathsf{Lv}_i\}_{i\in[1,0]}$ means $\emptyset$.
2. Identify one node $\tilde{\theta} \in \mathsf{KUNode}(\mathtt{BT}_{\mathsf{ID}_{|\ell-1}}, \mathsf{RL}_{\mathsf{ID}_{|\ell-1}}, \mathsf{T}) \cap \mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}})$ and set Lv_ℓ to be the level of $\tilde{\theta}$.
3. For all $i \in [1,\ell]$, recall $\tilde{R}_{(i,\mathsf{Lv}_i)}$ from $\mathsf{st}_{\mathsf{ID}_{|\ell}}$.

For all nodes $\theta \in \mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell}}, \mathsf{RL}_{\mathsf{ID}_{|\ell}}, \mathsf{T})$,

1. Recall R_θ from $\theta \in \mathsf{BT}_{\mathsf{ID}_{|\ell}}$. Note that R_θ is already defined during the key generation process.
2. Choose $\overrightarrow{\delta}_\theta \xleftarrow{\$} \mathbb{Z}_p^{\ell+1}$ and compute $\overrightarrow{f}_{\mathsf{ID}_{|\ell},\theta}$ by $(\overrightarrow{f}_{\mathsf{ID}_{|\ell-1},\tilde{\theta}},\ 1_{\mathbb{G}}) \circ ((R_\theta \prod_{i=1}^{\ell} \tilde{R}_{(i,\mathsf{Lv}_i)})^{-1} \overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{|\ell})^{\overrightarrow{\delta}_\theta},\ g^{\overrightarrow{\delta}_\theta},\ 1_{\mathbb{G}})$.

Return $\mathsf{ku}_{\mathsf{ID}_{|\ell},\mathsf{T}} = \{\{\mathsf{Lv}_i\}_{i\in[1,\ell]},\ \overrightarrow{f}_{\mathsf{ID}_{|\ell},\theta} \in \mathbb{G}^{\ell+3}\}_{\theta\in\mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell}},\mathsf{RL}_{\mathsf{ID}_{|\ell}},\mathsf{T})}$.

$\mathsf{DKG}(\mathsf{sk}_{\mathsf{ID}_{|\ell}}, \mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}, \mathsf{T}, \mathsf{mpk})$:

1. Parse $\mathsf{sk}_{\mathsf{ID}_{|\ell}} = \{\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell}} \in \mathbb{G}^{\ell+2}\}_{(i,j)\in[1,\ell]\times[1,n]}$ and $\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}} = \{\{\mathsf{Lv}_i\}_{i\in[1,\ell-1]},\ \overrightarrow{f}_{\mathsf{ID}_{|\ell-1},\theta} \in \mathbb{G}^{\ell+2}\}_{\theta\in\mathsf{J}}$.
2. If $\mathsf{J} \cap \mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}}) = \emptyset$, then return $\perp$. Otherwise, choose a node $\theta \in \mathsf{J} \cap \mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}})$ and let Lv_ℓ be the level of θ in $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}}) \subset \mathsf{BT}_{\mathsf{ID}_{|\ell-1}}$.
3. Compute and output $\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}} := \bigotimes_{i=1}^{\ell} (\overrightarrow{d}^{(i,\mathsf{Lv}_i)}_{\mathsf{ID}|\ell}) \circ \overrightarrow{f}_{\mathsf{ID}_{|\ell-1},\theta} \in \mathbb{G}^{\ell+2}$.

$\mathsf{Enc}(\mathsf{ID}_{|\ell}, \mathsf{T}, \mathsf{M}, \mathsf{mpk})$: Choose a random value $t \xleftarrow{\$} \mathbb{Z}_p$ and return

$$\mathsf{CT} = (\mathsf{M} \cdot e(g_1, g_2)^t,\ \ g^t,\ \ F_0(\mathsf{T})^t,\ F_1(\mathsf{I}_1)^t, \ldots, F_\ell(\mathsf{I}_\ell)^t).$$

$\mathsf{Dec}(\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}}, \mathsf{CT}, \mathsf{mpk})$: Parse $\mathsf{CT} = (A, B, C_0, C_1, \ldots, C_\ell)$ and $\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}} = (D',\ D_0,\ \ldots,\ D_\ell)$ and return

$$A \cdot \frac{\prod_{i=0}^{\ell} e(C_i, D_i)}{e(B, D')} = \mathsf{M}.$$

$\mathsf{Revoke}(\mathsf{ID}_{|\ell}, \mathsf{T}, \mathsf{RL}_{\mathsf{ID}_{|\ell-1}}, \mathsf{st}_{\mathsf{ID}_{|\ell-1}})$: Let ζ be the leaf node in $\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}$ associated with $\mathsf{ID}_{|\ell}$. Update the revocation list by $\mathsf{RL}_{\mathsf{ID}_{|\ell-1}} \leftarrow \mathsf{RL}_{\mathsf{ID}_{|\ell-1}} \cup \{(\zeta, \mathsf{T})\}$ and return the updated revocation list.

Efficiency. For encrypting to the ℓ-th level user, the ciphertext consists of $\ell + 2$ group elements in $\mathbb{G}$ and an element in $\mathbb{G}_t$. The decryption algorithm requires $\ell + 2$ pairings and $\ell + 3$ multiplications in $\mathbb{G}_t$. Each user in the ℓ-th level keeps $(\ell + 2)(\ell + 1) \log N$ group elements in $\mathbb{G}$ as its secret key.

5 Security Analysis

We provide a series of lemmas to thoroughly explain the forms of secret keys, key updates, and decryption keys well, and then give a theorem for the IND-sRID-CPA security of the proposed construction.

Lemma 5.1. *If a secret key $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$ is normally generated, it has the following form:*

$$\mathsf{sk}_{\mathsf{ID}_{|\ell}} \quad = \quad \left\{ \overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell}} \in \mathbb{G}^{\ell+2} \text{ for } (i,j) \in [1,\ell] \times [1,n] \right\},$$

where

1. $\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell}} = \left(R'_{(i,j)} \overrightarrow{F}(*, \mathsf{ID}_{|\ell})^{\overrightarrow{r}_{(i,j)}}, g^{\overrightarrow{r}_{(i,j)}} \right)$ *for a uniformly distributed vector* $\overrightarrow{r}_{(i,j)} \in \{0\} \times \mathbb{Z}_p^\ell$ *and uniformly distributed value* $R'_{(i,j)} \in \mathbb{G}$. *(This implies that all randomness used in a secret key is independent from the parent's secret key.)*

2. *For* $i \in [1, \ell-1] \times [1, n]$, *all children of* $\mathsf{ID}_{|\ell-1}$ *have the same value* $R'_{(i,j)}$.

3. $R'_{(\ell,j)}$ *is an associated value with the* j*-th level node on* $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}}) \subset \mathsf{BT}_{\mathsf{ID}_{|\ell-1}}$.

Proof. We prove the lemma using the mathematical induction methodology. When $\ell = 1$, the lemma is true since the output of KeyGen algorithm run by KGC is exactly of the form. Next, we assume that the lemma is true for $\ell \geq 2$, and then we show that the lemma is true for $\ell + 1$. If we run KeyGen algorithm with $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$ as input, we obtain the following equalities.

$$\begin{aligned}
\text{for } (i,j) \in [1,\ell] \times [1,n],\ \overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell+1}} &= (\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell}},\ 1_{\mathbb{G}}) \circ (\tilde{R}_{(i,j)} \overrightarrow{F}(*, \mathsf{ID}_{|\ell+1})^{\overrightarrow{\gamma}_{(i,j)}},\ g^{\overrightarrow{\gamma}_{(i,j)}}), \\
&\quad (\text{where } \overrightarrow{\gamma}_{(i,j)} \xleftarrow{\$} \{0\} \times \mathbb{Z}_p^{\ell+1}) \\
&= (R'_{(i,j)} \tilde{R}_{(i,j)} \overrightarrow{F}(*, \mathsf{ID}_{|\ell+1})^{(\overrightarrow{r}_{(i,j)},0)+\overrightarrow{\gamma}_{(i,j)}},\ g^{(\overrightarrow{r}_{(i,j)},0)+\overrightarrow{\gamma}_{(i,j)}}) \in \mathbb{G}^{\ell+3}. \\
\text{for } \theta \in \mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell+1}}),\ \overrightarrow{d}^{(\ell+1,j)}_{\mathsf{ID}_{|\ell+1}} &= (R_\theta \overrightarrow{F}(*, \mathsf{ID}_{|\ell+1})^{\overrightarrow{\gamma}_\theta},\ g^{\overrightarrow{\gamma}_\theta}) \in \mathbb{G}^{\ell+3},
\end{aligned}$$

where j is the level of θ in $\mathsf{BT}_{\mathsf{ID}_{|\ell}}$. In the process of KeyGen, for $(i,j) \in [1,\ell] \times [1,n]$, $\tilde{R}_{(i,j)}$, which is stored in $\mathsf{st}_{\mathsf{ID}_{|\ell}}$, is commonly used for all children $\mathsf{ID}_{|\ell+1}$, and $R'_{(i,j)}$ is also used independently from the children's identity by hypothesis. Furthermore, R_θ is an associated value, which is also stored in $\mathsf{st}_{\mathsf{ID}_{|\ell}}$, with the j-th level node θ on $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell+1}}) \subset \mathsf{BT}_{\mathsf{ID}_{|\ell}}$. From these facts and the above two equalities, we can see that $\mathsf{sk}_{\mathsf{ID}_{|\ell+1}}$ satisfies three conditions in the lemma. In particular, $\overrightarrow{\gamma}_{(i,j)}$, $\tilde{R}_{(i,j)}$, and R_θ are uniformly chosen from their domains, respectively, so that all randomness used in $\mathsf{sk}_{\mathsf{ID}_{|\ell}+1}$ are independent from those in $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$. Therefore, we complete the induction method, and so the proof. □

Lemma 5.2. *If a key update* $\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}$ *is normally generated, it has the following form:*

$$\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}} = \left\{ \{\mathsf{Lv}_i\}_{i\in[1,\ell-1]},\ \overrightarrow{f}_{\mathsf{ID}_{|\ell-1},\theta} \in \mathbb{G}^{\ell+2} \right\}_{\theta \in \mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}, \mathsf{RL}_{\mathsf{ID}_{|\ell-1}}, \mathsf{T})},$$

where for $\overrightarrow{s}_\theta \in \mathbb{Z}_p^\ell$ *and* $\overrightarrow{f}_{\mathsf{ID}_{|\ell-1},\theta} = \left(g_2^\alpha (R'_\theta \prod_{i=1}^{\ell-1} R'_{(i,\mathsf{Lv}_i)})^{-1} \overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{|\ell-1})^{\overrightarrow{s}_\theta},\ g^{\overrightarrow{s}_\theta},\ 1_{\mathbb{G}} \right)$, R'_θ *is an associated value with a node* θ *in* $\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}$, *and* $R'_{(i,j)}$ *is a value defined in the secret key* $\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell}}$ *of Lemma 5.1, which is the same for any children* $\mathsf{ID}_{|\ell}$ *of* $\mathsf{ID}_{|\ell-1}$. *Moreover, for some* $\mathsf{ID}_{|\ell}$, *if* $\theta \in \mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}, \mathsf{RL}_{\mathsf{ID}_{|\ell-1}}, \mathsf{T}) \cap \mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}})$, *then* $R'_\theta = R'_{(\ell,\mathsf{Lv}_\ell)}$, *where* Lv_ℓ *is the level of* θ *on* $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}}) \subset \mathsf{BT}_{\mathsf{ID}_{|\ell-1}}$, *and* $R'_{(\ell,\mathsf{Lv}_\ell)}$ *is a value defined in the secret key* $\overrightarrow{d}^{(\ell,\mathsf{Lv}_\ell)}_{\mathsf{ID}_{|\ell}}$ *of Lemma 5.1.*

Proof. First, we note that the statement in Lemma is well-defined; $\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}$ is generated by $\mathsf{ID}_{|\ell-1}$, but $R'_{(i,j)}$ is a value defined in the children's secret key $\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell}}$. However, by the second condition in Lemma 5.1, for $(i,j) \in [1,\ell-1] \times [1,n]$, $R'_{(i,j)}$ is independent from the children's identity so that the statement is well-defined.

When $\ell = 1$, it is straight from the output of KeyUp. We assume that Lemma is true for $\ell \geq 2$, and show that the case for $\ell + 1$ also holds. A user $\mathsf{ID}_{|\ell}$, who is not revoked, can generate a key update $\mathsf{ku}_{\mathsf{ID}_{|\ell},\mathsf{T}}$ by running KeyUp with $\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}$ as input. The output of KeyUp is

$$\mathsf{ku}_{\mathsf{ID}_{|\ell},\mathsf{T}} = \{\{\mathsf{Lv}_i\}_{i\in[1,\ell]},\ \overrightarrow{f}_{\mathsf{ID}_{|\ell},\theta} \in \mathbb{G}^{\ell+3}\}_{\theta \in \mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell}}, \mathsf{RL}_{\mathsf{ID}_{|\ell}}, \mathsf{T})},$$

where

$$\overrightarrow{f}_{\mathsf{ID}_{|\ell},\theta} := (\overrightarrow{f}_{\mathsf{ID}_{|\ell-1},\tilde{\theta}},\ 1_{\mathbb{G}}) \circ ((R_\theta \prod_{i=1}^{\ell} \tilde{R}_{(i,\mathsf{Lv}_i)})^{-1} \cdot \overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{|\ell})^{\overrightarrow{\delta}_\theta},\ g^{\overrightarrow{\delta}_\theta},\ 1_{\mathbb{G}})$$

and $\tilde{\theta}$ is the Lv_ℓ-th level node in $\mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}, \mathsf{RL}_{\mathsf{ID}_{|\ell-1}}, \mathsf{T}) \cap \mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}})$. Then, by hypothesis, $R'_\theta = R'_{(\ell,\mathsf{Lv}_\ell)}$, and so $\overrightarrow{f}_{\mathsf{ID}_{|\ell},\theta}$ is equal to

$$(g_2^\alpha(\prod_{i=1}^{\ell} R'_{(i,\mathsf{Lv}_i)})^{-1}\overrightarrow{F}(\mathsf{T},\mathsf{ID}_{|\ell-1})^{\overrightarrow{s}_{\tilde{\theta}}},\ g^{\overrightarrow{s}_{\tilde{\theta}}},\ 1_{\mathbb{G}},\ 1_{\mathbb{G}})\ \circ\ ((R_\theta\prod_{i=1}^{\ell}\tilde{R}_{(i,\mathsf{Lv}_i)})^{-1}\cdot\overrightarrow{F}(\mathsf{T},\mathsf{ID}_{|\ell})^{\overrightarrow{\delta}_\theta},\ g^{\overrightarrow{\delta}_\theta},\ 1_{\mathbb{G}}).$$

$$= (g_2^\alpha(R_\theta\prod_{i=1}^{\ell}\tilde{R}_{(i,\mathsf{Lv}_i)}R'_{(i,\mathsf{Lv}_i)})^{-1}\overrightarrow{F}(\mathsf{T},\mathsf{ID}_{|\ell})^{(\overrightarrow{s}_{\tilde{\theta}},0)+\overrightarrow{\delta}_\theta},\ g^{(\overrightarrow{s}_{\tilde{\theta}},0)+\overrightarrow{\delta}_\theta},\ 1_{\mathbb{G}}),$$

where for $(i,j)\in[1,\ell]\times[1,n]$, $R'_{(i,j)}$ is a value defined in the secret key $\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell}}$ of Lemma 5.1. As we showed in the proof of Lemma 5.1, for $(i,j)\in[1,\ell]\times[1,n]$, $\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell+1}} = (R'_{(i,j)}\tilde{R}_{(i,j)}\overrightarrow{F}(*,\mathsf{ID}_{|\ell+1})^{(\overrightarrow{r}_{(i,j)},0)+\overrightarrow{\gamma}_{(i,j)}}, g^{(\overrightarrow{r}_{(i,j)},0)+\overrightarrow{\gamma}_{(i,j)}})$ and $R'_{(i,j)}\tilde{R}_{(i,j)}$ is a value used in $\overrightarrow{f}_{\mathsf{ID}_{|\ell},\theta}$.

Moreover, R_θ is an associated with a node θ by the process of KeyUp and if for some $\mathsf{ID}_{|\ell+1}$, θ is in the $\mathsf{Lv}_{\ell+1}$-th level in $\mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell}}, \mathsf{RL}_{\mathsf{ID}_{|\ell}}, \mathsf{T})\cap\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell+1}})$, then we can see R_θ is also is used in the generation of decryption key part by $\overrightarrow{d}^{(\ell+1,\mathsf{Lv}_{\ell+1})}_{\mathsf{ID}_{|\ell+1}} = (R_{\theta_{(\ell+1,\mathsf{Lv}_{\ell+1})}}\overrightarrow{F}(*,\mathsf{ID}_{|\ell+1})^{\overrightarrow{\gamma}_{(\ell+1,\mathsf{Lv}_{\ell+1})}},\ g^{\overrightarrow{\gamma}_{(\ell+1,\mathsf{Lv}_{\ell+1})}})$. Therefore, we obtain the desired result. By the induction method, we complete the proof for all ℓ. □

Lemma 5.3. *If a decryption key $\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}}$ is normally generated, it has the following form:*

$$\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}} \quad = \quad (g_2^\alpha\overrightarrow{F}(\mathsf{T},\mathsf{ID}_{|\ell})^{\overrightarrow{s}}, g^{\overrightarrow{s}})\ \in\mathbb{G}^{\ell+2},$$

where $\overrightarrow{s}$ is a vector in $\mathbb{Z}_p^{\ell+1}$.

Proof. When θ is the Lv_ℓ-th level node in $\mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}, \mathsf{RL}_{\mathsf{ID}_{|\ell-1}}, \mathsf{T})\cap\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}})$, a decryption key $\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}}$ is computed by

$$\bigotimes_{i=1}^{\ell}(\overrightarrow{d}^{(i,\mathsf{Lv}_i)}_{\mathsf{ID}|\ell})\ \circ\ \overrightarrow{f}_{\mathsf{ID}_{|\ell-1},\theta}.$$

By Lemma 5.1 and Lemma 5.2, this is equal to

$$\bigotimes_{i=1}^{\ell}(R'_{(i,\mathsf{Lv}_i)}\overrightarrow{F}(*,\mathsf{ID}_{|\ell})^{\overrightarrow{r}_{(i,\mathsf{Lv}_i)}}, g^{\overrightarrow{r}_{(i,\mathsf{Lv}_i)}})\ \circ\ (g_2^\alpha(\prod_{i=1}^{\ell}R'_{(i,\mathsf{Lv}_i)})^{-1}\overrightarrow{F}(\mathsf{T},\mathsf{ID}_{|\ell-1})^{\overrightarrow{s}_\theta},\ g^{\overrightarrow{s}_\theta},\ 1_{\mathbb{G}})$$

$$= \left(g_2^\alpha\overrightarrow{F}(\mathsf{T},\mathsf{ID}_{|\ell})^{\sum_{i=1}^{\ell}\overrightarrow{r}_{(i,\mathsf{Lv}_\ell)}+\overrightarrow{s}_\theta},\ g^{\sum_{i=1}^{\ell}\overrightarrow{r}_{(i,\mathsf{Lv}_\ell)}+(\overrightarrow{s}_\theta,0)}\right).$$

□

Even though our KeyGen algorithm (KeyUp algorithm, respectively) is recursively defined, the above lemmas dictate that the secret key (key update, respectively) in each level has the same format and the randomness used in each level is totally independent from those in the other levels. This fact gives us an essential advantage when we construct a simulator in the security proof; when the simulator generates a secret key (key update, respectively), it is not necessary to generate all ancestor's secret keys (key updates, respectively), though KeyGen (KeyUp, respectively) is recursively defined in the real scheme. Instead, in the proof, the simulator can directly simulate with fresh randomness.

Theorem 5.1. *Assume that the original BB-HIBE scheme is IND-sID-CPA secure. Then, the proposed RHIBE scheme is IND-sRID-CPA secure.*

Proof. We will construct an attacker $\mathcal{B}$ of BB-HIBE with L level hierarchy by using an attacker $\mathcal{A}$ of our construction of RHIBE scheme with L level hierarchy. First, $\mathcal{B}$ receives the target identity $\mathsf{ID}^*_{|\ell^*}$ and time T^* from $\mathcal{A}$. Then, we separate adversarial types as follows: Let $\mathsf{ID}^*_{|1}, \ldots, \mathsf{ID}^*_{|\ell^*-1}$ be ancestors of $\mathsf{ID}^*_{|\ell^*}$.

- Type-0 adversary: The adversary does not issue any key extraction query for $\mathsf{ID}^*_{|1}, \ldots, \mathsf{ID}^*_{|\ell^*}$.
- Type-1 adversary: The adversary queries for the secret key of $\mathsf{ID}^*_{|1}$

  $\vdots$
- Type-i adversary: The adversary does not query for the secret key of $\mathsf{ID}^*_{|1}, \ldots, \mathsf{ID}^*_{|i-1}$, but does for $\mathsf{ID}^*_{|i}$.

  $\vdots$
- Type-ℓ^* adversary: The adversary queries for the secret key of $\mathsf{ID}^*_{|\ell^*}$, but she does not issue any key extraction query for all ancestor's secret key.

$\mathcal{B}$ can guess the adversarial type by tossing coins and his guess is information-theoretically hidden from the adversarial view so that $\mathcal{B}$ can success in his guess with at least $1/(L+1)$ probability. During simulation, $\mathcal{B}$ can generate only either secret key of the challenge identity or key update for the challenge identity. (If not, $\mathcal{B}$ can generate decryption key for the challenge ciphertext.) By guessing types of adversary, $\mathcal{B}$ can decide where the (unknown) master key part is contained between secret key and key update. (Recall that in our construction the master key is randomly divided into secret key and key update.)

$\mathcal{B}$ sends $\mathsf{ID}^*_{|\ell^*}$ to the IND-sID-CPA game challenger $\mathcal{C}$ as $\mathcal{B}$'s target identity. Then, $\mathcal{B}$ obtains the master public key $\mathsf{mpk} = \{g, g_1, g_2, h_1 \ldots, h_L\}$ of BB-HIBE along with the group description $(p, \mathbb{G}, \mathbb{G}_t, e)$. It chooses $c \xleftarrow{\$} \mathbb{Z}_p$, sets $h_0 = g_1^{-\mathsf{T}^*} g^c$, and sends the group description and $\{g, g_1, g_2, h_0, h_1 \ldots, h_L\}$ to $\mathcal{A}$ as public parameters of RHIBE scheme. The corresponding master key of the BB-HIBE scheme is $\{g_2^\alpha\}$, which is also the master key of our RHIBE scheme and unknown to $\mathcal{B}$, where $g_1 = g^\alpha$.

For other queries such as key update and secret key extraction queries, $\mathcal{B}$ responses according to his guess in the type of adversary. We will describe $\mathcal{B}$'s behaviors separately. However, regardless of the type of adversary, whenever receiving a key extraction or a key update query regarding $\mathsf{ID}_{|\ell}$, $\mathcal{B}$ assigns nodes $\zeta_{\mathsf{ID}_{|1}} \in \mathsf{BT}_0, \ldots, \zeta_{\mathsf{ID}_{|\ell}} \in \mathsf{BT}_{\mathsf{ID}_{|\ell}}$ at random if they are undefined.

Type-0 adversary

The KeyUp Oracle: Receive $\mathsf{ID}_{|\ell}$ and T. If $\mathsf{ID}_{|\ell}$ is not revoked (more precisely, for $i \in [1, \ell]$, $\mathsf{ID}_{|i}$ is a non-revoked user on the time T), then for $i \in [1, \ell]$ $\mathcal{B}$ can always choose a node θ on $\mathsf{Path}(\zeta_{\mathsf{ID}_{|i}}) \cap \mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|i-1}}, \mathsf{RL}_{\mathsf{ID}_{|i-1}}, \mathsf{T})$ and let Lv_i be the level of θ. $\mathcal{B}$ keeps $(\mathsf{ID}_{|i}, \theta, \mathsf{T})$ in its storage. When $\mathcal{B}$ is required to generate an another key update regarding an identity $\mathsf{ID}'_{|\ell'}$ on the same time period T, if $\mathsf{ID}'_{|i} = \mathsf{ID}_{|i}$, then $\mathcal{B}$ chooses the same node θ, and so uses the same Lv_i in key update.

For $(i, j) \in [1, \ell] \times [1, n]$, randomly choose $S_{(i,j)} \in \mathbb{G}$ and store them in $\mathsf{st}_{\mathsf{ID}_{|\ell}}$ if they are undefined. Otherwise, just recall them. For $\theta \in \mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell}}, \mathsf{RL}_{\mathsf{ID}_{|\ell}}, \mathsf{T})$, randomly choose $S_\theta \in \mathbb{G}$ and store it in the node θ in $\mathsf{BT}_{\mathsf{ID}_{|\ell}}$ if it is undefined. Otherwise, recall it. Randomly choose $\overrightarrow{s}_\theta \xleftarrow{\$} \mathbb{Z}_p^{\ell+1}$, compute

$$\overrightarrow{f}_{\mathsf{ID}_{|\ell},\theta} = \Big((S_\theta \prod_{i=1}^{\ell} S_{(i,\mathsf{Lv}_i)})^{-1} \overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{|\ell})^{\overrightarrow{s}_\theta},\ g^{\overrightarrow{s}_\theta},\ 1_{\mathbb{G}} \Big),$$

and return $\mathsf{ku}_{\mathsf{ID}_{|\ell},\mathsf{T}} = \Big\{ \{\mathsf{Lv}_i\}_{i\in[1,\ell]},\ \overrightarrow{f}_{\mathsf{ID}_{|\ell},\theta} \in \mathbb{G}^{\ell+3} \Big\}_{\theta \in \mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell}}, \mathsf{RL}_{\mathsf{ID}_{|\ell}}, \mathsf{T})}$.

The KeyGen Oracle: Start with receiving an identity $\mathsf{ID}_{|\ell+1}$. $\mathcal{B}$ sends $\mathsf{ID}_{|\ell+1}$ to $\mathcal{C}$ and receives the corresponding private key $(d', d_1, \ldots, d_{\ell+1})$ from $\mathcal{C}$. (Since $\mathsf{ID}_{|\ell+1} \notin \{\mathsf{ID}^*_{|1}, \ldots, \mathsf{ID}^*_{|\ell^*}\}$, $\mathcal{B}$ can always issue

such a key extraction query.) For $(i,j) \in [1,\ell] \times [1,n]$, recall $S_{(i,j)}$ from $\mathsf{st}_{\mathsf{ID}_{|\ell}}$ if it is defined. Otherwise, randomly choose $S_{(i,j)} \xleftarrow{\$} \mathbb{G}$ and store them in $\mathsf{st}_{\mathsf{ID}_{|\ell}}$. For a node $\theta \in \mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell+1}}) \subset \mathsf{BT}_{\mathsf{ID}_{|\ell}}$, recall S_θ if it is defined. Otherwise, randomly choose S_θ from $\mathbb{G}$ and store it in the node $\theta \in \mathsf{BT}_{\mathsf{ID}_{|\ell}}$. $\mathcal{B}$ computes $\mathsf{sk}_{\mathsf{ID}_{|\ell+1}} = \left\{ \overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell+1}} \in \mathbb{G}^{\ell+3} \text{ for } (i,j) \in [1,\ell+1] \times [1,n] \right\}$ as follows.

$$\begin{cases} \text{For } (i,j) \in [1,\ell] \times [1,n], & \overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell+1}} = \left(S_{(i,j)} \overrightarrow{F}(*, \mathsf{ID}_{|\ell+1})^{\overrightarrow{r}_{(i,j)}},\ g^{\overrightarrow{r}_{(i,j)}} \right), \\ & \text{where } \overrightarrow{r}_{(i,j)} \xleftarrow{\$} \{0\} \times \mathbb{Z}_p^{\ell+1}. \\ \text{For } (i,j) \in \{\ell+1\} \times [1,n], & \overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell+1}} = \left(S_\theta d' \overrightarrow{F}(*, \mathsf{ID}_{|\ell+1})^{\overrightarrow{r}_{(i,j)}},\ (1_{\mathbb{G}}, d_1, \ldots, d_{\ell+1}) \circ g^{\overrightarrow{r}_{(i,j)}} \right), \\ & \text{where } \overrightarrow{r}_{(i,j)} \xleftarrow{\$} \{0\} \times \mathbb{Z}_p^{\ell+1} \text{ and } \theta \text{ is a } j\text{-th level node on } \mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell+1}}). \end{cases}$$

Challenge Phase: $\mathcal{A}$ sends two messages M_0^* and $\mathsf{M}_1^* \in \mathcal{M}$. $\mathcal{B}$ transfers two messages to $\mathcal{C}$ as its challenge messages, and receives the challenge ciphertext $\mathsf{CT} = (A^*, B^*, C_1^*, \ldots, C_{\ell^*}^*)$ for the target identity $\mathsf{ID}^*_{|\ell^*}$ from $\mathcal{C}$. Then, $\mathcal{B}$ sends $\mathsf{CT}^* = (A^*, B^*, C_0^* = (B^*)^c, C_1^*, \ldots, C_\ell^*)$ to $\mathcal{A}$ as the challenge ciphertext. Lastly, $\mathcal{B}$ receives $\mathcal{A}$'s output bit and sends the same bit as its output.

Type-k adversary ($k \in [1, \ell^*]$) For type-k adversary, we carefully deal with queries for $\mathsf{ID}_{|\ell}$ such that $\mathsf{ID}_{|\ell-1} = \mathsf{ID}^*_{|i-1}$ for some $i \in [k, \ell^*]$. In the case of $\mathsf{ID}_{|\ell-1} = \mathsf{ID}^*_{|i-1}$, the leaf node $\zeta_{\mathsf{ID}_{|\ell}}$, which is assigned for $\mathsf{ID}_{|\ell}$, is in $\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}$ and the leaf node $\zeta_{\mathsf{ID}^*_{|i}}$, which is assigned for $\mathsf{ID}^*_{|i}$, is also in $\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}$. Therefore, $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}})$ may have an intersection with $\mathsf{Path}(\zeta_{\mathsf{ID}^*_{|i}})$. The type-$k$ adversary can obtain secret key for such $\mathsf{ID}_{|\ell}$, but not for key update on the challenge time T^*. For $\theta \in \mathsf{Path}(\zeta_{\mathsf{ID}^*_{|i}})$, we set the corresponding key update has the master key part, but the corresponding secret key does not. In the challenge time T^*, $\mathcal{B}$ does not need to generate key update for $\theta \in \mathsf{Path}(\zeta_{\mathsf{ID}^*_{|i}})$ and in the other time period $\mathcal{B}$ uses the selective security's technique to generate key update so that $\mathcal{B}$ can simulate all queries related to the node $\theta \in \mathsf{Path}(\zeta_{\mathsf{ID}^*_{|i}})$. For other nodes (that is, $\theta \notin \mathsf{Path}(\zeta_{\mathsf{ID}^*_{|i}})$), we set the corresponding secret key has the master key part, but the corresponding key update does not. For secret key queries, $\theta \notin \mathsf{Path}(\zeta_{\mathsf{ID}^*_{|i}})$ means that the corresponding identity is not $\mathsf{ID}^*_{|i}$ so that $\mathcal{B}$ can utilize his own key extraction oracle given from IND-sID-CPA challenge of BB-HIBE scheme. We provide the detailed description of $\mathcal{B}$ below.

The KeyUp Oracle: Take $\mathsf{ID}_{|\ell}$ and T as input. We divide cases according to $\mathsf{ID}_{|\ell-1} \stackrel{?}{=} \mathsf{ID}^*_{|i-1}$ for some $i \in [k, \ell^*]$.

1. *The case of $\mathsf{ID}_{|\ell-1} = \mathsf{ID}^*_{|i-1}$ for some $i \in [k, \ell^*]$:* If $\mathsf{ID}_{|\ell-1}$ is not revoked on time T (more precisely, for $i \in [1, \ell-1]$ $\mathsf{ID}_{|i}$ is a non-revoked user on time T), then for $i \in [1, \ell-1]$ $\mathcal{B}$ can always choose a node θ on $\mathsf{Path}(\zeta_{\mathsf{ID}_{|i}}) \cap \mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|i-1}}, \mathsf{RL}_{\mathsf{ID}_{|i-1}}, \mathsf{T})$ and let Lv_i be the level of such node θ. $\mathcal{B}$ keeps $(\mathsf{ID}_{|i}, \theta, \mathsf{T})$ in its storage. When $\mathcal{B}$ is required to generate key update regarding an identity $\mathsf{ID}'_{|\ell'}$ on time period T, if $\mathsf{ID}'_{|i} = \mathsf{ID}_{|i}$, then $\mathcal{B}$ chooses the same node θ, and so uses the same Lv_i in key update.
   For $(i,j) \in [1,\ell] \times [1,n]$, randomly choose $S_{(i,j)} \in \mathbb{G}$ and store them in $\mathsf{st}_{\mathsf{ID}_{|\ell}}$ if they are undefined. Otherwise, just recall them. For $\theta \in \mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell}}, \mathsf{RL}_{\mathsf{ID}_{|\ell}}, \mathsf{T})$, randomly choose $S_\theta \in \mathbb{G}$ and store it in the node θ if it is undefined. Otherwise, recall it. Randomly choose $\overrightarrow{s}_\theta \xleftarrow{\$} \mathbb{Z}_p^\ell$ and $s \xleftarrow{\$} \mathbb{Z}_p$, compute $\overrightarrow{f}_{\mathsf{ID}_{|\ell},\theta}$ as

$$\begin{cases} \left((S_\theta \prod_{i=1}^{\ell} S_{(i,\mathsf{Lv}_i)})^{-1} (g_1^{\mathsf{T}-\mathsf{T}^*})^s g_2^{-\frac{c}{(\mathsf{T}-\mathsf{T}^*)}} g^{cs} \overrightarrow{F}(*, \mathsf{ID}_{|\ell})^{(0, \overrightarrow{s}_\theta)},\ g^{-\frac{\beta}{(\mathsf{T}-\mathsf{T}^*)}+s},\ g^{\overrightarrow{s}_\theta},\ 1_{\mathbb{G}} \right), & \text{if } \theta \in \mathsf{Path}(\zeta_{\mathsf{ID}^*_{|i}}) \\ \left((S_\theta \prod_{i=1}^{\ell} S_{(i,\mathsf{Lv}_i)})^{-1} \overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{|\ell})^{(s, \overrightarrow{s}_\theta)},\ g^{(s, \overrightarrow{s}_\theta)},\ 1_{\mathbb{G}} \right), & \text{otherwise.} \end{cases}$$

and return $\mathsf{ku}_{\mathsf{ID}_{|\ell},\mathsf{T}} = \left\{\{\mathsf{Lv}_i\}_{i\in[1,\ell]},\ \overrightarrow{f}_{\mathsf{ID}_{|\ell},\theta} \in \mathbb{G}^{\ell+3}\right\}_{\theta\in\mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell}},\mathsf{RL}_{\mathsf{ID}_{|\ell}},\mathsf{T})}$. When $\mathsf{T} = \mathsf{T}^*$, $\mathsf{ID}^*_{|i}$ should be revoked, so that $\mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell}}, \mathsf{RL}_{\mathsf{ID}_{|\ell}}, \mathsf{T}) \cap \mathsf{Path}(\zeta_{\mathsf{ID}^*_{|i}}) = \emptyset$. Therefore, $\mathcal{B}$ is still able to compute $\overrightarrow{f}_{\mathsf{ID}_{|\ell},\theta}$ as the above.

2. *The case of* $\mathsf{ID}_{|\ell-1} \neq \mathsf{ID}^*_{|i-1}$ *for all* $i \in [k, \ell^*]$*:* $\mathcal{B}$ behaves as the KeyUp oracle in the case of type-0 adversary.

The KeyGen Oracle: Receive $\mathsf{ID}_{|\ell+1}$ from $\mathcal{A}$. For $(i,j) \in [1,\ell] \times [1,n]$, recall $S_{(i,j)}$ from $\mathsf{st}_{\mathsf{ID}_{|\ell}}$ if it is defined. Otherwise, randomly choose $R'_{(i,j)} \xleftarrow{\$} \mathbb{G}$ and store them in $\mathsf{st}_{\mathsf{ID}_{|\ell}}$. For a node $\theta \in \mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell+1}}) \subset \mathsf{BT}_{\mathsf{ID}_{|\ell}}$, recall S_θ if it is defined. Otherwise, randomly choose S_θ from $\mathbb{G}$ and store it in the node $\theta \in \mathsf{BT}_{\mathsf{ID}_{|\ell}}$. Next, $\mathcal{B}$ behaves differently according to whether $\mathsf{ID}_{|\ell+1} = \mathsf{ID}^*_{|m}$ and whether $\mathsf{ID}_{|\ell} = \mathsf{ID}^*_{|m-1}$ for some $m \in [k, \ell^*]$.

1. *The case of* $\mathsf{ID}_{|\ell} = \mathsf{ID}^*_{|m-1}$ *for some* $m \in [k, \ell^*]$*:* $\mathcal{B}$ queries to obtain a private key $(d', d_1, \ldots, d_{\ell+1})$ for $\mathsf{ID}_{|\ell+1}$.
   For $(i,j) \in [1,\ell] \times [1,n]$, $\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell+1}} = \left(S_{(i,j)}\overrightarrow{F}(*, \mathsf{ID}_{|\ell})^{\overrightarrow{r}_{(i,j)}},\ g^{\overrightarrow{r}_{(i,j)}}\right)$, where $\overrightarrow{r}_{(i,j)} \xleftarrow{\$} \{0\} \times \mathbb{Z}_p^{\ell+1}$.
   For $(i,j) \in \{\ell+1\} \times [1,n]$,

$$\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell+1}} = \begin{cases} \left(S_\theta \overrightarrow{F}(*, \mathsf{ID}_{|\ell+1})^{\overrightarrow{r}_{(i,j)}},\ g^{\overrightarrow{r}_{(i,j)}}\right) & \text{if } \theta \in \mathsf{Path}(\zeta_{\mathsf{ID}^*_{|m}}) \subset \mathsf{BT}_{\mathsf{ID}_{|\ell}}, \\ \left(S_\theta d' \overrightarrow{F}(*, \mathsf{ID}_{|\ell+1})^{\overrightarrow{r}_{(i,j)}},\ (1_{\mathbb{G}}, d_1, \ldots, d_{\ell+1}) \circ g^{\overrightarrow{r}_{(i,j)}}\right), & \text{otherwise.} \end{cases}$$

   where $\overrightarrow{r}_{(i,j)} \xleftarrow{\$} \{0\} \times \mathbb{Z}_p^{\ell+1}$ and θ is the j-th level node on $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell+1}}) \subset \mathsf{BT}_{\mathsf{ID}_{|\ell}}$.
2. *The case of* $\mathsf{ID}_{|\ell} \neq \mathsf{ID}^*_{|m-1}$ *for* $\forall m \in [k, \ell^*]$*:* $\mathcal{B}$ behaves as the KeyGen oracle in the case of type-0 adversary.

Challenge Phase: $\mathcal{B}$ behaves same as in the challenge phase in the case of type-0 adversary.

Analysis of $\mathcal{B}$: We argue that if $\mathcal{B}$ correctly guesses the type of adversary, then $\mathcal{B}$'s advantage in IND-sID-CPA game is equal to $\mathcal{A}$'s advantage in IND-sRID-CPA game.

In the simulation, $\mathcal{B}$ uses the same target identity, time period, and messages as those used by $\mathcal{A}$, and it delivers the output bit of $\mathcal{A}$ to $\mathcal{C}$. Therefore, for proof of Lemma, it is sufficient to show the simulated transcript between $\mathcal{A}$ and $\mathcal{B}$ is identical to those in the real experiment.

The public parameter mpk is uniformly distributed since h_0 is uniformly and independently distributed and other parts are given from the public parameter of BB-HIBE. For the challenge ciphertext, B^* can be written of the form g^t, Then, $C_0 = F_0(\mathsf{T}^*)^t = (g_1^{\mathsf{T}^*} h_0)^t = (g_1^{\mathsf{T}^*}(g_1^{-\mathsf{T}^*} g^c))^t = (g^c)^t = B^c$. Therefore, the challenge ciphertext is also well distributed identically to that of real experiment. Next, we consider the distribution of output of $\mathsf{KeyUp}(\cdot)$ and $\mathsf{KeyGen}(\cdot)$ oracles. From Lemma 5.1 and Lemma 5.2, we see the form of each secret key and key update. We show that secret keys and key updates simulated by $\mathcal{B}$ have the form in Lemma 5.1 and Lemma 5.2 according to the types of adversary.

Type-0 adversary: We argue that $S_{(i,j)}$ and $S_\theta g_2^\alpha$ in the output of the KeyGen oracle is equal to $R'_{(i,j)}$ and $R'_{(\ell+1,j)}$ in Lemma 5.1 (when we set $i = \ell + 1$ in the lemma), respectively, where θ is the j-th level node on $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell+1}}) \subset \mathsf{BT}_{\mathsf{ID}_{|\ell}}$. We check that $S_{(i,j)}$ and S_θ satisfy three conditions of $R'_{(i,j)}$ and R'_θ in Lemma 5.1. Since $S_{(i,j)}$ and $S_\theta \xleftarrow{\$} \mathbb{G}$ and $\overrightarrow{r}_{(i,j)} \xleftarrow{\$} \mathbb{Z}_p$, the first condition holds. $S_{(i,j)}$ is stored in $\mathsf{st}_{\mathsf{ID}_{|\ell}}$ and it is used for all children so that the second condition also holds. S_θ is stored in the node on $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell+1}}) \subset \mathsf{BT}_{\mathsf{ID}_{|\ell}}$, and so S_θ is a value associated with the node θ. Therefore, if θ is the j-th level node, then we can consider $S_\theta g_2^\alpha$ as $R'_{(\ell+1,j)}$ such that the third condition of Lemma 5.1 holds.

Next, we show that each key update satisfies all conditions in Lemma 5.2. In the simulation of key update, S_θ and $S_{(i,\mathsf{Lv}_i)}$ are used such that S_θ and $S_{(i,j)}$ are also used in the secret key of $\mathsf{ID}_{|\ell+1}$. From the above simulation of KeyGen, we know that $S_\theta g_2^\alpha = R'_{(\ell+1,j)}$ and $S_{(i,\mathsf{Lv}_i)} = R'_{(i,\mathsf{Lv}_i)}$, and so

$$(S_\theta \prod_{i=1}^{\ell} S_{(i,\mathsf{Lv}_i)})^{-1} = g_2^\alpha (R'_{(\ell+1,j)} \prod_{i=1}^{\ell} R'_{(i,\mathsf{Lv}_i)})^{-1}.$$

Therefore, the distribution of ku is equal to that in Lemma 5.2.

Type-k adversary: We argue that $S_{(i,j)}$ and S_θ in the output distribution of KeyGen are distributed with the following conditions: $S_{(i,j)}$ is equal to $R'_{(i,j)}$ in Lemma 5.1. S_θ is separately distributed according to the following cases.

1. *The case of* $\mathsf{ID}_{|\ell+1} = \mathsf{ID}^*_{|i}$ *for some* $i \in [k, \ell^*]$*:* S_θ is equal to $R'_{(\ell+1,j)}$ in Lemma 5.1 (when we set $i = \ell + 1$ in the lemma), where θ is the j-th level node on $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell+1}}) \subset \mathtt{BT}_{\mathsf{ID}_{|\ell}}$.
2. *The case of* $\mathsf{ID}_{|\ell+1} \neq \mathsf{ID}^*_{|i}$ *but* $\mathsf{ID}_{|\ell} = \mathsf{ID}^*_{|i-1}$ *for some* $i \in [k, \ell^*]$*:* For $\theta \in \mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell+1}}) \subset \mathtt{BT}_{\mathsf{ID}_{|\ell}}$, if $\theta \in \mathsf{Path}(\zeta_{\mathsf{ID}^*_{|i}}) \subset \mathtt{BT}_{\mathsf{ID}_{|\ell}}$, then S_θ is equal to $R'_{(\ell+1,j)}$ in Lemma 5.1 (when we set $i = \ell + 1$ in the lemma), where θ is the j-th level node on $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell+1}}) \subset \mathtt{BT}_{\mathsf{ID}_{|\ell}}$. Otherwise, that is, $\theta \in \mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell+1}}) \setminus \mathsf{Path}(\zeta_{\mathsf{ID}^*_{|i}})$, $S_\theta g_2^\alpha$ is equal to $R'_{(\ell+1,j)}$ in Lemma 5.1.
3. *The case of* $\mathsf{ID}_{|\ell} \neq \mathsf{ID}^*_{|i-1}$ *for* $\forall i \in [k, \ell^*]$*:* $S_\theta g_2^\alpha$ is equal to $R'_{(\ell+1,j)}$ in Lemma 5.1 (when we set $i = \ell + 1$ in the lemma), where θ is the j-th level node on $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell+1}}) \subset \mathtt{BT}_{\mathsf{ID}_{|\ell}}$.

We can easily check the above distribution about $S_{(i,j)}$ and S_θ. Since the proof is similar as that of the type-0 adversary, we omit it.

We can say about the above distribution as follows: Let $\mathsf{ID}_{|\ell}$ be the parent of $\mathsf{ID}_{|\ell+1}$. If for some $i \in [k, \ell^*]$, $\mathsf{ID}_{|\ell} = \mathsf{ID}^*_{|i}$ and θ is on $\mathsf{Path}(\zeta_{\mathsf{ID}^*_{|i}}) \subset \mathtt{BT}_{\mathsf{ID}_{|\ell}}$, then S_θ is equal to $R'_{(\ell+1,j)}$, where j is the level of θ. Otherwise, $S_\theta g_2^\alpha$ is equal to $R'_{(\ell+1,j)}$, where j is the level of θ.

Next, we show that each key update satisfies all conditions in Lemma 5.2. In the simulation of key update, S_θ and $S_{(i,\mathsf{Lv}_i)}$ are used such that S_θ and $S_{(i,j)}$ are also used in the secret key of $\mathsf{ID}_{|\ell+1}$. From the above simulation of KeyGen, we know that $S_{(i,\mathsf{Lv}_i)} = R'_{(i,\mathsf{Lv}_i)}$ and

$$\begin{cases} S_\theta & = R'_{(\ell+1,j)} & \text{if for some } i \in [k, \ell^*],\ \mathsf{ID}_{|\ell} = \mathsf{ID}^*_{|i} \text{ and } \theta \text{ is on } \mathsf{Path}(\zeta_{\mathsf{ID}^*_{|i}}) \subset \mathtt{BT}_{\mathsf{ID}_{|\ell}} \\ S_\theta g_2^\alpha & = R'_{(\ell+1,j)} & \text{otherwise} \end{cases}.$$

If for some $i \in [k, \ell^*]$, $\mathsf{ID}_{|\ell} = \mathsf{ID}^*_{|i}$ and θ is on $\mathsf{Path}(\zeta_{\mathsf{ID}^*_{|i}}) \subset \mathtt{BT}_{\mathsf{ID}_{|\ell}}$, then

$$\begin{aligned}
\overrightarrow{f}_{\mathsf{ID}_{|\ell},\theta} &= \Big((S_\theta \textstyle\prod_{i=1}^{\ell} S_{(i,\mathsf{Lv}_i)})^{-1} (g_1^{\mathsf{T}-\mathsf{T}^*} g^c)^s g_2^{-\frac{c}{(\mathsf{T}-\mathsf{T}^*)}} \overrightarrow{F}(*, \mathsf{ID}_{|\ell})^{\overrightarrow{s}_\theta},\ g^{-\frac{\beta}{(\mathsf{T}-\mathsf{T}^*)}+s},\ g^{\overrightarrow{s}_\theta},\ 1_{\mathbb{G}}\Big) \\
&= \Big(g_2^\alpha (S_\theta \textstyle\prod_{i=1}^{\ell} S_{(i,\mathsf{Lv}_i)})^{-1} (g_1^{\mathsf{T}-\mathsf{T}^*} g^c)^s g_2^{-\alpha-\frac{c}{(\mathsf{T}-\mathsf{T}^*)}} \overrightarrow{F}(*, \mathsf{ID}_{|\ell})^{\overrightarrow{s}_\theta},\ g^{-\frac{\beta}{(\mathsf{T}-\mathsf{T}^*)}+s},\ g^{\overrightarrow{s}_\theta},\ 1_{\mathbb{G}}\Big) \\
&= \Big(g_2^\alpha (S_\theta \textstyle\prod_{i=1}^{\ell} S_{(i,\mathsf{Lv}_i)})^{-1} (g_1^{\mathsf{T}-\mathsf{T}^*} g^c)^s g^{-\beta(\alpha+\frac{c}{(\mathsf{T}-\mathsf{T}^*)})} \overrightarrow{F}(*, \mathsf{ID}_{|\ell})^{\overrightarrow{s}_\theta},\ g^{-\frac{\beta}{(\mathsf{T}-\mathsf{T}^*)}+s},\ g^{\overrightarrow{s}_\theta},\ 1_{\mathbb{G}}\Big) \\
&= \Big(g_2^\alpha (S_\theta \textstyle\prod_{i=1}^{\ell} S_{(i,\mathsf{Lv}_i)})^{-1} (g_1^{\mathsf{T}-\mathsf{T}^*} g^c)^s (g_1^{\mathsf{T}-\mathsf{T}^*} g^c)^{-\frac{\beta}{(\mathsf{T}-\mathsf{T}^*)}} \overrightarrow{F}(*, \mathsf{ID}_{|\ell})^{\overrightarrow{s}_\theta},\ g^{-\frac{\beta}{(\mathsf{T}-\mathsf{T}^*)}+s},\ g^{\overrightarrow{s}_\theta},\ 1_{\mathbb{G}}\Big) \\
&= \Big(g_2^\alpha (S_\theta \textstyle\prod_{i=1}^{\ell} S_{(i,\mathsf{Lv}_i)})^{-1} (g_1^{\mathsf{T}-\mathsf{T}^*} g^c)^{s'} \overrightarrow{F}(*, \mathsf{ID}_{|\ell})^{\overrightarrow{s}_\theta},\ g^{s'},\ g^{\overrightarrow{s}_\theta},\ 1_{\mathbb{G}}\Big) \\
&= \Big(g_2^\alpha (S_\theta \textstyle\prod_{i=1}^{\ell} S_{(i,\mathsf{Lv}_i)})^{-1} \overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{|\ell})^{(s', \overrightarrow{s}_\theta)},\ g^{(s', \overrightarrow{s}_\theta)},\ 1_{\mathbb{G}}\Big)
\end{aligned}$$

where $s' = -\frac{\beta}{(\mathsf{T}-\mathsf{T}^*)} + s$ and $\beta = \log_g g_2$. Since s is uniformly chosen, s' is also uniformly distributed. From the above simulation of the KeyGen oracle, we know that $S_\theta = R'_{(\ell+1,j)}$ and $S_{(i,\mathsf{Lv}_i)} = R'_{(i,\mathsf{Lv}_i)}$. Therefore, $\overrightarrow{f}_{\mathsf{ID}_{|\ell},\theta}$ has the desired form

$$\Big(g_2^\alpha (R'_{(\ell+1,j)} \textstyle\prod_{i=1}^{\ell} R'_{(i,\mathsf{Lv}_i)})^{-1} \overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{|\ell})^{(s', \overrightarrow{s}_\theta)},\ g^{(s', \overrightarrow{s}_\theta)},\ 1_{\mathbb{G}}\Big),$$

where j is the level of θ.

In the other case (for some $i \in [k, \ell^*]$, $\mathsf{ID}_{|\ell} = \mathsf{ID}^*_{|i}$ and θ is not on $\mathsf{Path}(\zeta_{\mathsf{ID}^*_{|i}}) \subset \mathtt{BT}_{\mathsf{ID}_{|\ell}}$, or $\mathsf{ID}_{|\ell} \neq \mathsf{ID}^*_{|i}$ for all $i \in [k, \ell^*]$),

$$\overrightarrow{f}_{\mathsf{ID}_{|\ell},\theta} = \quad \Big((S_\theta \textstyle\prod_{i=1}^{\ell} S_{(i,\mathsf{Lv}_i)})^{-1} \overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{|\ell})^{(s, \overrightarrow{s}_\theta)},\ g^{(s, \overrightarrow{s}_\theta)},\ 1_{\mathbb{G}}\Big)$$

Since $S_\theta = R'_{(\ell+1,j)}$ and $S_{(i,\mathsf{Lv}_i)} g_2^\alpha = R'_{(i,\mathsf{Lv}_i)}$,

$$\overrightarrow{f}_{\mathsf{ID}_{|\ell},\theta} = \Big(g_2^\alpha (R'_{(\ell+1,j)} \prod_{i=1}^{\ell} R'_{(i,\mathsf{Lv}_i)})^{-1} \overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{|\ell})^{(s, \overrightarrow{s}_\theta)},\ g^{(s, \overrightarrow{s}_\theta)},\ 1_{\mathbb{G}}\Big),$$

where j is the level of θ. Therefore, we complete the proof of theorem.

□

From the theorem 2.1 and theorem 5.1, we obtain the following corollary.

Corollary 5.1. *If the DBDH assumption holds in the bilinear group $(p, \mathbb{G}, \mathbb{G}_t, e)$, then the proposed RHIBE scheme over $(p, \mathbb{G}, \mathbb{G}_t, e)$ is IND-sRID-CPA secure.*

6 Summary and Open problems

We proposed the first construction for efficient delegation of both key generating functionality and revocation functionality in the IBE system.

There are interesting open problems. Our construction is based on the BB-HIBE scheme and we proved only selective-security of our construction. Natural open problem is to construct RHIBE scheme based on more efficient (in the sense of the ciphertext size) and secure (in the sense of satisfying adaptive-security) HIBE scheme (e.g.,[15]). Another open problem is to combine HIBE scheme with so-called Subset Difference (SD) method [20] (instead of CS). It seems not easy to combine SD with (H)IBE scheme since the SD method requires more complicated key distributing method than CS method.

Acknolwdgement We thank anonymous reviewers of CT-RSA 2013 and members of Shin-Akarui-Angou-Benkyou-Kai for their helpful comments.

References

[1] M. Bellare and A. Palacio. Protecting against key exposure: strongly key-insulated encryption with optimal threshold. In *IACR Cryptology ePrint Archive 2002:064*, 2002.

[2] A. Boldyreva, V. Goyal, and V. Kumar. Identity-based encryption with efficient revocation. In *ACM CCS 2008*, pages 417–426, 2008.

[3] D. Boneh and X. Boyen. Efficient selective-id identity based encryption without random oracles. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer-Verlag, 2004.

[4] D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertexts. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer-Verlag, 2005.

[5] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–23. Springer-Verlag, 2001.

[6] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO 2006*, volume 4117 of *LNCS*, pages 290–307. Springer-Verlag, 2006.

[7] J. Chen, H. W. Lim, S. Ling, H. Wang, and K. Nguyen. Revocable identity-based encryption from lattices. In *ACISP 2012*, volume 7372, pages 390–403, 2012.

[8] Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-insulated public key cryptosystems. In *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 65–82. Springer, 2002.

[9] C. Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT 2006*, volume 4004 of *LNCS*. Springer-Verlag, 2006.

[10] C. Gentry and S. Halevi. Hierarchical identity base encryption with plynomially many levels. In *TCC 2009*, volume 5444 of *LNCS*, pages 437–456. Springer-Verlag, 2009.

[11] C. Gentry and A. Silverberg. Hierarchical id-based cryptography. In *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 149–155. Springer-Verlag, 2002.

[12] G. Hanaoka and J. Weng. Generic constructions of parallel key-insulated encryption. In *SCN*, volume 6280 of *LNCS*, pages 36–53. Springer, 2010.

[13] Y. Hanaoka, G. Hanaoka, J. Shikata, and H. Imai. Identity-based hierarchical strongly key-insulated encryption and its application. In *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 495–514. Springer, 2005.

[14] J. Horwitz and B. Lynn. Towards hierarchical identity-based encryption. In *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 466–481. Springer-Verlag, 2002.

[15] A. B. Lewko and B. Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *TCC 2010*, volume 5978 of *LNCS*, pages 455–479. Springer, 2010.

[16] A. B. Lewko and B. Waters. Unbounded HIBE and attribute-based encryption. In *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 547–567. Springer, 2011.

[17] B. Libert, J.-J. Quisquater, and M. Yung. Parallel key-insulated public key encryption without random oracles. In *PKC 2007*, volume 4450 of *LNCS*, pages 298–314. Springer, 2007.

[18] B. Libert and D. Vergnaud. Adaptive-ID secure revocable identity-based encryption. In *CT-RSA 2009*, volume 5473 of *LNCS*, pages 1–15. Springer, 2009.

[19] B. Libert and D. Vergnaud. Towards black-box accountable authority IBE with short ciphertexts and private keys. In *PKC 2009*, volume 5443 of *LNCS*, pages 235–255. Springer, 2009.

[20] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 41–62. Springer, 2001.

[21] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, 2005.

[22] J. H. Seo and K. Emura. Efficient delegation of key generation and revocation functionalities in identity-based encryption. In *CT-RSA 2013*, volume 7779 of *LNCS*, pages 343–358. Springer, 2013.

[23] J. H. Seo, T. Kobayashi, M. Ohkubo, and K. Suzuki. Anonymous hierarchical identity-based encryption with constant size ciphertexts. In G. Tsudik and S. Jarecki, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 215–234. Springer-Verlag, 2009.

[24] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO 1984*, volume 196 of *LNCS*, pages 47–53. Springer, 1984.

[25] B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer-Verlag, 2009.

[26] J. Weng, S. Liu, K. Chen, D. Zheng, and W. Qiu. Identity-based threshold key-insulated encryption without random oracles. In *CT-RSA 2008*, volume 4964 of *LNCS*, pages 203–220. Springer, 2008.