Revocable Hierarchical Identity-Based Encryption with Adaptive Security

Kwangsu Lee[*]

Abstract

Hierarchical identity-based encryption (HIBE) can be extended to revocable HIBE (RHIBE) if a private key of a user can be revoked when the private key is revealed or expired. Previously, many selectively secure RHIBE schemes were proposed, but it is still unsolved problem to construct an adaptively secure RHIBE scheme. In this work, we propose two RHIBE schemes in composite-order bilinear groups and prove their adaptive security under simple static assumptions. To prove the adaptive security, we use the dual system encryption framework, but it is not simple to use the dual system encryption framework in RHIBE since the security model of RHIBE is quite different with that of HIBE. We show that it is possible to solve the problem of the RHIBE security proof by carefully designing hybrid games.

Keywords: Hierarchical identity-based encryption, Key revocation, Adaptive security, Dual system encryption, Bilinear maps.

[*]Korea University, Seoul, Korea. Email: `kwangsu.lee@korea.ac.kr`.

1 Introduction

Hierarchical identity-based encryption (HIBE) is an important extension of identity-based encryption (IBE) [6] that uses the identity of a user as the public-key of the user. In HIBE, the identity of a user is represented as a hierarchical structure and a user with a private key can delegate his private key to next-level users. The concept of HIBE was introduced by Horwitz and Lynn [12] to reduce the burden of a private key generation in a trusted center and a secure HIBE scheme that supports arbitrary many levels are proposed by Gentry and Silverberg [10]. HIBE can be extended to broadcast encryption, forward-secure encryption, chosen-ciphertext secure encryption, and searchable encryption [1, 7–9] and it has many interesting applications like encryption systems for medical data and range query on encrypted data [2, 27].

To use an HIBE scheme in real applications, we should revoke the private key of a user if his private key is revealed or his credential is expired. Revocable HIBE (RHIBE) is an extension of HIBE that supports the revocation functionality by broadcasting an update key for non-revoked users per each time period. Previously, an efficient revocable IBE (RIBE) schemes were proposed by many researchers [3,15,19,21,24]. Seo and Emura [23] proposed the first RHIBE scheme by following the design strategy of Boldyreva et al. [3] that uses a binary tree and proved its selective security. After that, some efficient RHIBE schemes with improved parameters were proposed [17, 26], but these are also proven to be selectively secure.

The right security model of RHIBE is the adaptive model where an adversary can select a target in the challenge step. In RIBE, adaptively secure RIBE schemes were already proposed in [15, 19, 24]. However, all RHIBE schemes only provide the selective security where the challenge identity ID^* and the challenge time T^* should be submitted before receiving public parameters or the selective revocation list security where the challenge revocation set R^* should be additionally submitted [17, 23, 26]. Although an RHIBE scheme claimed to be adaptively secure was proposed in [25], the security proof that uses the dual system encryption technique has some flaws. Therefore, the construction of an adaptively secure RHIBE scheme is an unsolved open problem.

1.1 Our Results

In this paper, we give an answer to this unsolved problem by proposing two RHIBE schemes in composite-order bilinear groups and proving their adaptive security under simple static assumptions.

We first propose an RHIBE-CS scheme by combining the HIBE and IBE schemes of Lewko and Waters [18] and the complete subtree (CS) scheme of Naor, Naor, and Lotspiech [20] in a modular way. For the construction of our RHIBE-CS scheme, we follow the modular design approach of Lee and Park [17] except that the underlying HIBE and IBE schemes are replaced by the schemes of Lewko and Waters. We then prove the adaptive security of our RHIBE-CS scheme by using the dual system encryption framework [18, 28]. However, the naive approach of dual system encryption does not work for RHIBE since an adversary can query a private key for ID that is a prefix of ID^* and an update key for T^* where ID^* is the challenge identity and T^* is the challenge time, and these private key and update key cannot be easily converted from normal to semi-functional. Thus, solving this problem of RHIBE when the dual system encryption was used is the core of the security proof. The main technical idea of solving this problem is described in the later part of this section.

Next, we propose an RHIBE-SD scheme by using the subset difference (SD) scheme instead of using the CS scheme to reduce the size of an update key. As mentioned before, we also follow the modular design approach of Lee and Park [17]. Our RHIBE-SD scheme has $O(r)$ number of group elements in an update key and $O(\log^2 N_{max})$ number of group elements in a private key whereas our RHIBE-CS scheme has $O(r \log(N_{max}/r))$ number of group elements in an update key and $O(\log N_{max})$ number of group elements

Table 1: Comparison of revocable hierarchical identity-based encryption schemes

Scheme	PP Size	SK Size	UK Size	CT Size	Model	Assumption
SE (CS) [23]	$O(\ell)$	$O(\ell^2 \log N)$	$O(\ell r \log \frac{N}{r})$	$O(\ell)$	SE-IND	DBDH
SE (CS) [26]	$O(\ell)$	$O(\ell \log N)$	$O(\ell r \log \frac{N}{r})$	$O(1)$	SE-IND	q-Type
SE (SD) [26]	$O(\ell)$	$O(\ell \log^2 N)$	$O(\ell r)$	$O(1)$	SRL-IND	q-Type
LP (CS) [17]	$O(1)$	$O(\log N)$	$O(\ell + r \log \frac{N}{r})$	$O(\ell)$	SE-IND	q-Type
LP (SD) [17]	$O(1)$	$O(\log^2 N)$	$O(\ell + r)$	$O(\ell)$	SRL-IND	q-Type
Ours (CS)	$O(\ell)$	$O(\ell \log N)$	$O(\ell + r \log \frac{N}{r})$	$O(1)$	AD-IND	Static
Ours (SD)	$O(\ell)$	$O(\ell \log^2 N)$	$O(\ell + r)$	$O(1)$	AD-IND	Static

We let N be the number of maximum users in each level, r be the number of revoked users, and ℓ be the depth of a hierarchical identity. We count the number of group elements to measure the size of parameters. We use symbols SE-IND for selective IND-CPA, SRL-IND for selective revocation list IND-CPA, and AD-IND for adaptive IND-CPA.

in a private key. The detailed comparison of RHIBE schemes is given in Table 1. To prove the adaptive security of our RHIBE-SD scheme, we carefully use the proof technique of Lee et al. [15] that was used to prove the adaptive security of their RIBE scheme.

1.2 Our Techniques

To prove the adaptive security of an HIBE scheme, the dual system encryption framework was introduced by Waters [28]. In the dual system encryption framework, ciphertexts and private keys can be normal or semi-functional in which a normal ciphertext can be decrypted by a normal or semi-functional private key whereas a semi-functional ciphertext cannot be decrypted by a semi-functional private key. To prove the adaptive security, a normal challenge ciphertext is changed to be semi-functional, and then each normal private key is changed to be semi-functional one by one through hybrid games. The main obstacle of this proof is to overcome the paradox of dual system encryption in which a simulator can check whether a private key is normal or semi-functional by decrypting a semi-functional ciphertext since a simulator can generate a ciphertext and a private key for any identity. Lewko and Waters [18] solved this problem by introducing the nominally semi-functional type of private keys where a semi-functional ciphertext can be decrypted by a nominally semi-functional private key. Note that an information theoretic argument should be given to argue that a nominally semi-functional key is indistinguishable from a semi-functional key.

For the security proof of an RHIBE scheme, one may simply use the dual system encryption technique that changes private keys and update keys from normal types to semi-functional types one by one through hybrid games. However, this simple strategy does not work since the adversary of RHIBE can query a private key for ID that is a prefix of ID^* and an update key for $T = T^*$ where ID^* and T^* are the challenge identity and time. That is, we cannot show the information theoretic argument for these private key and update key since ID is a prefix of ID^* and $T = T^*$. In HIBE, the restriction of an adversary that ID is not a prefix of ID^* is essentially used to show the information theoretic argument. Thus, it is not easy to prove the adaptive security of an RHIBE scheme by using the dual system encryption framework.

Our strategy to overcome this problem is that private keys and update keys of an RHIBE scheme are first divided into smaller component keys and then these component keys that are related to the same node in a binary tree are grouped together. Next, these component keys that belong to the same group are changed from normal types to semi-functional types one by one through hybrid games. Similar proof strategy was used in [14, 15, 22]. In particular we consider an RHIBE-CS scheme that use the CS scheme. A private key consists of many HIBE private keys that are related to a path in a binary tree and an update key also consists of many IBE private keys that are related to a cover set in a binary tree. By the grouping of HIBE private keys and IBE private keys with the same node, we can use the restriction of the RHIBE security model to show an information theoretic argument.

For example, if an adversary requests a private key for $ID \in \mathbf{Prefix}(ID^*)$ and one HIBE private key of this private key is related to a node v^*, then all IBE private keys in update keys satisfy $T \neq T^*$ for this node v^* since this private key should be revoked on time T^* by the restriction of the security model. Thus, we first change IBE private key related to v^* from normal to semi-functional one by one by using $T \neq T^*$, and then we change HIBE private keys related to v^* from normal to semi-functional at once. Note that there is no paradox of dual system encryption when we change HIBE private keys from normal to semi-functional since IBE private keys are already semi-functional. Recall that an information theoretic argument is not needed if nominally semi-functional keys are not used. Similar argument also applies when the adversary requests an update key for $T = T^*$ and one IBE private key of this update key is related to a node v^* since we have $ID \notin \mathbf{Prefix}(ID^*)$ for all HIBE private key for this node v^*.

To prove the adaptive security of our RHIBE-SD scheme, we also use the similar proof strategy that private keys and update keys are divided into smaller component keys and these component keys that belong to the same group are changed from normal to semi-functional. In our RHIBE-CS scheme, a group is simply defined by a node v_j in a binary tree. In our RHIBE-SD scheme, a group is defined as a set of subsets $S_{i,j}$ such that v_i is the same and the depth d_j of v_j is the same where $S_{i,j}$ is defined by two nodes v_i and v_j in a binary tree. To change HIBE private keys and IBE private keys in the same group from normal to semi-functional, we carefully design hybrid games since a group is very complex. Note that Lee et al. [15] also used this proof strategy to prove the adaptive security of their RIBE-SD scheme.

1.3 Related Work

An IBE scheme with key revocation was first proposed by Boneh and Franklin [6] in which each user should retrieve his private key from a trusted center for the identity $ID \| T$ per each time period T. Boldyreva, Goyal, and Kumar [3] proposed a scalable RIBE scheme by combining a fuzzy IBE scheme and the CS scheme in which an update key is broadcasted to non-revoked users per each time period. This design method that uses the CS scheme for key revocation was also used to build other adaptively secure RIBE schemes [19,24]. The SD scheme is an improvement on the CS scheme since the size of a broadcasting set can be reduced [20]. Lee et al. [15] proposed an RIBE scheme that uses the SD scheme to improve the size of an update key and proved its adaptive security under static assumptions. An RIBE scheme based on a binary tree cannot have short private keys and update keys. To overcome this problem, Park et al. [21] proposed an RIBE scheme with short private keys and update keys by using multilinear maps.

As mentioned before, the first selectively secure RHIBE scheme was proposed by Seo and Emura [23] by combining the HIBE scheme of Boneh and Boyen [4] and the CS scheme. This RHIBE scheme is relatively inefficient since a user should retrieve all update keys generated by his ancestors to decrypt a ciphertext. To solve this problem of inefficiency, Seo and Emura [26] proposed another selectively secure RHIBE scheme with history-free updates that uses the CS (or SD) scheme where a user only needs to retrieve an update key generated by his parent. Recently, Lee and Park [17] proposed new RHIBE schemes with shorter private keys and update keys by combining a new HIBE scheme that has short intermediate private keys and the CS (or SD) scheme in a modular way.

An attribute-based encryption (ABE) scheme also can be extended to support the key revocation. A revocable ABE (RABE) scheme was also proposed by Boldyreva et al. [3] by combining a key-policy ABE scheme and the CS scheme and its selective revocation list security was claimed. To securely protect information stored in cloud storage, one may use an RABE scheme since it provides the access control on encrypted data as well as the key revocation. Sahai et al. [22] pointed out that RABE is not enough for cloud storage and then they proposed a revocable-storage ABE (RS-ABE) scheme that supports the key revocation and the ciphertext update. After that, Lee et al. showed that an RS-ABE scheme can be improved by using a self-updatable encryption (SUE) scheme [13, 14]. Recently, Lee et al. [16] proposed a weaker CCA-secure RS-ABE scheme and proved its selective security.

2 Preliminaries

In this section, we introduce composite-order bilinear groups and complexity assumptions. Next, we define the syntax and the adaptive security model of RHIBE.

2.1 Notation

Let $\mathcal{I}$ be the identity space. A hierarchical identity ID with a depth k is defined as an identity vector $ID = (I_1, \ldots, I_k) \in \mathcal{I}^k$. We let $ID|_j$ be a vector $(I_1, \ldots, I_j)$ of size j derived from ID. If $ID = (I_1, \ldots, I_k)$, then we have $ID = ID|_k$. We define $ID|_0 = \varepsilon$ for simplicity. The function $\mathbf{Prefix}(ID|_k)$ returns a set of prefix vectors $\{ID|_j\}$ for all $1 \leq j \leq k$ where $ID|_k = (I_1, \ldots, I_k) \in \mathcal{I}^k$ for some k. For two hierarchical identities $ID|_i$ and $ID|_j$ with $i < j$, $ID|_i$ is an ancestor of $ID|_j$ and $ID|_j$ is a descendant of $ID|_i$ if $ID|_i \in \mathbf{Prefix}(ID|_j)$.

2.2 Bilinear Groups of Composite Order

Let $N = p_1 p_2 p_3$ where p_1, p_2, and p_3 are distinct prime numbers. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of same composite order N and g be a generator of $\mathbb{G}$. The bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ has the following properties:

1. Bilinearity: $\forall u, v \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}_N$, $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $\exists g$ such that $e(g, g)$ has order N, that is, $e(g, g)$ is a generator of $\mathbb{G}_T$.

We say that $\mathbb{G}$ is a bilinear group if the group operations in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the bilinear map e are all efficiently computable. Furthermore, we assume that the description of $\mathbb{G}$ and $\mathbb{G}_T$ includes generators of $\mathbb{G}$ and $\mathbb{G}_T$ respectively. We use the notation $\mathbb{G}_{p_i}$ to denote the subgroups of order p_i of $\mathbb{G}$ respectively. Similarly, we use the notation $\mathbb{G}_{T,p_i}$ to denote the subgroups of order p_i of $\mathbb{G}_T$ respectively.

2.3 Complexity Assumptions

Assumption 1 (Subgroup Decision, SD)**.** Let $(N, \mathbb{G}, \mathbb{G}_T, e)$ be a description of the bilinear group of composite order $N = p_1 p_2 p_3$. Let g_1, g_2, g_3 be generators of subgroups $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}, \mathbb{G}_{p_3}$ respectively. The SD assumption is that if the challenge tuple

$$D = ((N, \mathbb{G}, \mathbb{G}_T, e), g_1, g_3) \text{ and } Z$$

are given, no PPT algorithm $\mathcal{A}$ can distinguish $Z = Z_0 = X_1 \in \mathbb{G}_{p_1}$ from $Z = Z_1 = X_1 R_1 \in \mathbb{G}_{p_1 p_2}$ with more than a negligible advantage. The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}_{\mathcal{A}}^{SD}(1^\lambda) = \big|\Pr[\mathcal{A}(D, Z_0) = 0] - \Pr[\mathcal{A}(D, Z_1) = 0]\big|$ where the probability is taken over random choices of $X_1 \in \mathbb{G}_{p_1}$ and $R_1 \in \mathbb{G}_{p_2}$.

Assumption 2 (General Subgroup Decision, GSD)**.** Let $(N, \mathbb{G}, \mathbb{G}_T, e)$ be a description of the bilinear group of composite order $N = p_1 p_2 p_3$. Let g_1, g_2, g_3 be generators of subgroups $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}, \mathbb{G}_{p_3}$ respectively. The GSD assumption is that if the challenge tuple

$$D = ((N, \mathbb{G}, \mathbb{G}_T, e), g_1, g_3, X_1 R_1, R_2 Y_1) \text{ and } Z$$

are given, no PPT algorithm $\mathcal{A}$ can distinguish $Z = Z_0 = X_2 Y_2 \in \mathbb{G}_{p_1 p_3}$ from $Z = Z_1 = X_2 R_3 Y_2 \in \mathbb{G}_{p_1 p_2 p_3}$ with more than a negligible advantage. The advantage of $\mathcal{B}$ is defined as $\mathbf{Adv}_{\mathcal{A}}^{GSD}(1^{\lambda}) = \big|\Pr[\mathcal{A}(D, T_0) = 0] - \Pr[\mathcal{A}(D, T_1) = 0]\big|$ where the probability is taken over random choices of $X_1, X_2 \in \mathbb{G}_{p_1}$, $R_1, R_2, R_3 \in \mathbb{G}_{p_2}$, and $Y_1, Y_2 \in \mathbb{G}_{p_3}$.

Assumption 3 (Composite Diffie-Hellman, ComDH)**.** Let $(N, \mathbb{G}, \mathbb{G}_T, e)$ be a description of the bilinear group of composite order $N = p_1 p_2 p_3$. Let g_1, g_2, g_3 be generators of subgroups $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}, \mathbb{G}_{p_3}$ respectively. The ComDH assumption is that if the challenge tuple

$$D = ((N, \mathbb{G}, \mathbb{G}_T, e), g_1, g_2, g_3, g_1^a R_1, g_1^b R_2) \text{ and } Z$$

are given, no PPT algorithm $\mathcal{A}$ can distinguish $Z = Z_0 = e(g_1, g_1)^{ab}$ from $Z = Z_1 = e(g_1, g_1)^c$ with more than a negligible advantage. The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}_{\mathcal{A}}^{ComDH}(1^{\lambda}) = \big|\Pr[\mathcal{A}(D, Z_0) = 0] - \Pr[\mathcal{A}(D, Z_1) = 0]\big|$ where the probability is taken over random choices of $a, b, c \in \mathbb{Z}_N$, and $R_1, R_2 \in \mathbb{G}_{p_2}$.

2.4 Pseudo-Random Functions

A pseudo-random function (PRF) [11] is an efficiently computable function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ where $\mathcal{K}$ is the key space, $\mathcal{X}$ is the domain, and $\mathcal{Y}$ is the range. Let $F(k, \cdot)$ be an oracle for a uniformly chosen $k \in \mathcal{K}$ and $f(\cdot)$ be an oracle for a uniformly chosen function $f : \mathcal{X} \to \mathcal{Y}$. We say that a PRF is secure if for all efficient adversaries $\mathcal{A}$ the advantage $\mathbf{Adv}_{\mathcal{A}}^{PRF}(1^{\lambda}) = \big|\Pr[\mathcal{A}^{F(k,\cdot)} = 1] - \Pr[\mathcal{A}^{f(\cdot)} = 1]\big|$ is negligible.

2.5 Revocable HIBE

RHIBE is an extension of HIBE and it provides the revocation functionality in which each user can revoke child users if the private key of a child user is revealed. In RHIBE, each user additionally provides an update key UK per each time period and a child user can derive a (short-term) decryption key DK to decrypt a ciphertext by combining his (long-term) private key SK and the update key UK if he is not revoked in the update key. The syntax of RHIBE with history-free updates [26] is defined as follows:

Definition 2.1 (Revocable HIBE)**.** An RHIBE scheme with history-free updates for the identity space $\mathcal{I}$, the time space $\mathcal{T}$, and the message space $\mathcal{M}$, consists of seven algorithms **Setup**, **GenKey**, **UpdateKey**, **DeriveKey**, **Encrypt**, **Decrypt**, and **Revoke**, which are defined as follows:

Setup$(1^{\lambda}, L, N_{max})$: This algorithm takes as input a security parameter 1^{λ} and the maximum number N_{max} of users in each level. It outputs a master key MK, an (empty) revocation list RL_{ε}, a state ST_{ε}, and public parameters PP.

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

UpdateKey$(T, RL_{ID|_{k-1}}, DK_{ID|_{k-1},T}, ST_{ID|_{k-1}}, PP)$: This algorithm takes as input time $T \in \mathcal{T}$, a revocation list $RL_{ID|_{k-1}}$, a decryption key $DK_{ID|_{k-1},T}$, and public parameters PP. It outputs an update key $UK_{ID|_{k-1},T,R}$ where R is the set of revoked identities on time T derived from $RL_{ID|_{k-1}}$.

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

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

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

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

The correctness of RHIBE is defined as follows: For all MK and PP generated by **Setup**$(1^\lambda, L, N_{max})$, $SK_{ID|_k}$ generated by **GenKey**$(ID|_k, ST|_{k-1}, PP)$, $UK_{ID|_{k-1},T,R}$ generated by **UpdateKey**$(T, RL_{ID|_{k-1}}, DK_{ID|_{k-1},T}, ST_{ID|_{k-1}}, PP)$, $CT_{ID|_\ell,T}$ generated by **Encrypt**$(ID|_\ell, T, M, PP)$, it is required that

- If $ID|_k \notin R$, then **DeriveKey**$(SK_{ID|_k}, UK_{ID|_{k-1},T,R}, PP) = DK_{ID|_k,T}$.
- If $ID|_k \in R$, then **DeriveKey**$(SK_{ID|_k}, UK_{ID|_{k-1},T,R}, PP) = \perp$.
- If $(ID'|_k \in$ **Prefix**$(ID|_\ell)) \wedge (T = T')$, then **Decrypt**$(CT_{ID|_\ell,T}, DK_{ID'|_k,T'}, PP) = M$.
- If $(ID'|_k \notin$ **Prefix**$(ID|_\ell)) \vee (T \neq T')$, then **Decrypt**$(CT_{ID|_\ell,T}, DK_{ID'|_k,T'}, PP) = \perp$.

The adaptive security model of RHIBE can be defined by extending the adaptive security model of RIBE. We use the adaptive model of RHIBE by extending the selective model of Seo and Emura [26]. In the adaptive security model of RHIBE, an adversary can adaptively request a private key query for any ID and an update key query for time T. In the challenge step, the adversary selects the challenge identity ID^* and challenge time T^*, and two challenge messages M_0^*, M_1^* with some restrictions. After receiving the challenge ciphertext, the adversary guesses the encrypted message in the challenge ciphertext. The formal security definition of RHIBE is given as follows:

Definition 2.2 (Adaptive IND-CPA Security)**.** The adaptive IND-CPA security (AD-IND-CPA) of RHIBE is defined in terms of the following experiment between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$:

1. **Setup**: $\mathcal{C}$ obtains a master key MK, a revocation list RL_ε, a state ST_ε, and public parameters PP by running **Setup**$(1^\lambda, L, N_{max})$. It keeps $MK, RL_\varepsilon, ST_\varepsilon$ to itself and gives PP to $\mathcal{A}$.
2. **Phase 1**: $\mathcal{A}$ adaptively requests a polynomial number of queries. These queries are processed as follows:
   - **Private key**. If it is a private key query for a hierarchical identity $ID|_k$, then $\mathcal{C}$ gives a private key $SK_{ID|_k}$ and a state $ST_{ID|_k}$ by running **GenKey**$(ID|_k, ST_{ID|_{k-1}}, PP)$.
   - **Update key**. If it is an update key query for a hierarchical identity $ID|_{k-1}$ and time T, then $\mathcal{C}$ gives an update key $UK_{ID|_{k-1},T,R}$ by running **UpdateKey**$(T, RL_{ID|_{k-1}}, DK_{ID|_{k-1},T}, ST_{ID|_{k-1}}, PP)$ with the restriction: If $ID|_{k-1}$ or one of its ancestors is revoked on time T, then this update key query cannot be requested since $DK_{ID|_{k-1},T}$ cannot be derived.
   - **Decryption key**. If it is a decryption key query for a hierarchical identity $ID|_k$ and time T, then $\mathcal{C}$ gives a decryption key $DK_{ID|_k,T}$ by running **DeriveKey**$(SK_{ID|_k}, UK_{ID|_{k-1},T,R}, PP)$.

- **Revocation**. If it is a revocation query for a hierarchical identity $ID|_k$ and time T, then $\mathcal{C}$ updates a revocation list $RL_{ID|_{k-1}}$ by running **Revoke**$(ID|_k, T, RL_{ID|_{k-1}}, ST_{ID|_{k-1}})$ with the restriction: A revocation query for $ID|_k$ on time T cannot be requested if an update key query for $ID|_k$ on the time T was requested.

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

3. **Challenge**: $\mathcal{A}$ submits a challenge hierarchical identity $ID^*|_\ell = (I_1^*, \dots, I_\ell^*)$, challenge time T^*, and two challenge messages M_0^*, M_1^* with the following restrictions:
   - If a private key query for $ID|_k \in$ **Prefix**$(ID^*|_\ell)$ where $k \leq \ell$ was requested, then $ID|_k$ or one of its ancestors must be revoked at some time $T \leq T^*$.
   - A decryption key query for the challenge hierarchical identity $ID^*|_k$ or its ancestors on the challenge time T^* was not requested.

   $\mathcal{C}$ flips a random coin $\mu \in \{0,1\}$ and gives the challenge ciphertext $CT^*_{ID^*|_\ell, T^*}$ to $\mathcal{A}$ by running **Encrypt**$(ID^*|_\ell, T^*, M^*_\mu, PP)$.

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

5. **Guess**: Finally, $\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$, and wins the game if $\mu = \mu'$.

The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}^{AD\text{-}IND\text{-}CPA}_{RHIBE,\mathcal{A}}(1^\lambda) = \big|\Pr[\mu = \mu'] - \frac{1}{2}\big|$ where the probability is taken over all the randomness of the experiment. An RHIBE scheme is AD-IND-CPA secure if for all probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the above experiment is negligible in the security parameter λ.

3 Hierarchical IBE

In this section, we describe an HIBE scheme and an IBE scheme that are used as the building blocks of our RHIBE schemes.

3.1 HIBE Scheme

For the underlying HIBE scheme of our RHIBE schemes, we use a key encapsulation mechanism (KEM) version of the HIBE scheme of Lewko and Waters (LW-HIBE) [18]. The LW-HIBE scheme is very similar to the HIBE scheme of Boneh et al. (BBG-HIBE) [5] that has short ciphertexts except that it uses composite-order groups. To build our RHIBE schemes in a modular way, we additionally define some useful algorithms that are introduced by Lee and Park [17].

HIBE.Setup(GDS, L): Let $GDS = ((N, \mathbb{G}, \mathbb{G}_T, e), g_1, g_3)$ be the description of a bilinear group with generators $g_1 \in \mathbb{G}_{p_1}, g_3 \in \mathbb{G}_{p_3}$. It selects random elements $h, u_1, \dots, u_L \in \mathbb{G}_{p_1}$ and a random exponent $\gamma \in \mathbb{Z}_N$. It outputs a master key $MK = \gamma$ and public parameters $PP = \big((N, \mathbb{G}, \mathbb{G}_T, e), g = g_1, Y = g_3, h, u_1, \dots, u_L, \Lambda = e(g,g)^\gamma\big)$.

HIBE.GenKey($ID|_k, MK, PP$)**:** Let $ID|_k = (I_1, \ldots, I_k) \in \mathcal{I}^k$ and $MK = \gamma$. It chooses random $r \in \mathbb{Z}_N, Y_0, Y_1, \{Y_{2,i}\}_{i=k+1}^{L} \in \mathbb{G}_{p_3}$ and outputs a private key $SK_{ID|_k} = \big(K_0 = g^{\gamma}(h\prod_{i=1}^{k} u_i^{I_i})^r Y_0, K_1 = g^{-r} Y_1, \big\{K_{2,i} = u_i^r Y_{2,i}\big\}_{i=k+1}^{L}\big)$.

HIBE.RandKey($SK_{ID|_k}, PP$)**:** Let $SK_{ID|_k} = (K'_0, K'_1, \{K'_{2,i}\}_{i=k+1}^{L})$. It chooses random $r' \in \mathbb{Z}_N, Y'_0, Y'_1, \{Y'_{2,i}\}_{i=k+1}^{L} \in \mathbb{G}_{p_3}$ and outputs a randomized private key $SK_{ID|_k} = \big(K_0 = K'_0 \cdot (h\prod_{i=1}^{k} u^{I_i})^{r'} Y'_0, K_1 = K'_1 \cdot g^{-r'} Y'_1, \big\{K_{2,i} = K'_{2,i} \cdot u^{r'} Y'_{2,i}\big\}_{i=k+1}^{L}\big)$.

HIBE.Delegate($ID|_k, SK_{ID|_{k-1}}, PP$)**:** Let $ID|_k = (I_1, \ldots, I_k) \in \mathcal{I}^k$ and $SK_{ID|_{k-1}} = \big(K'_0, K'_1, \{K'_{2,i}\}_{i=k}^{L}\big)$ where $ID|_{k-1}$ is a prefix of $ID|_k$. It creates a temporal private key $TSK = \big(K_0 = K'_0 \cdot (K'_{2,k})^{I_k}, K_1 = K'_1, \big\{K_{2,i} = K'_{2,i}\big\}_{i=k+1}^{L}\big)$. Next, it outputs a delegated private key $SK_{ID|_k}$ by running **HIBE.RandKey**(TSK, PP).

HIBE.Encaps($ID|_\ell, t, PP$)**:** Let $ID|_\ell = (I_1, \ldots, I_\ell) \in \mathcal{I}^\ell$. It outputs a ciphertext header $CH_{ID|_\ell} = \big(C_0 = g^t, C_1 = (h\prod_{i=1}^{\ell} u^{I_i})^t\big)$ and a session key $EK = \Lambda^t$.

HIBE.Decaps($CT_{ID|_\ell}, SK_{ID'|_k}, PP$)**:** Let $CH_{ID|_\ell} = (C_0, C_1, C_2)$ and $SK_{ID'|_k} = (K_0, K_1, \{K_{2,i}\}_{i=k+1}^{L})$. If $ID'|_k \in$ **Prefix**$(ID|_\ell)$, then it outputs a session key EK by calculating $e(C_0, K_0) \cdot e(C_1, K_1 \prod_{i=k+1}^{\ell} (K_{2,i})^{I_i})$. Otherwise, it outputs $\perp$.

The following two additional algorithms are very important for our modular RHIBE construction. The **ChangeKey** algorithm can change the master key exponent of an HIBE private key by supporting the addition of exponent values and the scalar multiplication. The **MergeKey** algorithm can add master key exponents of two HIBE private keys by merging two HIBE private keys for the same hierarchical identity. As pointed by Lee and Park [17], HIBE schemes in the commutative blinding category can easily support these algorithms [4, 5, 18]. The additional two algorithms are described as follows:

HIBE.ChangeKey($SK_{ID|_k}, \{(op_i, \delta_i)\}_{i=1}^{n}, PP$)**:** Let $SK_{ID|_k} = (K'_0, K'_1, \{K'_{2,i}\}_{i=k+1}^{L})$ and $op_i \in \{+, \times\}$. It sets $TSK^{(0)} = SK_{ID|_k}$. For each (op_i, δ_i), it performs: If $op_i = +$, then it sets $TSK^{(i)} = (K_0^{(i)} = K_0^{(i-1)} \cdot g^{\delta_i}, K_1^{(i)} = K_1^{(i-1)}, \{K_{2,j}^{(i)} = K_{2,j}^{(i-1)}\}_{j=k+1}^{L})$. If $op_i = \times$, then it sets $TSK^{(i)} = (K_0^{(i)} = (K_0^{(i-1)})^{\delta_i}, K_1^{(i)} = (K_1^{(i-1)})^{\delta_i}, \{K_{2,j}^{(i)} = (K_{2,j}^{(i-1)})^{\delta_i}\}_{j=k+1}^{L})$. It outputs a new private key $SK_{ID|_k}$ by running **HIBE.RandKey** ($TSK^{(n)}, PP$).

HIBE.MergeKey($SK_{ID|_k}^{(1)}, SK_{ID|_k}^{(2)}, \eta, PP$)**:** Let $SK_{ID|_k}^{(1)} = (K'_0, K'_1, \{K'_{2,i}\}_{i=k+1}^{L})$ and $SK_{ID|_k}^{(2)} = (K''_0, K''_1, \{K''_{2,i}\}_{i=k+1}^{L})$ be two private keys for the same identity $ID|_k$. It computes a temporal private key $TSK = \big(K_0 = K'_0 \cdot K''_0, K_1 = K'_1 \cdot K''_1, \big\{K_{2,i} = K'_{2,i} \cdot K''_{2,i}\big\}_{i=k+1}^{L}\big)$. Next, it outputs a merged private key $SK_{ID|_k}$ by running **HIBE.ChangeKey**($TSK, (+, \eta), PP$). Note that the master key part is $\gamma_1 + \gamma_2 + \eta$ if the master key parts of $SK_{ID|_k}^{(1)}$ and $SK_{ID|_k}^{(2)}$ are γ_1 and γ_2 respectively.

Theorem 3.1 ([18])**.** *The above HIBE scheme is AD-IND-CPA secure if the SD, GSD, and ComDH assumptions hold.*

3.2 IBE Scheme

We use a KEM version of the LW-IBE scheme that is a special case of the LW-HIBE scheme [18]. For our modular RHIBE construction, we also define two additional algorithms of IBE.

IBE.Setup(GDS): Let $GDS = ((N, \mathbb{G}, \mathbb{G}_T, e), g_1, g_3)$ be the description of a bilinear group with generators $g_1 \in \mathbb{G}_{p_1}, g_3 \in \mathbb{G}_{p_3}$. It selects random elements $v, w \in \mathbb{G}_{p_1}$ and a random exponent $\beta \in \mathbb{Z}_N$. It outputs a master key $MK = \beta$ and public parameters $PP = \big((N, \mathbb{G}, \mathbb{G}_T, e), g = g_1, Y = g_3, v, w, \Lambda = e(g, g)^{\beta}\big)$.

IBE.GenKey(T, MK, PP): Let $T \in \mathcal{T}$ and $MK = \beta$. It chooses random $r \in \mathbb{Z}_N, Y_0, Y_1 \in G_{p_3}$ and outputs a private key $SK_T = \big(K_0 = g^{\beta}(vw^T)^r Y_0, K_1 = g^{-r} Y_1\big)$.

IBE.RandKey(SK_T, PP): Let $SK_T = (K_0', K_1')$. It chooses random $r' \in \mathbb{Z}_N, Y_0', Y_1' \in \mathbb{G}_{p_3}$ and outputs a randomized private key $SK_T = \big(K_0 = K_0' \cdot (vw^T)^{r'} Y_0', K_1 = K_1' \cdot g^{-r'} Y_1'\big)$.

IBE.Encaps(T, t, PP): It outputs a ciphertext header $CH_T = \big(C_0 = g^t, C_1 = (vw^T)^t\big)$ and a session key $EK = \Lambda^t$.

IBE.Decaps($CT_T, SK_{T'}, PP$): Let $CH_T = (C_0, C_1)$ and $SK_{T'} = (K_0, K_1)$. If $T' = T$, then it outputs a session key EK by calculating $e(C_0, K_0) \cdot e(C_1, K_1)$. Otherwise, it outputs $\perp$.

IBE.ChangeKey($SK_T, \{(op_i, \delta_i)\}_{i=1}^n, PP$): Let $SK_T = (K_0', K_1')$ and $op_i \in \{+, \times\}$. It sets $TSK^{(0)} = SK_T$. For each (op_i, δ_i), it performs: If $op_i = +$, then it sets $TSK^{(i)} = (K_0^{(i)} = K_0^{(i-1)} \cdot g^{\delta_i}, K_1^{(i)} = K_1^{(i-1)})$. If $op_i = \times$, then it sets $TSK^{(i)} = (K_0^{(i)} = (K_0^{(i-1)})^{\delta_i}, K_1^{(i)} = (K_1^{(i-1)})^{\delta_i})$. It outputs a new private key SK_T by running **IBE.RandKey**$(TSK^{(n)}, PP)$.

IBE.MergeKey($SK_T^{(1)}, SK_T^{(2)}, \eta, PP$): Let $SK_T^{(1)} = (K_0', K_1')$ and $SK_T^{(2)} = (K_0'', K_1'')$ be two private keys for the same T. It computes a temporal private key $TSK = \big(K_0 = K_0' \cdot K_0'', K_1 = K_1' \cdot K_1''\big)$. Next, it outputs a merged private key SK_T by running **IBE.ChangeKey**$(TSK, (+, \eta), PP)$. Note that the master key part is $\beta_1 + \beta_2 + \eta$ if the master key parts of $SK_T^{(1)}$ and $SK_T^{(2)}$ are β_1 and β_2 respectively.

Theorem 3.2 ([18]). *The above IBE scheme is AD-IND-CPA secure if the SD, GSD, and ComDH assumptions hold.*

4 Revocable HIBE with Complete Subtree

In this section, we propose an RHIBE scheme via the complete subtree method and prove its adaptive security under simple static assumptions.

4.1 The CS Scheme

The complete subtree (CS) scheme is a specific instance of the subset cover framework of Naor et al. [20]. We follow the definition the CS scheme in the work of Lee and Park [17].

CS.Setup(N_{max}): Let $N_{max} = 2^n$ for simplicity. It first sets a full binary tree $\mathcal{BT}$ of depth n. Each user is assigned to a different leaf node in $\mathcal{BT}$. The collection $\mathcal{S}$ is defined as $\{S_i\}$ where S_i is the set of all leaves in a subtree $\mathcal{T}_i$ with a subroot $v_i \in \mathcal{BT}$. It outputs the full binary tree $\mathcal{BT}$.

CS.Assign($\mathcal{BT}, ID$): Let v_{ID} be a leaf node of $\mathcal{BT}$ that is assigned to the user ID. Let $(v_{k_0}, v_{k_1}, \ldots, v_{k_n})$ be the path from the root node $v_{k_0} = v_0$ to the leaf node $v_{k_n} = v_{ID}$. For all $j \in \{k_0, \ldots, k_n\}$, it adds S_j into PV_{ID}. It outputs the private set $PV_{ID} = \{S_j\}$.

CS.Cover($\mathcal{BT}, R$): It first computes the Steiner tree $ST(R)$. Let $\mathcal{T}_{k_1}, \ldots \mathcal{T}_{k_m}$ be all the subtrees of $\mathcal{BT}$ that hang off $ST(R)$, that is all subtrees whose roots $v_{k_1}, \ldots v_{k_m}$ are not in $ST(R)$ but adjacent to nodes of outdegree 1 in $ST(R)$. For all $i \in \{k_1, \ldots, k_m\}$, it adds S_i into CV_R. It outputs a covering set $CV_R = \{S_i\}$.

CS.Match(CV_R, PV_{ID}): It finds a subset S_k with $S_k \in CV_R$ and $S_k \in PV_{ID}$. If there is such a subset, it outputs (S_k, S_k). Otherwise, it outputs $\perp$.

Lemma 4.1 ([20]). *In the CS scheme, the size of a private set is $O(\log N_{max})$ and the size of a covering set is $O(r \log(N_{max}/r))$ where N_{max} is the maximum number of leaf nodes and r is the size of revoked users R.*

4.2 Construction

To build an RHIBE-CS scheme, we follow the modular design strategy of Lee and Park [17]. That is, we construct an RHIBE-CS scheme by combining HIBE and IBE schemes with special properties and the CS scheme. As mentioned before, we use the LW-HIBE scheme in composite-order bilinear groups as the underlying HIBE scheme for our RHIBE scheme. The LW-HIBE scheme has short ciphertexts similar to the BBG-HIBE scheme, but it is fully secure under static assumptions [18]. Lee and Park [17] also pointed out that the BBG-HIBE scheme also can be used to build a selectively secure RHIBE scheme in a modular way. In this work, we prove the adaptive security of our RHIBE scheme. Our RHIBE-CS scheme is very similar to that of Lee and Park [17] except that it uses composite-order bilinear groups and the underlying HIBE and IBE schemes are replaced by the HIBE and IBE schemes of Lewko and Waters [18].

RHIBE-CS.Setup($1^\lambda, L, N_{max}$): Let λ be a security parameter, L be the maximum depth of a hierarchical identity, and N_{max} be the maximum number of users for each level.

1. It first generates bilinear groups $\mathbb{G}, \mathbb{G}_T$ of composite order $N = p_1 p_2 p_3$ where p_1, p_2, and p_3 are random primes. It sets $GDS = ((N, \mathbb{G}, \mathbb{G}_T, e), g_1, g_2)$ where g_i is a random generator of $\mathbb{G}_{p_i}$. It obtains MK_{HIBE} and PP_{HIBE} by running **HIBE.Setup**(GDS, L). It also obtains MK_{IBE} and PP_{IBE} by running **IBE.Setup**(GDS).
2. It selects a random exponent $\alpha \in \mathbb{Z}_N$ and outputs a master key $MK = \alpha$ and public parameters $PP = \big(PP_{HIBE}, PP_{IBE}, \Omega = e(g_1, g_1)^\alpha, N_{max}\big)$. For notational simplicity, we define $SK_{ID|_0} = MK$.

RHIBE-CS.GenKey($ID|_k, ST_{ID|_{k-1}}, PP$): Let $ID|_k = (I_1, \ldots, I_k) \in \mathcal{I}^k$ be a hierarchical identity with $k \geq 1$ and $ST_{ID|_{k-1}}$ be a state information.

1. If $ST_{ID|_{k-1}}$ is empty (since it is first called), then it obtains $\mathcal{BT}_{ID|_{k-1}}$ by running **CS.Setup**(N_{max}) and generates a false master key $\beta_{ID|_{k-1}}$ and a PRF key $z_{ID|_{k-1}}$. Next, it sets $ST_{ID|_{k-1}} = (\mathcal{BT}_{ID|_{k-1}}, \beta_{ID|_{k-1}}, z_{ID|_{k-1}})$.
2. It assigns $ID|_k$ to a random leaf node $v \in \mathcal{BT}_{ID|_{k-1}}$ and obtains a private set $PV_{ID|_k} = \{S_j\}$ by running **CS.Assign**$(\mathcal{BT}_{ID|_{k-1}}, ID|_k)$.
3. For each $S_j \in PV_{ID|_k}$, it computes $\gamma_j = \mathbf{PRF}(z_{ID|_{k-1}}, L_j)$ where $L_j = \mathbf{Label}(S_j)$ and obtains an HIBE private key SK_{HIBE,S_j} by running **HIBE.GenKey**$(ID|_k, \gamma_j, PP)$.
4. Finally, it outputs a private key $SK_{ID|_k} = \big(PV_{ID|_k}, \{SK_{HIBE,S_j}\}_{S_j \in PV_{ID|_k}}\big)$. Note that the master key part of SK_{HIBE,S_j} is γ_j.

RHIBE-CS.UpdateKey($T, RL_{ID|_{k-1}}, DK_{ID|_{k-1},T}, ST_{ID|_{k-1}}, PP$): Let $DK_{ID|_{k-1},T} = (RSK_{HIBE,ID|_{k-1}}, RSK_{IBE,T})$ and $ST_{ID|_{k-1}} = (\mathcal{BT}_{ID|_{k-1}}, \beta_{ID|_{k-1}}, z_{ID|_{k-1}})$ with $k \geq 1$.

1. It first obtains a randomized decryption key $RDK_{ID|_{k-1},T} = (RSK_{HIBE}, RSK_{IBE})$ by running **RHIBE-CS.RandDK**$(DK_{ID|_{k-1},T}, -\beta_{ID|_{k-1}}, PP)$.
2. It derives the set of revoked identities R at time T from $RL_{ID|_{k-1}}$. Next, it obtains a covering set $CV_R = \{S_i\}$ by running **CS.Cover**$(\mathcal{BT}_{ID|_{k-1}}, R)$.
3. For each $S_i \in CV_R$, it computes $\gamma_i = \mathbf{PRF}(z_{ID|_{k-1}}, L_i)$ where $L_i = \mathbf{Label}(S_i)$ and obtains SK_{IBE,S_i} by running **IBE.GenKey**$(T, \beta_{ID|_{k-1}} - \gamma_i, PP)$.
4. Finally, it outputs an update key $UK_{ID|_{k-1},T,R} = \big(RDK_{ID|_{k-1},T}, CV_R, \{SK_{IBE,S_i}\}_{S_i \in CV_R}\big)$. Note that the master key parts of RSK_{HIBE}, RSK_{IBE}, and SK_{IBE,S_i} are η', $\alpha - \eta' - \beta_{ID|_{k-1}}$, and $\beta_{ID|_{k-1}} - \gamma_i$ for some random η' respectively.

RHIBE-CS.DeriveKey($ID|_k, T, SK_{ID|_k}, UK_{ID|_{k-1},T,R}, PP$**):** Let $ID|_k = (I_1, \ldots, I_k)$ with $k \geq 0$, $SK_{ID|_k} = (PV_{ID|_k}, \{SK_{HIBE,S_j}\}_{S_j \in PV_{ID|_k}})$, and $UK_{ID|_{k-1},T,R} = (RDK_{ID|_{k-1},T}, CV_R, \{SK_{IBE,S_i}\}_{S_i \in CV_R})$ where $RDK_{ID|_{k-1},T} = (RSK'_{HIBE,ID|_{k-1}}, RSK'_{IBE,T})$.

If $k = 0$, then $SK_{ID|_0} = MK = \alpha$ and $UK_{ID|_{-1},T,R}$ is empty. It proceeds as follows:

1. It selects a random exponent $\eta \in \mathbb{Z}_N$. It then obtains $RSK_{HIBE,ID|_0}$ and $RSK_{IBE,T}$ by running **HIBE.GenKey**$(ID|_0, \eta, PP)$ and **IBE.GenKey**$(T, \alpha - \eta, PP)$ respectively.
2. It outputs a decryption key $DK_{ID|_0,T} = (RSK_{HIBE,ID|_0}, RSK_{IBE,T})$.

If $k \geq 1$, then it proceeds as follows:

1. If $ID|_k \notin R$, then it obtains (S_i, S_i) by running **CS.Match**$(CV_R, PV_{ID|_k})$. Otherwise, it outputs $\perp$. Next, it retrieves SK_{HIBE,S_i} from $SK_{ID|_k}$ and SK_{IBE,S_i} from $UK_{ID|_{k-1},T,R}$.
2. It obtains $RSK''_{HIBE,ID|_k}$ by running **HIBE.Delegate**$(ID|_k, RSK'_{HIBE,ID|_{k-1}}, PP)$ since $ID|_{k-1} \in$ **Prefix**$(ID|_k)$. Next, it selects a random exponent $\eta \in \mathbb{Z}_N$ and obtains $RSK_{HIBE,ID|_k}$ and $RSK_{IBE,T}$ by running **HIBE.MergeKey**$(RSK''_{HIBE,ID|_k}, SK_{HIBE,S_i}, \eta, PP)$ and **IBE.MergeKey**$(RSK'_{IBE}, SK_{IBE,S_i}, -\eta, PP)$ respectively.
3. Finally, it outputs a decryption key $DK_{ID|_k,T} = \big(RSK_{HIBE,ID|_k}, RSK_{IBE,T}\big)$.

Note that the master key parts of $RSK_{HIBE,ID|_k}$ and $RSK_{IBE,T}$ are η' and $\alpha - \eta'$ for some random η' respectively.

RHIBE-CS.RandDK($DK_{ID|_k,T}, \beta, PP$**):** Let $DK_{ID|_k,T} = \big(RSK'_{HIBE,ID|_k}, RSK'_{IBE,T}\big)$ and $\beta \in \mathbb{Z}_N$ be an exponent. It first selects a random exponent $\eta \in \mathbb{Z}_N$ and obtains $RSK_{HIBE,ID|_k}$ and $RSK_{IBE,T}$ by running **HIBE.ChangeKey**$(RSK'_{HIBE,ID|_k}, (+, \eta), PP)$ and **IBE.ChangeKey**$(RSK'_{IBE,T}, (+, -\eta + \beta), PP)$ respectively. It outputs a randomized decryption key $DK_{ID|_k,T} = \big(RSK_{HIBE,ID|_k}, RSK_{IBE,T}\big)$. Note that the master key parts of $RSK_{HIBE,ID|_k}$ and $RSK_{IBE,T}$ are η' and $\alpha - \eta' + \beta$ respectively.

RHIBE-CS.Encrypt($ID|_\ell, T, M, PP$**):** Let $ID|_\ell = (I_1, \ldots, I_\ell) \in \mathcal{I}^\ell$ be a hierarchical identity with $\ell \geq 1$. It first chooses a random exponent $t \in \mathbb{Z}_N$. Next, it obtains $CH_{HIBE,ID|_\ell}$ and EK_{HIBE} by running **HIBE.Encaps**$(ID|_\ell, t, PP)$. It also obtains $CH_{IBE,T}$ and EK_{IBE} by running **IBE.Encaps**(T, t, PP). It outputs a ciphertext $CT_{ID|_k,T} = \big(CH_{HIBE,ID|_\ell}, CH_{IBE,T}, C = \Omega^t \cdot M\big)$.

RHIBE-CS.Decrypt($CT_{ID|_\ell,T}, DK_{ID'|_k,T'}, PP$**):** Let $CT_{ID|_\ell,T} = (CH_{HIBE,ID|_\ell}, CH_{IBE,T}, C)$ and $DK_{ID'|_k,T'} = (RSK_{HIBE,ID'|_k}, RSK_{IBE,T'})$. If $ID'|_k \in$ **Prefix**$(ID|_\ell)$ and $T = T'$, then it obtains EK_{HIBE} and EK_{IBE} by running **HIBE.Decaps**$(CH_{HIBE,ID|_\ell}, RSK_{HIBE,ID|_k}, PP)$ and **IBE.Decaps**$(CH_{IBE,T}, RSK_{IBE,T'}, PP)$ respectively. Otherwise, it outputs $\perp$. It outputs an encrypted message $M = C \cdot (EK_{HIBE} \cdot EK_{IBE})^{-1}$.

RHIBE-CS.Revoke($ID|_k, T, RL_{ID|_{k-1}}, ST_{ID|_{k-1}}$)**:** If $ID|_k$ is not assigned in $\mathcal{BT}_{ID|_{k-1}}$, then it outputs $\perp$. Otherwise, it updates $RL_{ID|_{k-1}}$ by adding $(ID|_k, T)$ to $RL_{ID|_{k-1}}$.

Remark 1. The RHIBE scheme of Lee and Park [17] uses intermediate private keys of HIBE to reduce the size of a private key instead of using normal private keys of HIBE. However, our RHIBE-CS scheme just uses normal private keys of HIBE since the BBG-HIBE and LW-HIBE schemes cannot have intermediate private keys.

The correctness of our RHIBE-CS scheme can be easily checked by using the correctness of HIBE, IBE, and CS schemes. We omit the correctness.

4.3 Security Analysis

To prove the adaptive security of our RHIBE-CS scheme, we use the dual system encryption proof technique of Lewko and Waters [18]. As mentioned before, we simply cannot change normal private keys and normal update keys into semi-functional keys one by one through hybrid games. Instead, we divide private keys and update keys into small component keys and these small component keys are grouped together if they are related to the same node in a binary tree. The security proof is described as follows.

Theorem 4.2. *The above RHIBE-CS scheme is AD-IND-CPA secure if the SD, GSD, and ComDH assumptions hold.*

Proof. We first define the semi-functional type of HIBE private keys, HIBE ciphertext, IBE private keys, and IBE ciphertexts. For the semi-functional type, we let g_2 denote a fixed generator of the subgroup $\mathbb{G}_{p_2}$.

HIBE.GenKeySF-1. Let $SK'_{ID|_k} = (K'_0, K'_1, \{K'_{2,i}\}_{i=k+1}^{L})$ be a normal private key. It chooses random exponents $a_0, b_0, \{z_i\}_{i=k+1}^{L} \in \mathbb{Z}_N$ and outputs a semi-functional type-1 private key $SK_{ID|_k} = \big(K_0 = K'_0 g_2^{a_0}$, $K_1 = K'_1 g_2^{b_0}, \{K_{2,i} = K'_{2,i} g_2^{b_0 z_i}\}_{i=k+1}^{L}\big)$.

HIBE.GenKeySF-2. Let $SK'_{ID|_k} = (K'_0, K'_1, \{K'_{2,i}\}_{i=k+1}^{L})$ be a normal private key. It chooses a random exponent $a_0 \in \mathbb{Z}_N$ and outputs a semi-functional type-2 HIBE private key $SK_{ID|_k} = \big(K_0 = K'_0 g_2^{a_0}, K_1 = K'_1, \{K_{2,i} = K'_{2,i}\}_{i=k+1}^{L}\big)$.

HIBE.GenKeySF. Let $SK'_{ID|_k} = (K'_0, K'_1, \{K'_{2,i}\}_{i=k+1}^{L})$ be a normal private key. Let $\delta_{j,0} \in \mathbb{Z}_N$ be a fixed random exponent that will be defined in RHIBE. It outputs a semi-functional HIBE private key $SK_{ID|_k} = \big(K_0 = K'_0 g_2^{\delta_{j,0}}, K_1 = K'_1, \{K_{2,i} = K'_{2,i}\}_{i=k+1}^{L}\big)$.

HIBE.EncryptSF. Let $CH'_{ID|_\ell} = (C'_0, C'_1)$ be a normal ciphertext header. It chooses random exponents $c, d_0 \in \mathbb{Z}_N$ and outputs a semi-functional HIBE ciphertext header $CH_{ID|_\ell} = \big(C_0 = C'_0 g_2^{c}, C_1 = C'_1 g_2^{cd_0}\big)$.

Note that if a semi-functional type-1 HIBE private key are used to decrypt a semi-functional HIBE ciphertext, then an additional random element $e(g_2, g_2)^{c(a_0 + \sum_{i=k+1}^{\ell} b_0 z_i I_i - b_0 d_0)}$ is left. If $a_0 + \sum_{i=k+1}^{\ell} b_0 z_i I_i = b_0 d_0$, then this HIBE private key is *nominally* semi-functional type-1.

IBE.GenKeySF-1. Let $SK'_T = (K'_0, K'_1)$ be a normal private key. It chooses random exponents $a_1, b_1 \in \mathbb{Z}_N$ and outputs a semi-functional type-1 IBE private key $SK_T = \big(K_0 = K'_0 g_2^{a_1}, K_1 = K'_1 g_2^{b_1}\big)$.

IBE.GenKeySF-2. Let $SK'_T = (K'_0, K'_1)$ be a normal private key. It chooses a random exponent $a_1 \in \mathbb{Z}_N$ and outputs a semi-functional type-2 IBE private key $SK_T = \big(K_0 = K'_0 g_2^{a_1}, K_1 = K'_1\big)$.

IBE.GenKeySF. Let $SK'_T = (K'_0, K'_1)$ be a normal private key. Let $\delta_{i,1} \in \mathbb{Z}_N$ be a fixed random exponent that will be defined in RHIBE. It outputs a semi-functional IBE private key $SK_T = \big(K_0 = K'_0 g_2^{\delta_{i,1}}, K_1 = K'_1\big)$.

IBE.EncryptSF. Let $CH'_T = (C'_0, C'_1)$ be a normal ciphertext header. It chooses random exponents $c, d_1 \in \mathbb{Z}_N$ and outputs a semi-functional IBE ciphertext header $CH_T = \big(C_0 = C'_0 g_2^c, C_1 = C'_1 g_2^{cd_1}\big)$.

Note that if a semi-functional type-1 IBE private key is used to decrypt a semi-functional IBE ciphertext, then an additional random element $e(g_2, g_2)^{c(a_1 - b_1 d_1)}$ is left. If $a_1 = b_1 d_1$, then this IBE private key is *nominally* semi-functional type-1.

We now define the semi-functional types of private keys, update keys, decryption keys, and ciphertexts in RHIBE by using the semi-functional HIBE and IBE types.

RHIBE-CS.GenKeySF. It first creates a normal private key $SK'_{ID|_k} = (PV_{ID|_k}, \{SK'_{HIBE,S_j}\}_{S_j \in PV_{ID|_k}})$ by using MK where $PV_{ID|_k} = \{S_j\}$. For each $S_j \in PV_{ID|_k}$, it fixes a random exponent $\delta_{j,0} \in \mathbb{Z}_N$ for $S_j \in \mathcal{BT}_{ID|_{k-1}}$ and converts a normal SK'_{HIBE,S_j} to a semi-functional SK_{HIBE,S_j} with the exponent $\delta_{j,0}$. It outputs a semi-functional private key $SK_{ID|_k} = (PV_{ID|_k}, \{SK_{HIBE,S_j}\}_{S_j \in PV_{ID|_k}})$.

RHIBE-CS.UpdateKeySF. It first creates a normal update key $UK'_{ID|_{k-1},T,R} = (RDK'_{ID|_{k-1},T}, CV_R, \{SK'_{IBE,S_i}\}_{S_i \in CV_R})$ by using MK where $CV_R = \{S_i\}$. Let $RDK'_{ID|_{k-1},T} = (RSK'_{HIBE,ID|_{k-1}}, RSK'_{IBE,T})$. It chooses random exponents $a_0, a_{ID|_{k-1}} \in \mathbb{Z}_N$ where $a_{ID|_{k-1}}$ is fixed for $\mathcal{BT}_{ID|_{k-1}}$. It converts a normal $RSK'_{HIBE,ID|_{k-1}}$ to a semi-functional type-2 $RSK_{HIBE,ID|_{k-1}}$ with the exponent a_0. It also converts a normal $RSK'_{IBE,T}$ to a semi-functional type-2 $RSK_{IBE,T}$ with the exponent $a_{ID|_{k-1}}$. It sets a semi-functional $RDK_{ID|_{k-1},T} = (RSK_{HIBE,ID|_{k-1}}, RSK_{IBE,T})$. For each $S_i \in CV_R$, it fixes a random exponent $\delta_{i,1} \in \mathbb{Z}_N$ for $S_i \in \mathcal{BT}_{ID|_{k-1}}$ and converts a normal SK'_{IBE,S_i} to a semi-functional SK_{IBE,S_j} with the exponent $\delta_{i,1}$. It outputs a semi-functional update key $UK_{T,R} = (RDK_{ID|_{k-1},T}, CV_R, \{SK_{IBE,S_i}\}_{S_i \in CV_R})$.

RHIBE-CS.DeriveKeySF. It first creates a normal decryption key $DK'_{ID|_k,T} = (RSK'_{HIBE,ID|_k}, RSK'_{IBE,T})$ by using MK. It chooses random exponents $a_0, a_1 \in \mathbb{Z}_N$. It converts a normal $RSK'_{HIBE,ID|_k}$ to a semi-functional type-2 $RSK_{HIBE,ID|_k}$ with the exponent a_0. It also converts a normal $RSK'_{IBE,T}$ to a semi-functional type-2 $RSK_{IBE,T}$ with the exponent a_1. It outputs a semi-functional decryption key $DK_{ID|_k,T} = (RSK_{HIBE,ID|_k}, RSK_{IBE,T})$.

RHIBE-CS.EncryptSF. It first creates a normal ciphertext $CT'_{ID|_\ell,T} = (CH'_{HIBE,ID|_\ell}, CH'_{IBE,T}, C')$. It chooses random exponents $c, d_0, d_1 \in \mathbb{Z}_N$. It converts a normal $CH'_{HIBE,ID|_\ell}$ to a semi-functional $CH_{HIBE,ID|_\ell}$ with exponents c, d_0. It also converts a normal $CH'_{IBE,T}$ to a semi-functional $CH_{IBE,T}$ with exponents c, d_1. It outputs a semi-functional ciphertext $CT_{ID|_\ell,T} = (CH_{HIBE,ID|_k}, CH_{IBE,T}, C')$.

The security proof consists of a sequence of hybrid games: The first game will be the original security game and the last one will be a game in which an adversary has no advantage. We define the games as follows:

Game $\mathbf{G}_0$. This game is the original security game. In this game, all private keys, update keys, decryption keys and the challenge ciphertext are normal.

Game $\mathbf{G}_1$. In the game $\mathbf{G}_1$, the PRFs that are used in the generation of private keys and update keys are changed to be truly random functions.

Game $\mathbf{G}_2$. In this game, the challenge ciphertext is changed to be semi-functional. All other keys are still normal.

Game $\mathbf{G}_3$. Next, we define a new game $\mathbf{G}_3$. In this game, all private keys and all update keys are changed to be semi-functional.

Game $\mathbf{G}_4$. In this game $\mathbf{G}_4$, the remaining decryption keys are changed to be semi-functional. That is, all private keys, update keys, decryption keys, and the challenge ciphertext are now semi-functional.

Game $\mathbf{G}_5$. In the final game $\mathbf{G}_5$, the session key in the semi-functional challenge ciphertext is changed to be random. In this game, the adversary cannot distinguish the challenge messages since the session key is random.

Let $\mathbf{Adv}_{\mathcal{A}}^{G_j}$ be the advantage of $\mathcal{A}$ in the game $\mathbf{G}_j$. We have that $\mathbf{Adv}_{RHIBE,\mathcal{A}}^{AD\text{-}IND\text{-}CPA}(1^\lambda) = \mathbf{Adv}_{\mathcal{A}}^{G_0}$, and $\mathbf{Adv}_{\mathcal{A}}^{G_5} = 0$. From the following Lemmas 4.3, 4.4, 4.5, 4.8, and 4.11, we obtain the equation

$$\begin{aligned}\mathbf{Adv}_{RHIBE,\mathcal{A}}^{AD\text{-}IND\text{-}CPA}(1^\lambda) &\leq \sum_{j=1}^{5} \left|\mathbf{Adv}_{\mathcal{A}}^{G_{j-1}} - \mathbf{Adv}_{\mathcal{A}}^{G_j}\right| \\ &\leq O(qL)\mathbf{Adv}_{\mathcal{B}}^{PRF}(1^\lambda) + \mathbf{Adv}_{\mathcal{B}}^{SD}(1^\lambda) + O(q\log N_{max} + qr_{max}\log N_{max})\mathbf{Adv}_{\mathcal{B}}^{GSD}(1^\lambda) + \\ &\quad \mathbf{Adv}_{\mathcal{B}}^{ComDH}(1^\lambda).\end{aligned}$$

This completes the proof. □

Lemma 4.3. *If the PRF is secure, then no polynomial-time adversary can distinguish $\mathbf{G}_0$ from $\mathbf{G}_1$ with a non-negligible advantage.*

This proof of Lemma 4.3 is relatively straightforward from the security of PRF. That is, we can use additional hybrid games that change a PRF to a truly random function. Note that there are at most $O(qL)$ number of binary trees in the security proof where q is the number of key queries and L is the maximum level of a hierarchical identity. We omit the proof of this lemma.

Lemma 4.4. *If the SD assumption holds, then no polynomial-time adversary can distinguish $\mathbf{G}_1$ from $\mathbf{G}_2$ with a non-negligible advantage.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that distinguishes $\mathbf{G}_1$ from $\mathbf{G}_2$ with a non-negligible advantage. A simulator $\mathcal{B}$ that solves the SD assumption using $\mathcal{A}$ is given: a challenge tuple $D = ((N, \mathbb{G}, \mathbb{G}_T, e), g_{p_1}, g_{p_3})$ and Z where $Z = Z_0 = X_1 \in \mathbb{G}_{p_1}$ or $Z = Z_1 = X_1 R_1 \in \mathbb{G}_{p_1 p_2}$. Then $\mathcal{B}$ that interacts with $\mathcal{A}$ is described as follows:

Setup: $\mathcal{B}$ first chooses random exponents $h', u'_1, \ldots, u'_L, v', w', \alpha \in \mathbb{Z}_N$ and random elements $\Lambda_1, \Lambda_2 \in \mathbb{G}_{T,p_1}$. Next, it builds $PP_{HIBE} = \big((N, \mathbb{G}, \mathbb{G}_T, e), g = g_{p_1}, Y = g_{p_3}, h = g_{p_1}^{h'}, u_1 = g_{p_1}^{u'_1}, \ldots, u_L = g_{p_1}^{u'_L}, \Lambda_1\big)$ and $PP_{IBE} = \big((N, \mathbb{G}, \mathbb{G}_T, e), g = g_{p_1}, Y = g_{p_3}, v = g_{p_1}^{v'}, w = g_{p_1}^{w'}, \Lambda_2\big)$. It sets $MK = \alpha$ and publishes $PP = \big(PP_{HIBE}, PP_{IBE}, \Omega = e(g,g)^\alpha\big)$.

Phase 1: $\mathcal{B}$ creates normal keys by running normal algorithms except that each γ_j is randomly chosen in $\mathbb{Z}_N$ instead of calculating it by running **PRF**. Note that it cannot create semi-functional keys since g_{p_2} is not given.

Challenge: $\mathcal{B}$ builds $CH_{HIBE,ID^*|_\ell} = \big(C_0 = Z, C_1 = (Z)^{h' + \sum_{i=1}^{\ell} u'_i I^*_i}\big)$ and $CH_{IBE,T^*} = \big(C_0 = Z, C_1 = (Z)^{v' + w' T^*}\big)$. Next, it flips a random coin $\mu \in \{0,1\}$ and creates a challenge ciphertext $CT^*_{ID^*|_\ell, T^*} = \big(CH_{HIBE,ID^*|_\ell}, CH_{IBE,T^*}, C = e(Z,g)^\alpha \cdot M^*_\mu\big)$.

Phase 2: Same as Phase 1.
Guess: $\mathcal{A}$ outputs a guess μ'. If $\mu = \mu'$, then $\mathcal{B}$ outputs 1. Otherwise, it outputs 0.

If $Z = Z_0 = X_1$, then the simulation is the same as $\mathbf{G}_1$. If $Z = Z_1 = X_1R_1$, then it is the same as $\mathbf{G}_2$ since the challenge ciphertext is semi-functional by implicitly setting $d_0 \equiv h' + \sum_{i=1}^{\ell} u'I_i^* \mod p_2$ and $d_1 \equiv v' + w'T^* \mod p_2$. Note that d_0 and d_1 are random since $h', u'_1, \ldots, u'_L, v', w'$ modulo p_2 are not correlated with their values modulo p_1 by the Chinese Remainder Theorem (CRT). This completes our proof. $\square$

Lemma 4.5. *If the GSD assumption holds, then no polynomial-time adversary can distinguish $\mathbf{G}_2$ from $\mathbf{G}_3$ with a non-negligible advantage.*

Proof. For the proof of this lemma, we cannot use simple hybrid games that change a normal private key (or normal update key) to a semi-functional private key (or semi-functional update key) one by one since the adversary of RHIBE can query a private key for $ID|_k \in \mathbf{Prefix}(ID^*|_\ell)$ and an update key for T^*. Note that these normal keys cannot directly converted to semi-functional keys since an information theoretic argument cannot be used.

To solve this problem, we first divide each private key and update key into small HIBE private keys and IBE private keys. Recall that a private key $SK_{ID|_k}$ consists of many HIBE private keys and an update key $UK_{ID|_{k-1},T,R}$ consists of a randomized decryption key and many IBE private keys where each HIBE private key (or an IBE private key) is associated with a node v_j (or a subset S_j) in $\mathcal{BT}_{ID|_{k-1}}$. Next, HIBE private keys and IBE private keys that are related to the same node v_j in $\mathcal{BT}_{ID|_{k-1}}$ are grouped together. To uniquely identify a node $v_j \in \mathcal{BT}_{ID|_{k-1}}$, we define a node identifier *NID* of this node as a string $ID|_{k-1}\|L_j$ where $L_j = \mathbf{Label}(S_j)$. To prove this lemma, we change normal HIBE private keys and normal IBE private keys that are related to the same node identifier *NID* into semi-functional keys by defining additional hybrid games. This additional hybrid games are performed for all node identifiers that are used in the key queries of the adversary.

For additional hybrid games that change HIBE private keys (or IBE private keys) that are related to the same node identifier $NID = ID|_{k-1}\|L_j$ from normal keys to semi-functional keys, we need to define an index pair (i_n, i_c) for an HIBE private key (or an IBE private key) that is related to the node $v_j \in \mathcal{BT}_{ID|_{k-1}}$ where i_n is a node index and i_c is a counter index. Suppose that an HIBE private key (or an IBE private key) is related to a node *NID*. The node index i_n for the HIBE private key (or the IBE private key) is assigned as follows: If the node $v_j \in \mathcal{BT}_{ID|_{k-1}}$ with a node identifier *NID* appears first time in key queries, then we set i_n as the number of distinct node identifiers in previous key queries plus one. If the node identifier *NID* already appeared before in key queries, then we set i_n as the value i'_n of previous HIBE private key (or IBE private key) with the same node identifier. The counter index i_c of an HIBE private key is assigned as follows: If the node identifier *NID* appears first time in HIBE private key queries, then we set i_c as one. If the node identifier *NID* appeared before in HIBE private key queries, then we set i_c as the number of HIBE private keys with the same node identifier that appeared before plus one. Similarly, we assigns the counter index i_c of an IBE private key.

For the security proof, we define a sequence of additional hybrid games $\mathbf{G}_{2,1}, \ldots, \mathbf{G}_{2,h}, \ldots, \mathbf{G}_{2,q_n}$ where $\mathbf{G}_2 = \mathbf{G}_{2,0}$ and q_n is the number of all node identifiers that are used in HIBE private keys and IBE private keys of an adversary. In the game $\mathbf{G}_{2,h}$ for $1 \leq h \leq q_n$, the challenge ciphertext is semi-functional, HIBE private keys and IBE private keys with a node index $i_n \leq h$ are semi-functional, the remaining HIBE private keys and IBE private keys with a node index $i_n > h$ are normal, and all randomized decryption keys are still normal. In the game $\mathbf{G}_3$, the remaining randomized decryption keys are changed to be semi-functional.

Let $\mathbf{Adv}_{\mathcal{A}}^{G_j}$ be the advantage of $\mathcal{A}$ in the game $\mathbf{G}_j$. From the following Lemmas 4.6 and 4.7, we have the following equation

$$\begin{aligned}\mathbf{Adv}_{\mathcal{A}}^{G_2} - \mathbf{Adv}_{\mathcal{A}}^{G_3} &\leq \sum_{h=1}^{q_n} \left|\mathbf{Adv}_{\mathcal{A}}^{G_{2,h-1}} - \mathbf{Adv}_{\mathcal{A}}^{G_{2,h}}\right| + \left|\mathbf{Adv}_{\mathcal{A}}^{G_{2,q_n}} - \mathbf{Adv}_{\mathcal{A}}^{G_3}\right| \\ &\leq O(q \log N_{max} + q r_{max} \log N_{max}) \mathbf{Adv}_{\mathcal{B}}^{GSD}(1^{\lambda}).\end{aligned}$$

This completes the proof. □

Lemma 4.6. *If the GSD assumption holds, then no polynomial-time adversary can distinguish $\mathbf{G}_{2,h-1}$ from $\mathbf{G}_{2,h}$ with a non-negligible advantage.*

Proof. We first divide the behavior of an adversary as two types: Type-I and Type-II. We next show that this lemma holds for two types of the adversary. Let $ID^*|_\ell$ be the challenge hierarchical identity and T^* be the challenge time. The adversary types are formally defined as follows:

Type-I. An adversary is Type-I if it queries on a hierarchical identity $ID|_k \notin \mathbf{Prefix}(ID^*|_\ell)$ for all HIBE private keys with the node index h, and it queries on time $T = T^*$ for at least one IBE private key with the node index h.

Type-II. An adversary is Type-II if it queries on time $T \neq T^*$ for all IBE private keys with the node index h. Note that it may query on a hierarchical identity $ID|_k \in \mathbf{Prefix}(ID^*|_\ell)$ for at least one HIBE private key with h, or it may query on a hierarchical identity $ID|_k \notin \mathbf{Prefix}(ID^*|_\ell)$ for all HIBE private keys with h.

If an adversary is Type-I, then all HIBE private keys related with the node index h are changed to be semi-functional through hybrid games by using the restriction $ID|_k \notin \mathbf{Prefix}(ID^*|_\ell)$. After that, the remaining IBE private keys with h are change to be semi-functional. Note that there is no paradox of the dual system encryption when the remaining IBE private keys are changed since HIBE private keys are already semi-functional. If an adversary is Type-II, then all IBE private keys are changed to be semi-functional by using the restriction $T \neq T^*$ and then the remaining HIBE private keys are changed to be semi-functional.

For the Type-I adversary $\mathcal{A}_I$, we define hybrid games $\mathbf{H}_{1,1}, \mathbf{H}_{1,2}, \ldots, \mathbf{H}_{q_c,1}, \mathbf{H}_{q_c,2} = \mathbf{H}'_{q_c,2}, \mathbf{H}'_{q_c,1}, \ldots, \mathbf{H}'_{1,2}, \mathbf{H}'_{1,1}, \mathbf{H}'_{0,2}, \mathbf{H}''$ where $\mathbf{G}_{2,h-1} = \mathbf{H}_{0,2}$, $\mathbf{H}'' = \mathbf{G}_{2,h}$, and q_c is the maximum number of HIBE private key queries for the node index h. The games are formally defined as follows:

Game $\mathbf{H}_{h_c,1}$. This game $\mathbf{H}_{h_c,1}$ for $1 \leq h_c \leq q_c$ is almost the same as $\mathbf{G}_{2,h-1}$ except the generation of HIBE private keys and IBE private keys with the node index h. An IBE private key with an index pair (h, i_c) is generated as normal. An HIBE private key with an index pair (h, i_c) is generated as follows:

- $i_c < h_c$: It generates a normal SK'_{HIBE,S_j} and converts the key to a semi-functional type-2 SK_{HIBE,S_j} by selecting a new random exponent $a_0 \in \mathbb{Z}_N$.
- $i_c = h_c$: It generates a normal SK'_{HIBE,S_j} and converts the key to a semi-functional type-1 SK_{HIBE,S_j} by selecting new random exponents $a_0, b_0, \{z_i\} \in \mathbb{Z}_N$.
- $i_c > h_c$: It simply creates a normal HIBE private key.

Recall that if $a_0 + \sum_{i=k+1}^{\ell} b_0 z_i I_i = b_0 d_0$, then this HIBE private key is *nominally* semi-functional type-1 where d_0 is the exponent of the challenge HIBE ciphertext header.

Game $\mathbf{H}_{h_c,2}$. This game $\mathbf{H}_{h_c,2}$ is almost the same as $\mathbf{H}_{h_c,1}$ except that the HIBE private key for the index pair $(h, i_c = h_c)$ is generated with $b_0 = 0$. That is, this HIBE private key is generated as semi-functional type-2. In the game $\mathbf{H}_{q_c,2}$, all HIBE private keys with the node index h are semi-functional type-2, but all IBE private keys with the node index h are still normal.

Game $\mathbf{H}'_{h_c,1}$. This game $\mathbf{H}'_{h_c,1}$ is almost the same as $\mathbf{H}_{h_c,1}$ except the generation of an HIBE private key with an index pair $(h, i_c \geq h_c)$. This HIBE private key is generated as follows:

- $i_c \geq h_c$: It first generates $SK''_{HIBE,S_j} = (K''_0, K''_1, \{K''_{2,i}\})$ as the same as $\mathbf{H}_{h_c,1}$. Note that this can be normal or semi-functional type-1. It chooses a random exponent $\delta_{j,0} \in \mathbb{Z}_N$ once for the subset S_j and creates a semi-functional HIBE private key $SK_{HIBE,S_j} = \big(K_0 = K''_0 g_2^{\delta_{j,0}}, K_1 = K''_1, \{K_{2,i} = K''_{2,i}\}\big)$.

Game $\mathbf{H}'_{h_c,2}$. This game $\mathbf{H}'_{h_c,2}$ is almost the same as $\mathbf{H}'_{h_c,1}$ except that the HIBE private key with the index pair $(h, i_c = h_c)$ is generated with $b_0 = 0$. The modification is similar to the game $\mathbf{H}'_{h_c,1}$. In the game $\mathbf{H}'_{0,2}$, all HIBE private keys with the node index h are semi-functional where a fixed $\delta_{j,0}$ is used for a subset S_j, but all IBE private keys with the node index h are still normal.

Game $\mathbf{H}''$. This game $\mathbf{H}''$ is the same as $\mathbf{G}_{2,h}$. Compared to the game $\mathbf{H}'_{0,2}$, all normal IBE private keys with the node index h are changed to be semi-functional by using a fixed $\delta_{i,1}$ for a subset S_i.

Let $\mathbf{Adv}_{\mathcal{A}_I}^{H_i}$ be the advantage of $\mathcal{A}_I$ in a game $\mathbf{H}_i$. From the following Lemmas 4.12, 4.13, 4.14, 4.15, and 4.16, we obtain the following equation

$$\begin{aligned}
\mathbf{Adv}_{\mathcal{A}_I}^{H_{0,2}} - \mathbf{Adv}_{\mathcal{A}_I}^{H''} \leq & \sum_{h_c=1}^{q_c} \big|\mathbf{Adv}_{\mathcal{A}_I}^{H_{h_c-1,2}} - \mathbf{Adv}_{\mathcal{A}_I}^{H_{h_c,1}}\big| + \sum_{h_c=1}^{q_c} \big|\mathbf{Adv}_{\mathcal{A}_I}^{H_{h_c,1}} - \mathbf{Adv}_{\mathcal{A}_I}^{H_{h_c,2}}\big| + \\
& \sum_{h_c=1}^{q_c} \big|\mathbf{Adv}_{\mathcal{A}_I}^{H'_{h_c,2}} - \mathbf{Adv}_{\mathcal{A}_I}^{H'_{h_c,1}}\big| + \sum_{h_c=1}^{q_c} \big|\mathbf{Adv}_{\mathcal{A}_I}^{H'_{h_c,1}} - \mathbf{Adv}_{\mathcal{A}_I}^{H'_{h_c-1,2}}\big| + \big|\mathbf{Adv}_{\mathcal{A}_I}^{H'_{0,2}} - \mathbf{Adv}_{\mathcal{A}_I}^{H''}\big| \\
\leq & \, O(q_c)\mathbf{Adv}_{\mathcal{B}}^{GSD}(1^\lambda).
\end{aligned}$$

For the Type-II adversary $\mathcal{A}_I$, we define hybrid games $\mathbf{I}_{1,1}, \mathbf{I}_{1,2}, \ldots, \mathbf{I}_{q_c,1}, \mathbf{I}_{q_c,2} = \mathbf{I}'_{q_c,2}, \mathbf{I}'_{q_c,1}, \ldots, \mathbf{I}'_{1,2}, \mathbf{I}'_{1,1}, \mathbf{I}'_{0,2}, \mathbf{I}''$ where $\mathbf{G}_{2,h-1} = \mathbf{I}_{0,2}$, $\mathbf{I}'' = \mathbf{G}_{2,h}$, and q_c is the maximum number of IBE private key queries for the node index h. The games are formally defined as follows:

Game $\mathbf{I}_{h_c,1}$. This game $\mathbf{I}_{h_c,1}$ for $1 \leq h_c \leq q_c$ is almost the same as $\mathbf{G}_{1,h-1}$ except the generation of HIBE private keys and IBE private keys with the node index h. An HIBE private key with an index pair (h, i_c) is generated as normal. An IBE private key with an index pair (h, i_c) is generated as follows:

- $i_c < h_c$: It generates a normal SK'_{IBE,S_i} and converts the key to a semi-functional type-2 SK_{IBE,S_i} by selecting a new random exponent $a_1 \in \mathbb{Z}_N$.
- $i_c = h_c$: It generates a normal SK'_{IBE,S_i} and converts the key to a semi-functional type-1 SK_{IBE,S_i} by selecting new random exponents $a_1, b_1 \in \mathbb{Z}_N$.
- $i_c > h_c$: It simply creates a normal IBE private key.

Recall that if $a_1 = b_1 d_1$, then this IBE private key is *nominally* semi-functional type-1 where d_1 is the exponent of the challenge IBE ciphertext header.

Game $\mathbf{I}_{h_c,2}$. This game $\mathbf{I}_{h_c,2}$ is almost the same as $\mathbf{I}_{h_c,1}$ except that the IBE private key for the index pair $(h, i_c = h_c)$ is generated with $b_1 = 0$. That is, this IBE private key is generated as semi-functional type-2. In the game $\mathbf{I}_{q_c,2}$, all IBE private keys with the node index h are semi-functional type-2, but all HIBE private keys with the node index h are still normal.

Game $\mathbf{I}'_{h_c,1}$. This game $\mathbf{I}'_{h_c,1}$ is almost the same as $\mathbf{I}_{h_c,1}$ except the generation of an IBE private key with an index pair $(h, i_c \geq h_c)$. This IBE private key is generated as follows:

- $i_c \geq h_c$: It first generates $SK''_{IBE,S_i} = (K''_0, K''_1)$ as the same as $\mathbf{I}_{h_c,1}$. Note that this can be normal or semi-functional type-1. It chooses a random exponent $\delta_{i,1} \in \mathbb{Z}_N$ once for the subset S_i and creates a semi-functional IBE private key $SK_{IBE,S_i} = \big(K_0 = K''_0 g_2^{\delta_{i,1}}, K_1 = K''_1\big)$.

Game $\mathbf{I}'_{h_c,2}$. This game $\mathbf{I}'_{h_c,2}$ is almost the same as $\mathbf{I}'_{h_c,1}$ except that the IBE private key with the index pair $(h, i_c = h_c)$ is generated with $b_1 = 0$. This modification is similar to the game $\mathbf{I}'_{h_c,1}$. In the game $\mathbf{I}'_{0,2}$, all IBE private keys with the node index h are semi-functional where a fixed $\delta_{i,1}$ is used for a subset S_i, but all HIBE private keys with the node index h are still normal.

Game $\mathbf{I}''$. This game $\mathbf{I}''$ is the same as $\mathbf{G}_{2,h}$. Compared to the game $\mathbf{I}'_{0,2}$, all normal HIBE private keys with the node index h are changed to be semi-functional by using a fixed $\delta_{i,1}$ for a subset S_i.

Let $\mathbf{Adv}^{I_i}_{\mathcal{A}_I}$ be the advantage of $\mathcal{A}_I$ in a game $\mathbf{I}_i$. From the following Lemmas 4.17, 4.18, 4.19, 4.20, and 4.21, we obtain the following equation

$$\begin{aligned}
\mathbf{Adv}^{I_{0,2}}_{\mathcal{A}_{II}} - \mathbf{Adv}^{I''}_{\mathcal{A}_I} &\leq \sum_{h_c=1}^{q_c} \big|\mathbf{Adv}^{I_{h_c-1,2}}_{\mathcal{A}_{II}} - \mathbf{Adv}^{I_{h_c,1}}_{\mathcal{A}_{II}}\big| + \sum_{h_c=1}^{q_c} \big|\mathbf{Adv}^{I_{h_c,1}}_{\mathcal{A}_{II}} - \mathbf{Adv}^{I_{h_c,2}}_{\mathcal{A}_{II}}\big| + \\
&\quad \sum_{h_c=1}^{q_c} \big|\mathbf{Adv}^{I'_{h_c,2}}_{\mathcal{A}_{II}} - \mathbf{Adv}^{I'_{h_c,1}}_{\mathcal{A}_{II}}\big| + \sum_{h_c=1}^{q_c} \big|\mathbf{Adv}^{I'_{h_c,1}}_{\mathcal{A}_{II}} - \mathbf{Adv}^{I'_{h_c-1,2}}_{\mathcal{A}_{II}}\big| + \big|\mathbf{Adv}^{I'_{0,2}}_{\mathcal{A}_{II}} - \mathbf{Adv}^{I''}_{\mathcal{A}_{II}}\big| \\
&\leq O(q_c)\mathbf{Adv}^{GSD}_{\mathcal{B}}(1^\lambda).
\end{aligned}$$

This completes our proof. □

Lemma 4.7. *If the GSD assumption holds, then no polynomial-time adversary can distinguish $\mathbf{G}_{2,q_n}$ from $\mathbf{G}_3$ with a non-negligible advantage.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that distinguishes $\mathbf{G}_{2,q_n}$ from $\mathbf{G}_3$ with a non-negligible advantage. A simulator $\mathcal{B}$ that solves the GSD assumption using $\mathcal{A}$ is given: a challenge tuple $D = ((N, \mathbb{G}, \mathbb{G}_T, e), g_{p_1}, g_{p_3}, X_1R_1, R_2Y_1)$ and Z where $Z = Z_0 = X_2Y_2$ or $Z = Z_1 = X_2R_3Y_2$. Then $\mathcal{B}$ that interacts with $\mathcal{A}$ is described as follows:

Setup: $\mathcal{B}$ first chooses random exponents $h', u'_1, \ldots, u'_L, v', w', \alpha \in \mathbb{Z}_N$ and random elements $\Lambda_1, \Lambda_2 \in \mathbb{G}_{T,p_1}$. Next, it builds $PP_{HIBE} = \big((N, \mathbb{G}, \mathbb{G}_T, e), g = g_{p_1}, Y = g_{p_3}, h = g_{p_1}^{h'}, u_1 = g_{p_1}^{u'_1}, \ldots, u_L = g_{p_1}^{u'_L}, \Lambda_1\big)$ and $PP_{IBE} = \big((N, \mathbb{G}, \mathbb{G}_T, e), g = g_{p_1}, Y = g_{p_3}, v = g_{p_1}^{v'}, w = g_{p_1}^{w'}, \Lambda_2\big)$. It sets $MK = \alpha$ and publishes $PP = \big(PP_{HIBE}, PP_{IBE}, \Omega = e(g,g)^\alpha\big)$.

Phase 1: For each query, $\mathcal{B}$ proceeds as follows: If this is a private key query, then it creates a semi-functional one by using MK and R_2Y_1. If this is a decryption key query, then it creates a normal one. If this is an update key query for $ID|_{k-1}$, T, and R, then it creates IBE private keys as semi-functional and it creates a randomized decryption key as follows:

- It first chooses random exponents $\eta', r_1, r_2 \in \mathbb{Z}_N$ and random elements $Y_0', Y_1', \{Y_{2,i}'\}, Y_0'', Y_1'' \in \mathbb{G}_{p_3}$. It also fixes a random exponent $\delta'_{ID|_{k-1}}$ for $\mathcal{BT}_{ID|_{k-1}}$. Next, it builds an HIBE private key $RSK_{HIBE,ID|_{k-1}} = \big(K_0 = (Z)^{\eta'}(h\prod_{i=1}^{k-1} u_i^{I_i})^{r_1} Y_0', K_1 = g^{-r_1}Y_1', \{K_{2,i} = u_i^{r_1}Y_{2,i}'\}\big)$ and an IBE private key $RSK_{IBE,T} = \big(K_0 = g^{\alpha}(Z)^{-\eta'-\beta'_{ID|_{k-1}}}(vw^T)^{r_2}Y_0'', K_1 = g^{-r_2}Y_1''\big)$. It creates $RDK_{ID|_{k-1},T} = \big(RSK_{HIBE,ID|_{k-1}}, RSK_{IBE,T}\big)$.

Challenge: $\mathcal{B}$ first builds $CH_{HIBE,ID^*|_\ell} = \big(C_0 = X_1R_1,\ C_1 = (X_1R_1)^{h'+\sum_{i=1}^{\ell} u_i' I_i^*}\big)$ and $CH_{IBE,T^*} = (C_0 = X_1R_1,\ C_1 = (X_1R_1)^{v'+w'T^*})$. Next, it flips a random coin $\mu \in \{0,1\}$ and creates the semi-functional challenger ciphertext $CT_{ID^*|_\ell,T^*} = (CH_{HIBE,ID^*|_\ell}, CH_{IBE,T^*}, C = e(X_1R_1, g)^{\alpha} \cdot M_\mu^*)$.
Phase 2: Same as Phase 1.
Guess: $\mathcal{A}$ outputs a guess μ'. If $\mu = \mu'$, then $\mathcal{B}$ outputs 1. Otherwise, it outputs 0.

If $Z = Z_0 = X_2Y_2$, then the simulation is the same as $\mathbf{G}_{2,q_n}$. If $Z = Z_1 = X_2R_3Y_2$, then the simulation is the same as $\mathbf{G}_3$ since a randomized decryption key in an update key is generated as semi-functional by implicitly setting $a_0 \equiv c\eta' \mod p_2$ and $a_{ID|_{k-1}} \equiv -c\eta' - c\beta'_{ID|_{k-1}} \mod p_2$ where $c \equiv \log_{g_2}(R_3) \mod p_2$ and two values $\eta' \mod p_2, \beta'_{ID|_{k-1}} \mod p_2$ are random by CRT. Note that there is no paradox of dual system encryption since HIBE private keys in a private key and IBE private keys in an update key are already semi-functional. □

Lemma 4.8. *If the GSD assumption holds, then no polynomial-time adversary can distinguish* $\mathbf{G}_3$ *from* $\mathbf{G}_4$ *with a non-negligible advantage.*

Proof. For the security proof, we define a sequence of hybrid games $\mathbf{G}_{3,1}, \mathbf{G}_{3,2}, \dots, \mathbf{G}_{3,q_{dk}}$ where $\mathbf{G}_3 = \mathbf{G}_{3,0}$ and q_{dk} is the number of decryption key queries of an adversary. In the game $\mathbf{G}_{3,h_d}$ for $1 \le h_d \le q_{dk}$, all private keys, all update keys, and the challenge ciphertext are generated as semi-functional, but decryption keys are generated as follows: The first h_d decryption keys are generated as semi-functional and the remaining decryption keys are generated as normal.

To show that an adversary cannot distinguish $\mathbf{G}_{3,h_d-1}$ from $\mathbf{G}_{3,h_d}$, we additionally define games $\mathbf{J}_{h_d,1}$, $\mathbf{J}_{h_d,2}, \mathbf{J}_{h_d,3}$ where $\mathbf{J}_{h_d-1,3} = \mathbf{G}_{3,h_d-1}$. These games are defined as follows:

Game $\mathbf{J}_{h_d,1}$. This game is almost similar to the game $\mathbf{G}_{3,h_d-1}$ except the generation of h_dth decryption key. Let $RSK'_{HIBE,ID|_k} = (K_0', K_1', \{K_{2,i}'\})$ be a normal HIBE private key and $RSK'_{IBE,T} = (K_0'', K_1'')$ be a normal IBE private key. The h_dth decryption key consists of a semi-functional type-1 HIBE private key $RSK_{HIBE,ID|_k} = (K_0 = K_0' g_2^{a_0}, K_1 = K_1' g_2^{b_0}, \{K_{2,i} = K_{2,i}' g_2^{b_0 z_i}\})$ and a semi-functional type-1 IBE private key $RSK_{IBE,T} = (K_0 = K_0'' g_2^{a_1}, K_1 = K_1'' g_2^{b_1})$ where $a_0, b_0, \{z_i\}, a_1, b_1$ are random exponents in $\mathbb{Z}_N$.

Game $\mathbf{J}_{h_d,2}$. In this game, the HIBE private key and the IBE private key in in the h_dth decryption key is changed to be semi-functional type-2. It is obvious that $\mathbf{J}_{h_d,2} = \mathbf{G}_{3,h_d}$.

Recall that the decryption key is *nominally* semi-functional type-1 if $a_0 + \sum_{i=k+1}^{\ell} b_0 z_i I_i + a_1 = b_0 d_0 + b_1 d_1$.

Let $\mathbf{Adv}_{\mathcal{A}}^{J_{h_d,i}}$ be the advantage of $\mathcal{A}$ in the game $\mathbf{J}_{h_d,i}$. We have that $\mathbf{Adv}_{\mathcal{A}}^{G_3} = \mathbf{Adv}_{\mathcal{A}}^{J_{1,0}}$ and $\mathbf{Adv}_{\mathcal{A}}^{G_4} = \mathbf{Adv}_{\mathcal{A}}^{J_{q_{dk},2}}$. From the following Lemmas 4.9 and 4.10, we obtain the following equation

$$\mathbf{Adv}_{\mathcal{A}}^{G_3} - \mathbf{Adv}_{\mathcal{A}}^{G_4} \le \sum_{h_d=1}^{q_{dk}} \left(\left|\mathbf{Adv}_{\mathcal{A}}^{J_{h_d,0}} - \mathbf{Adv}_{\mathcal{A}}^{J_{h_d,1}}\right| + \left|\mathbf{Adv}_{\mathcal{A}}^{J_{h_d,1}} - \mathbf{Adv}_{\mathcal{A}}^{J_{h_d,2}}\right| \right) \le O(q_{dk}) \mathbf{Adv}_{\mathcal{B}}^{GSD}(1^\lambda).$$

This completes our proof. □

Lemma 4.9. *If the GSD assumption holds, then no polynomial-time adversary can distinguish $\mathbf{J}_{h_d-1,2}$ from $\mathbf{J}_{h_d,1}$ with a non-negligible advantage.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that distinguishes $\mathbf{J}_{h_d-1,2}$ from $\mathbf{J}_{h_d,1}$ with a non-negligible advantage. A simulator $\mathcal{B}$ that solves the GSD assumption using $\mathcal{A}$ is given: a challenge tuple $D = ((N,\mathbb{G},\mathbb{G}_T,e), g_{p_1}, g_{p_3}, X_1R_1, R_2Y_1)$ and Z where $Z = Z_0 = X_2Y_2$ or $Z = Z_1 = X_2R_3Y_2$. Then $\mathcal{B}$ that interacts with $\mathcal{A}$ is described as follows:

Setup: $\mathcal{B}$ first chooses random exponents $h', u'_1, \ldots, u'_L, v', w', \alpha \in \mathbb{Z}_N$ and random elements $\Lambda_1, \Lambda_2 \in \mathbb{G}_{T,p_1}$. Next, it builds $PP_{HIBE} = \big((N,\mathbb{G},\mathbb{G}_T,e), g = g_{p_1}, Y = g_{p_3}, h = g_{p_1}^{h'}, u_1 = g_{p_1}^{u'_1}, \ldots, u_L = g_{p_1}^{u'_L}, \Lambda_1\big)$ and $PP_{IBE} = \big((N,\mathbb{G},\mathbb{G}_T,e), g = g_{p_1}, Y = g_{p_3}, v = g_{p_1}^{v'}, w = g_{p_1}^{w'}, \Lambda_2\big)$. It sets $MK = \alpha$ and publishes $PP = \big(PP_{HIBE}, PP_{IBE}, \Omega = e(g,g)^{\alpha}\big)$.

Phase 1: For each query, $\mathcal{B}$ proceeds as follows: If this is a private key (or update key) query, then it creates a semi-functional one by using MK and R_2Y_1. If this is an h_dth decryption key query for $ID|_k$ and T, then it handles this query as follows:

- $j < h_d$: It creates a semi-functional decryption key by using MK and R_2Y_1.
- $j = h_d$: It first chooses random exponents $\eta', r'_1, r'_2 \in \mathbb{Z}_N$ and random elements $Y'_0, Y'_1, \{Y'_{2,i}\}, Y''_0, Y''_1 \in \mathbb{G}_{p_3}$. Next, it builds an HIBE private key $RSK_{HIBE,ID|_k} = \big(K_0 = (Z)^{\eta' + (h' + \sum_{i=1}^{k} u'_i I_i) r'_1} Y'_0, K_1 = (Z)^{-r'_1} Y'_1, \{K_{2,i} = (Z)^{u'_i r'_1} Y'_{2,i}\}\big)$ and an IBE private key $RSK_{IBE,T} = \big(K_0 = g^{\alpha} (Z)^{-\eta' + (v' + w'T) r'_2} Y''_0,\ K_1 = (Z)^{-r'_2} Y''_1\big)$. It creates $DK_{ID|_k,T} = \big(RSK_{HIBE,ID|_k}, RSK_{IBE,T}\big)$.
- $j > h_d$: It creates a normal decryption key by using MK.

Challenge: $\mathcal{B}$ first builds $CH_{HIBE,ID^*|_\ell} = \big(C_0 = X_1R_1,\ C_1 = (X_1R_1)^{h' + \sum_{i=1}^{\ell} u'_i I^*_i}\big)$ and $CH_{IBE,T^*} = (C_0 = X_1R_1,\ C_1 = (X_1R_1)^{v' + w'T^*})$. Next, it flips a random coin $\mu \in \{0,1\}$ and creates the semi-functional challenger ciphertext $CT_{ID^*|_\ell,T^*} = (CH_{HIBE,ID^*|_\ell}, CH_{IBE,T^*}, C = e(X_1R_1, g)^{\alpha} \cdot M^*_{\mu})$.

Phase 2: Same as Phase 1.

Guess: $\mathcal{A}$ outputs a guess μ'. If $\mu = \mu'$, then $\mathcal{B}$ outputs 1. Otherwise, it outputs 0.

If $Z = Z_0 = X_2Y_2$, then the simulation is the same as $\mathbf{J}_{h_d-1,2}$. If $Z = Z_1 = X_2R_3Y_2$, then the simulation is almost the same as $\mathbf{J}_{h_d,1}$ except that the h_dth decryption key is generated as a nominally semi-functional type-1 by implicitly setting $a_0 \equiv c\eta' + cr'_1(h' + \sum_{i=1}^{k} u'_i I_i) \mod p_2$, $b_0 \equiv cr'_1 \mod p_2$, $z_i \equiv u'_i \mod p_2$, $a_1 \equiv -c\eta' + cr'_2(v' + w'T) \mod p_2$, and $b_1 \equiv cr'_2 \mod p_2$ where $c \equiv \log_{g_2}(R_3)$. Note that we solve the paradox of dual system encryption by introducing the nominally semi-functional decryption key. To finish the proof, we should argue that the adversary cannot distinguish a nominally semi-functional decryption key from a semi-functional decryption key. For this argument, we can easily show an information theoretic argument by using the restriction of a decryption key query in the security model and CRT. We omit the details of this argument. □

Lemma 4.10. *If the GSD assumption holds, then no polynomial-time adversary can distinguish $\mathbf{J}_{h_d,1}$ from $\mathbf{J}_{h_d,2}$ with a non-negligible advantage.*

Proof. The proof of this lemma is almost the same as that of Lemma 4.9, except the generation of the h_dth decryption key. The h_dth decryption key for $ID|_k$ and T is generated as follows:

- $j = h_d$: It chooses random exponents $\eta', r'_1, r'_2, a'_0, a'_1 \in \mathbb{Z}_N$ and random elements $Y'_0, Y'_1, \{Y'_{2,i}\}, Y''_0, Y''_1 \in \mathbb{G}_{p_3}$. Next, it builds an HIBE private key $RSK_{HIBE,ID|_k} = \big(K_0 = (Z)^{\eta' + (h' + \sum_{i=1}^{k} u'_i I_i) r'_1} (R_2 Y_1)^{a'_0} Y'_0,\ K_1 = (Z)^{-r'_1} Y'_1,\ \{K_{2,i} = (Z)^{u'_i r'_1} Y'_{2,i}\}\big)$ and an IBE private key $RSK_{IBE,T} = \big(K_0 = g^{\alpha} (Z)^{-\eta' + (v' + w'T) r'_2} (R_2 Y_1)^{a'_1} Y''_0,$ $K_1 = (Z)^{-r'_1} Y''_1\big)$. It creates $DK_{ID|_k,T} = \big(RSK_{HIBE,ID|_k}, RSK_{IBE,T}\big)$.

Note that the h_dth decryption key is no longer correlated with the challenge ciphertext since K_0 is randomized by $(R_2Y_1)^{a'_0}$. If $Z = Z_0 = X_2Y_2$, then the simulation is the same as $\mathbf{J}_{h_d,2}$. If $Z = Z_1 = X_2R_3Y_2$, then the simulation is the same as $\mathbf{J}_{h_d,1}$. □

Lemma 4.11. *If the ComDH assumption holds, then no polynomial-time adversary can distinguish $\mathbf{G}_4$ from $\mathbf{G}_5$ with a non-negligible advantage.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that distinguish $\mathbf{G}_4$ from $\mathbf{G}_5$ with a non-negligible advantage. A simulator $\mathcal{B}$ that solves the ComDH assumption using $\mathcal{A}$ is given: a challenge tuple $D = ((N, \mathbb{G}, \mathbb{G}_T, e),$ $g_{p_1}, g_{p_2}, g_{p_3}, g_{p_1}^a R_1, g_{p_1}^b R_2)$ and Z where $Z = Z_0 = e(g_{p_1}, g_{p_1})^{ab}$ or $Z = Z_1 = e(g_{p_1}, g_{p_1})^c$. Then $\mathcal{B}$ that interacts with $\mathcal{A}$ is described as follows:

Setup: $\mathcal{B}$ first chooses random exponents $h', u'_1, \ldots, u'_L, v', w' \in \mathbb{Z}_N$ and random elements $\Lambda_1, \Lambda_2 \in \mathbb{G}_{T,p_1}$. Next, it builds $PP_{HIBE} = \big((N, \mathbb{G}, \mathbb{G}_T, e), g = g_{p_1}, Y = g_{p_3}, h = g_{p_1}^{h'}, u_1 = g_{p_1}^{u'_1}, \ldots, u_L = g_{p_1}^{u'_L}, \Lambda_1\big)$ and $PP_{IBE} = \big((N, \mathbb{G}, \mathbb{G}_T, e), g = g_{p_1}, Y = g_{p_3}, v = g_{p_1}^{v'}, w = g_{p_1}^{w'}, \Lambda_2\big)$. It implicitly sets $\alpha = a$ from $g_{p_1}^a R_1$ and publishes $PP = \big(PP_{HIBE}, PP_{IBE}, \Omega = e(g, g_{p_1}^a R_1)\big)$.
Phase 1: For each query, $\mathcal{B}$ creates a semi-functional key by using $g_2^a R_1$ and g_2 that are given in the assumption. Note that it cannot create a normal update key since $g_{p_1}^a$ is not given.
Challenge: $\mathcal{B}$ first builds $CH_{HIBE,ID^*|_\ell} = \big(C_0 = g_{p_1}^b R_2,\ C_1 = (g_{p_1}^b R_2)^{h' + \sum_{i=1}^{\ell} u'_i I_i^*}\big)$ and $CH_{IBE,T^*} = (C_0 = g_{p_1}^b R_2,\ C_1 = (g_{p_1}^b R_2)^{v' + w'T^*})$. Next, it flips a random coin $\mu \in \{0,1\}$ and creates a challenger ciphertext $CT_{ID^*|_\ell,T^*} = (CH_{HIBE,ID^*|_\ell}, CH_{IBE,T^*}, C = Z \cdot M_\mu^*)$.
Phase 2: Same as Phase 1.
Guess: $\mathcal{A}$ outputs a guess μ'. If $\mu = \mu'$, then $\mathcal{B}$ outputs 1. Otherwise, it outputs 0.

If $Z = Z_0$, then the simulation is the same as $\mathbf{G}_4$. If $Z = Z_1$, then the simulation is the same as $\mathbf{G}_5$ since C is random. □

4.4 Type-I Adversary

Lemma 4.12. *If the GSD assumption holds, then no polynomial-time Type-I adversary can distinguish $\mathbf{H}_{h_c-1,2}$ from $\mathbf{H}_{h_c,1}$ with a non-negligible advantage.*

Proof. Suppose there exists an adversary $\mathcal{A}_I$ that distinguishes $\mathbf{H}_{h_c-1,2}$ from $\mathbf{H}_{h_c,1}$ with a non-negligible advantage. A simulator $\mathcal{B}$ that solves the GSD assumption using $\mathcal{A}_I$ is given: a challenge tuple $D = \big((N, \mathbb{G}, \mathbb{G}_T, e), g_{p_1}, g_{p_3}, X_1R_1, R_2Y_1\big)$ and Z where $Z = Z_0 = X_2Y_2$ or $Z = Z_1 = X_2R_3Y_2$. Then $\mathcal{B}$ that interacts with $\mathcal{A}_I$ is described as follows:

Setup: $\mathcal{B}$ first chooses random exponents $h', u'_1, \ldots, u'_L, v', w', \alpha \in \mathbb{Z}_N$ and random elements $\Lambda_1, \Lambda_2 \in \mathbb{G}_{p_3}$. Next, it builds $PP_{HIBE} = \big((N, \mathbb{G}, \mathbb{G}_T, e), g = g_{p_1}, Y = g_{p_3}, h = g_{p_1}^{h'}, u_1 = g_{p_1}^{u'_1}, \ldots, u_L = g_{p_1}^{u'_L}, \Lambda_1\big)$ and $PP_{IBE} = \big((N, \mathbb{G}, \mathbb{G}_T, e), g = g_{p_1}, Y = g_{p_3}, v = g_{p_1}^{v'}, w = g_{p_1}^{w'}, \Lambda_2\big)$. It sets $MK = \alpha$ and publishes $PP = \big(PP_{HIBE}, PP_{IBE},$ $\Omega = e(g,g)^{\alpha}\big)$.

Phase 1: For each query, $\mathcal{B}$ proceeds as follows: If this is a decryption key query, then it creates a normal key. If this is an HIBE private key or an IBE private key query with indexes (i_n, i_c), then $\mathcal{B}$ handles this query as follows:

- **Case** $i_n < h$: It first builds a normal key by using MK and converts the key to a semi-functional one with fixed random exponents $\delta_{j,0}, \delta_{j,1} \in \mathbb{Z}_N$ for the subset S_j by using R_2Y_1.
- **Case** $i_n = h$: If this is an IBE private key query, then it creates a normal key by using MK. If this is an HIBE private key query, then it proceeds as follows:
  - $i_c < h_c$: It first builds a normal HIBE private key and converts the key to a semi-functional type-2 key with a random exponent $a_0 \in \mathbb{Z}_N$ by using R_2Y_1.
  - $i_c = h_c$: It chooses random elements $Y'_0, Y'_1, \{Y'_{2,i}\} \in \mathbb{G}_{p_3}$ and creates an HIBE private key $SK_{HIBE} = \big(K_0 = g^{\gamma_j}(Z)^{h'+\sum_{i=1}^{k} u'_i I_i} Y'_0,\ K_1 = (Z)^{-1} Y'_1,\ \{K_{2,i} = (Z)^{u'_i} Y'_{2,i}\}\big)$.
  - $i_c > h_c$: It creates a normal HIBE private key by using MK.
- **Case** $i_n > h$: It creates a normal HIBE private key or a normal IBE private key.

Challenge: $\mathcal{B}$ first builds $CH_{HIBE,ID^*|_\ell} = \big(C_0 = X_1R_1,\ C_1 = (X_1R_1)^{h'+\sum_{i=1}^{\ell} u'_i I^*_i}\big)$ and $CH_{IBE,T^*} = (C_0 = X_1R_1,\ C_1 = (X_1R_1)^{v'+w'T^*})$. Next, it flips a random coin $\mu \in \{0,1\}$ and creates the semi-functional challenger ciphertext $CT_{ID^*|_\ell,T^*} = (CH_{HIBE,ID^*|_\ell}, CH_{IBE,T^*}, C = e(X_1R_1, g)^{\alpha} \cdot M^*_\mu)$.
Phase 2: Same as Phase 1.
Guess: $\mathcal{A}$ outputs a guess μ'. If $\mu = \mu'$, then $\mathcal{B}$ outputs 1. Otherwise, it outputs 0.

If $Z = Z_0 = X_2Y_2$, then the simulation is the same as $\mathbf{H}_{h_c-1,2}$ since the HIBE private key with $(i_n = h) \wedge (i_c = h_c)$ and the semi-functional challenge HIBE ciphertext header are correctly distributed. If $Z = Z_1 = X_2R_3Y_2$, then the simulation is almost the same as $\mathbf{H}_{h_c,1}$ except that the HIBE private key with $(i_n = h) \wedge (i_c = h_c)$ is generated as a nominally semi-functional type-1 key by implicitly setting $a_0 \equiv \log_{g_2}(R_3)(h' + \sum_{i=1}^{k} u'_i I_i) \mod p_2$, $b_0 \equiv \log_{g_2}(R_3) \mod p_2$, and $z_i \equiv u'_i \mod p_2$. Note that the paradox of dual system encryption is solved by introducing the nominally semi-functional type-1 key. That is, the simulator cannot check whether the HIBE private key is normal or nominally semi-functional type-1 since the nominally semi-functional type-1 key is correlated to the challenge ciphertext.

Next, we should argue that the Type-I adversary cannot distinguish a nominally semi-functional type-1 HIBE private key from a semi-functional type-1 HIBE private key. For this argument, we show an information theoretic argument by using the fact that $ID|_k \notin \mathbf{Prefix}(ID^*|_\ell)$ for all HIBE private key queries with the node index h. Suppose there exists an unbounded Type-I adversary. If the query with $(i_n = h) \wedge (i_c = h_c)$ is an HIBE private key, then the adversary can gather the values $a_0 \equiv b_0(h' + \sum_{i=1}^{k} u'_i I_i) \mod p_2, b_0 \mod p_2$ from the HIBE private key and $d_0 \equiv h' + \sum_{i=1}^{\ell} u'_i I^*_i \mod p_2$ from the challenge HIBE ciphertext header. We easily obtain that $h' + \sum_{i=1}^{k} u'_i I_i \mod p_2$ looks random to the adversary since $h' + u'_j I_j$ is a pair-wise independent function, $\exists j$ such that $I_j \neq I^*_j$ if $ID|_k \notin \mathbf{Prefix}(ID^*|_\ell)$, and $h' \mod p_2$ and $u'_j \mod p_2$ are information theoretically hidden to the adversary. This completes our proof. □

Lemma 4.13. *If the GSD assumption holds, then no polynomial-time Type-I adversary can distinguish $\mathbf{H}_{h_c,1}$ from $\mathbf{H}_{h_c,2}$ with a non-negligible advantage.*

Proof. The proof of this lemma is almost the same as that of Lemma 4.12. The only difference is the generation of an HIBE private key with indexes $(h, i_c = h_c)$. This HIBE private key is generated as follows:

- $i_c = h_c$: It chooses random $a'_0 \in \mathbb{Z}_N, Y'_0, Y'_1, \{Y'_{2,i}\} \in \mathbb{G}_{p_3}$ and creates an HIBE private key $SK_{HIBE,ID|_k} = \big(K_0 = g^{\gamma_j}(Z)^{h'+\sum_{i=1}^{k} u'_i I_i} Y'_0 (R_2 Y_1)^{a'_0},\ K_1 = (Z)^{-1} Y'_1,\ \{K_{2,i} = (Z)^{u'_i} Y'_{2,i}\}\big)$.

Note that this HIBE private key with indexes (h, h_c) is no longer correlated with the challenge ciphertext since K_0 is randomized by $(R_2 Y_1)^{a'_0}$. □

Lemma 4.14. *If the GSD assumption holds, then no polynomial-time Type-I adversary can distinguish* $\boldsymbol{H}'_{h_c-1,2}$ *from* $\boldsymbol{H}'_{h_c,1}$ *with a non-negligible advantage.*

Lemma 4.15. *If the GSD assumption holds, then no polynomial-time Type-I adversary can distinguish* $\boldsymbol{H}'_{h_c,1}$ *from* $\boldsymbol{H}'_{h_c,2}$ *with a non-negligible advantage.*

The proofs of Lemmas 4.14 and 4.15 are almost the same as those of Lemmas 4.12 and 4.13 respectively. The only difference is that each element K_0 of HIBE private keys with the node index h that is generated in Lemma 4.12 (or Lemma 4.13) is additionally multiplied by $(R_2 Y_1)^{\delta'_{j,0}}$ where a fixed exponent $\delta'_{j,0}$ is related with the node v_j. This modification is possible since $R_2 Y_1$ is given in the assumption. In this case, HIBE private keys with the node index h additionally contain $R_2^{\delta'_{j,0}}$. We omit the detailed proofs of these lemmas.

Lemma 4.16. *If the GSD assumption holds, then no polynomial-time Type-I adversary can distinguish* $\boldsymbol{H}'_{0,2}$ *from* $\boldsymbol{H}''$ *with a non-negligible advantage.*

Proof. The proof of this lemma is the important part of the security proof since it changes the IBE private key for T^* from a normal type to a semi-functional type. It should be noted that this changes from normal to semi-functional cannot be handled by introducing a nominally semi-functional type since an information theoretic argument for T^* cannot be used. To solve this problem, we directly change normal keys with the index h to semi-functional keys without introducing nominally semi-functional keys.

Many part of this proof is similar to that of Lemma 4.12 except that the generation of HIBE private keys and IBE private keys with the node index h. These keys with the node index $i_n = h$ are generated as follows:

- **Case** $i_n = h$: Let $\delta'_{j,0}$ be a fixed exponent in $\mathbb{Z}_N$ for the subset S_j in this node index h.
  If this is an HIBE private key query, then it selects random $r_1 \in \mathbb{Z}_N, Y'_0, Y'_1, \{Y'_{2,i}\} \in \mathbb{G}_{p_3}$ and creates an HIBE private key $SK_{HIBE,ID|_k} = \big(K_0 = Z(h\prod_{i=1}^{k} u^{I_i})^{r_1} Y'_0 \cdot (R_2 Y_1)^{\delta'_{j,0}},\ K_1 = g^{-r_1} Y'_1,\ \{K_{2,i} = u_i^{r_1} Y'_{2,i}\}\big)$.
  If this is an IBE private key query, then it selects random $r_2 \in \mathbb{Z}_N, Y''_0, Y''_1 \in \mathbb{G}_{p_3}$ and creates an IBE private key $SK_{IBE,T} = \big(K_0 = g^{\beta_{ID|_{k-1}}}(Z)^{-1}(vw^T)^{-r_2} Y''_0,\ K_1 = g^{-r_2} Y''_1\big)$.

If $Z = Z_0 = X_2 Y_2$, then the simulation is the same as $\mathbf{H}'_{0,2}$ since all HIBE private keys with h are semi-functional and all IBE private keys with h are normal. If $Z = Z_1 = X_2 R_3 Y_2$, then the simulation is the same as $\mathbf{H}''$ since it implicitly sets $\delta_{j,0} = \log_{g_{p_2}}(R_3) + \log_{g_{p_2}}(R_2)\delta'_{j,0} \mod p_2$ and $\delta_{j,1} = -\log_{g_{p_2}}(R_3) \mod p_2$.

We now show that the paradox of dual system encryption does not occur. To check whether an IBE private key with h is normal or semi-functional, the simulator may try to decrypt a semi-functional ciphertext by deriving a decryption key from these keys with h. However, the simulator always derive a semi-functional decryption key from those keys since the HIBE private key with h is already semi-functional. Thus, the simulator cannot check whether the IBE private key with h is normal or semi-functional since the decryption always fails. This completes our proof. □

4.5 Type-II Adversary

Lemma 4.17. *If the GSD assumption holds, then no polynomial-time Type-II adversary can distinguish $\boldsymbol{I}_{h_c-1,2}$ from $\boldsymbol{I}_{h_c,1}$ with a non-negligible advantage.*

Lemma 4.18. *If the GSD assumption holds, then no polynomial-time Type-II adversary can distinguish $\boldsymbol{I}_{h_c,1}$ from $\boldsymbol{I}_{h_c,2}$ with a non-negligible advantage.*

Lemma 4.19. *If the GSD assumption holds, then no polynomial-time Type-II adversary can distinguish $\boldsymbol{I}'_{h_c-1,2}$ from $\boldsymbol{I}'_{h_c,1}$ with a non-negligible advantage.*

Lemma 4.20. *If the GSD assumption holds, then no polynomial-time Type-II adversary can distinguish $\boldsymbol{I}'_{h_c,1}$ from $\boldsymbol{I}'_{h_c,2}$ with a non-negligible advantage.*

Lemma 4.21. *If the GSD assumption holds, then no polynomial-time Type-II adversary can distinguish $\boldsymbol{I}'_{0,2}$ from $\boldsymbol{I}''$ with a non-negligible advantage.*

The proofs of Lemmas 4.17, 4.18, 4.19, 4.20, and 4.21 are almost the same as those of Lemmas 4.12, 4.13, 4.14, 4.15, and 4.16 respectively except that IBE private keys are first changed to semi-functional by using the restriction $T \neq T^*$ and the master key part of an IBE private key is set with the exponent $\beta_{ID|_{k-1}} - \gamma_i$. Note that the IBE scheme is a specific case of the HIBE scheme. We omit the detailed proofs of these lemmas.

5 Revocable HIBE with Subset Difference

In this section, we propose an RHIBE-SD scheme by combining HIBE, IBE, and SD schemes and prove its adaptive security under simple static assumptions.

5.1 The SD Scheme

The subset difference (SD) scheme is also a specific instance of the subset cover framework of Naor et al. [20]. We also follow the SD definition of Lee and Park [17].

SD.Setup(N_{max}): Let $N_{max} = 2^n$ for simplicity. It first sets a full binary tree $\mathcal{BT}$ of depth n. Each user is assigned to a different leaf node in $\mathcal{BT}$. The collection $\mathcal{S}$ of SD is the set of all subsets $\{S_{i,j}\}$ where $v_i, v_j \in \mathcal{BT}$ and v_j is a descendant of v_i. It outputs the full binary tree $\mathcal{BT}$.

SD.Assign($\mathcal{BT}$, ID): Let v_{ID} be the leaf node of $\mathcal{BT}$ that is assigned to the user ID. Let $(v_{k_0}, v_{k_1}, \ldots, v_{k_n})$ be the path from the root node v_{k_0} to the leaf node $v_{k_n} = v_{ID}$. For all $i, j \in \{k_0, \ldots, k_n\}$ such that v_j is a descendant of v_i, it adds the subset $S_{i,j}$ defined by two nodes v_i and v_j in the path into PV_{ID}. It outputs the private set $PV_{ID} = \{S_{i,j}\}$.

SD.Cover($\mathcal{BT}$, R): It first sets a subtree T as the Steiner tree $ST(R)$ that is the minimum subtree of $\mathcal{BT}$ that connects all the leaf nodes in R and the root node, and then it builds a covering set CV_R iteratively by removing nodes from T until T consists of just a single node as follows:

1. It finds two leaf nodes v_i and v_j in T such that the least-common-ancestor v of v_i and v_j does not contain any other leaf nodes of T in its subtree. Let v_l and v_k be the two child nodes of v such that v_i is a descendant of v_l and v_j is a descendant of v_k. If there is only one leaf node left, it makes $v_i = v_j$ to the leaf node, v to be the root of T and $v_l = v_k = v$.

2. If $v_l \neq v_i$, then it adds the subset $S_{l,i}$ to CV_R; likewise, if $v_k \neq v_j$, it adds the subset $S_{k,j}$ to CV_R.
3. It removes from T all the descendants of v and makes v a leaf node.

It outputs the covering set $CV_R = \{S_{i,j}\}$.

SD.Match(CV_R, PV_{ID})**:** It finds two subsets $S_{i,j}$ and $S'_{i',j'}$ such that $S_{i,j} \in CV_R$, $S'_{i',j'} \in PV_{ID}$, $i = i'$, $d_j = d_{j'}$, and $j \neq j'$ where d_j is the depth of v_j. If it found two subsets, then it outputs $(S_{i,j}, S'_{i',j'})$. Otherwise, it outputs $\perp$.

Lemma 5.1 ([20])**.** *In the SD scheme, the size of a private set if $O(\log^2 N_{max})$ and the size of a covering set is $O(r)$ where N_{max} is the maximum number of leaf nodes and r is the size of revoked users R.*

5.2 Construction

Our RHIBE-SD scheme is also very similar to that of Lee and Park [17] except that the underlying HIBE and IBE schemes are replaced by the LW-HIBE and LW-IBE schemes. Recall that Lee and Park proposed a modular design approach of RHIBE, so we can also follow their design approach. Let $\Delta_{i,I}$ be a Lagrange coefficient which is defined as $\Delta_{i,I}(x) = \prod_{j \in I, j \neq i} \frac{x-j}{i-j}$ for an index $i \in \mathbb{Z}_p$ and a set of indexes I in $\mathbb{Z}_p$.

RHIBE-SD.Setup($1^\lambda, L, N_{max}$)**:** Let λ be a security parameter, L be the maximum depth of a hierarchical identity, and N_{max} be the maximum number of users for each level.

1. It first generates bilinear groups $\mathbb{G}, \mathbb{G}_T$ of composite order $N = p_1 p_2 p_3$ where p_1, p_2, and p_3 are random primes. It sets $GDS = ((N, \mathbb{G}, \mathbb{G}_T, e), g_1, g_2)$ where g_i is a random generator of $\mathbb{G}_{p_i}$. It obtains MK_{HIBE} and PP_{HIBE} by running **HIBE.Setup**(GDS, L). It also obtains MK_{IBE} and PP_{IBE} by running **IBE.Setup**(GDS).
2. It selects a random exponent $\alpha \in \mathbb{Z}_N$ and outputs a master key $MK = \alpha$ and public parameters $PP = \big(PP_{HIBE}, PP_{IBE}, \Omega = e(g,g)^\alpha, N_{max}\big)$. For notational simplicity, we define $SK_{ID|_0} = MK$.

RHIBE-SD.GenKey($ID|_k, ST_{ID|_{k-1}}, PP$)**:** Let $ID|_k = (I_1, \ldots, I_k) \in \mathcal{I}^k$ be a hierarchical identity with $k \geq 1$, and $ST_{ID|_{k-1}}$ be a state information.

1. If $ST_{ID|_{k-1}}$ is empty (since it is first called), then it obtains $\mathcal{BT}_{ID|_{k-1}}$ by running **SD.Setup**(N_{max}) and generates a false master key $\beta_{ID|_{k-1}} \in \mathbb{Z}_p$ and a PRF key $z_{ID|_{k-1}}$. Next, it sets $ST_{ID|_{k-1}} = (\mathcal{BT}_{ID|_{k-1}}, \beta_{ID|_{k-1}}, z_{ID|_{k-1}})$.
2. It assigns $ID|_k$ to a random leaf node $v \in \mathcal{BT}_{ID|_{k-1}}$ and obtains a private set $PV_{ID|_k} = \{S_{i,j}\}$ by running **SD.Assign**($\mathcal{BT}_{ID|_{k-1}}, ID|_k$).
3. For each $S_{i,j} \in PV_{ID|_k}$, it defines $f_{GL}(x) = a_{GL}x + \beta_{ID|_{k-1}}$ by computing $a_{GL} = \mathbf{PRF}(z_{ID|_{k-1}}, GL)$ where $(L_i, L_j) = \mathbf{Label}(S_{i,j})$, $d_j = \mathbf{Depth}(S_j)$, and $GL = L_i \| d_j$, and then it obtains SK_{HIBE,S_j} by running **HIBE.GenKey**($ID|_k, f_{GL}(L_j), PP$).
4. Finally, it outputs a private key $SK_{ID|_k} = \big(PV_{ID|_k}, \{SK_{HIBE,S_{i,j}}\}_{S_{i,j} \in PV_{ID|_k}}\big)$. Note that the master key part of $SK_{HIBE,S_{i,j}}$ is $f_{GL}(L_j)$.

RHIBE-SD.UpdateKey($T, RL_{ID|_{k-1}}, DK_{ID|_{k-1},T}, ST_{ID|_{k-1}}, PP$)**:** Let $DK_{ID|_{k-1},T} = (RSK_{HIBE,ID|_{k-1}}, RSK_{IBE,T})$ and $ST_{ID|_{k-1}} = (\mathcal{BT}_{ID|_{k-1}}, \beta_{ID|_{k-1}}, z_{ID|_{k-1}})$ with $k \geq 1$.

1. It first obtains a randomized decryption key $RDK_{ID|_{k-1},T} = (RSK_{HIBE}, RSK_{IBE})$ by running **RHIBE-SD.RandDK**$(DK_{ID|_{k-1},T}, -\beta_{ID|_{k-1}}, PP)$.
2. It derives the set R of revoked identities at time T from $RL_{ID|_{k-1}}$. Next, it obtains a covering set $CV_R = \{S_{i,j}\}$ by running **SD.Cover**$(\mathcal{BT}_{ID|_{k-1}}, R)$.
3. For each $S_{i,j} \in CV_R$, it defines $f_{GL}(x) = a_{GL}x + \beta_{ID|_{k-1}}$ by computing $a_{GL} = \mathbf{PRF}(z_{ID|_{k-1}}, GL)$ where $(L_i, L_j) = \mathbf{Label}(S_{i,j})$, $d_j = \mathbf{Depth}(S_j)$, and $GL = L_i \| d_j$, and then it obtains $SK_{IBE,S_{i,j}}$ by running **IBE.GenKey**$(T, f_{GL}(L_j), PP)$.
4. Finally, it outputs an update key $UK_{ID|_{k-1},T,R} = \big(RDK_{ID|_{k-1},T}, CV_R, \{SK_{IBE,S_{i,j}}\}_{S_{i,j} \in CV_R}\big)$. Note that the master key parts of RSK_{HIBE}, RSK_{IBE}, and SK_{IBE,S_i} are η', $\alpha - \eta' - \beta_{ID|_{k-1}}$, and $f_{GL}(L_j)$ for some random η' respectively.

RHIBE-SD.DeriveKey$(ID|_k, T, SK_{ID|_k}, UK_{ID|_{k-1},T,R}, PP)$**:** Let $ID|_k = (I_1, \ldots, I_k)$ with $k \geq 0$, $SK_{ID|_k} = (PV_{ID|_k}, \{SK_{HIBE,S_{i,j}}\}_{S_{i,j} \in PV_{ID|_k}})$, and $UK_{ID|_{k-1},T,R} = (RDK_{ID|_{k-1},T}, CV_R, \{SK_{IBE,S_{i,j}}\}_{S_{i,j} \in CV_R})$ where $RDK_{ID|_{k-1},T} = (RSK'_{HIBE,ID|_{k-1}}, RSK'_{IBE,T})$.

If $k = 0$, then $SK_{ID|_0} = MK$ and UK is empty. It proceeds as follows:

1. It selects a random exponent $\eta \in \mathbb{Z}_p$. It then obtains $RSK_{HIBE,ID|_0}$ and $RSK_{IBE,T}$ by running **HIBE.GenKey**$(ID|_0, \eta, PP)$ and **IBE.GenKey**$(T, MK - \eta, PP)$ respectively.
2. It outputs a decryption key $DK_{ID|_0,T} = (RSK_{HIBE,ID|_0}, RSK_{IBE,T})$.

If $k \geq 1$, then it proceeds as follows:

1. If $ID|_k \notin R$, then it obtains $(S_{i,j}, S_{i',j'})$ by running **SD.Match**$(CV_R, PV_{ID|_k})$. Otherwise, it outputs $\perp$. Next, it retrieves $SK_{HIBE,S_{i',j'}}$ from $SK_{ID|_k}$ and $SK_{IBE,S_{i,j}}$ from $UK_{ID|_{k-1},T,R}$.
2. It sets $I = \{L_j, L_{j'}\}$ and calculates two Lagrange coefficients $\Delta_{L_j,I}(0)$ and $\Delta_{L_{j'},I}(0)$ by using the fact $L_j \neq L_{j'}$. It obtains TSK_{HIBE} and TSK_{IBE} by running **HIBE.ChangeKey**$(SK_{HIBE,S_{i',j'}}, (\times, \Delta_{L_{j'},I}(0)), PP)$ and **IBE.ChangeKey**$(SK_{IBE,S_{i,j}}, (\times, \Delta_{L_j,I}(0)), PP)$ respectively.
3. It obtains $RSK''_{HIBE,ID|_k}$ by running **HIBE.Delegate**$(ID|_k, RSK'_{HIBE,ID|_{k-1}}, PP)$. It selects a random exponent $\eta \in \mathbb{Z}_p$. Next, it obtains $RSK_{HIBE,ID|_k}$ and $RSK_{IBE,T}$ by running **HIBE.MergeKey** $(RSK''_{HIBE}, TSK_{HIBE}, \eta, PP)$ and **IBE.MergeKey**$(RSK'_{IBE}, TSK_{IBE}, -\eta, PP)$ respectively.
4. Finally, it outputs a decryption key $DK_{ID|_k,T} = \big(RSK_{HIBE,ID|_k}, RSK_{IBE,T}\big)$.

Note that the master key parts of $RSK_{HIBE,ID|_k}$ and $RSK_{IBE,T}$ are η' and $\alpha - \eta'$ for some random η' respectively.

RHIBE-SD.RandDK$(DK_{ID|_k,T}, \beta, PP)$**:** It is the same as the randomization algorithm in Section 4.2.

RHIBE-SD.Encrypt$(ID|_\ell, T, M, PP)$**:** It is the same as the encryption algorithm in Section 4.2.

RHIBE-SD.Decrypt$(CT_{ID|_\ell,T}, DK_{ID'|_k,T'}, PP)$**:** It is the same as the decryption algorithm in Section 4.2.

RHIBE-SD.Revoke$(ID|_k, T, RL_{ID|_{k-1}}, ST_{ID|_{k-1}})$**:** It is the same as the revocation algorithm in Section 4.2.

5.3 Security Analysis

We also use the dual system encryption proof technique of Lewko and Waters [18] to prove the adaptive security of our RHIBE-SD scheme. The overall strategy of this security proof is somewhat similar to that of our RHIBE-CS scheme, but we use a different grouping method of small component keys because of the difference between the CS scheme and the SD scheme. The details of the security proof are given as follows.

Theorem 5.2. *The above RHIBE-SD scheme is AD-IND-CPA secure if the SD, GSD, and ComDH assumptions hold.*

Proof. We first define the semi-functional types of private keys, update keys, decryption keys, and ciphertexts in RHIBE by using the semi-functional types of HIBE and IBE in Theorem 4.2. For the semi-functional type, we let g_2 denote a fixed generator of the subgroup $\mathbb{G}_{p_2}$.

RHIBE-SD.GenKeySF. It first creates a normal private key $SK'_{ID|_k} = (PV_{ID|_k}, \{SK'_{HIBE,S_{i,j}}\}_{S_{i,j} \in PV_{ID|_k}})$ by using MK where $PV_{ID|_k} = \{S_{i,j}\}$. For each $S_{i,j} \in PV_{ID|_k}$, it chooses a random exponent $\delta_{i,j} \in \mathbb{Z}_N$ once for $S_{i,j} \in \mathcal{BT}_{ID|_{k-1}}$ and converts a normal $SK'_{HIBE,S_{i,j}}$ to a semi-functional $SK_{HIBE,S_{i,j}}$ with the exponent $\delta_{i,j}$. It outputs a semi-functional private key $SK_{ID|_k} = (PV_{ID|_k}, \{SK_{HIBE,S_{i,j}}\}_{S_{i,j} \in PV_{ID|_k}})$.

RHIBE-SD.UpdateKeySF. It first creates a normal update key $UK'_{ID|_{k-1},T,R} = (RDK'_{ID|_{k-1},T}, CV_R, \{SK'_{IBE,S_{i,j}}\}_{S_{i,j} \in CV_R})$ by using MK. Let $RDK'_{ID|_{k-1},T} = (RSK'_{HIBE,ID|_{k-1}}, RSK'_{IBE,T})$. It chooses random exponents $a_0, a_{ID|_{k-1}} \in \mathbb{Z}_N$ where $a_{ID|_{k-1}}$ is fixed for $\mathcal{BT}_{ID|_{k-1}}$. It converts a normal $RSK'_{HIBE,ID|_{k-1}}$ to a semi-functional type-2 $RSK_{HIBE,ID|_{k-1}}$ with the exponent a_0. It also converts a normal $RSK'_{IBE,T}$ to a semi-functional type-2 $RSK_{IBE,T}$ with the exponent $a_{ID|_{k-1}}$. It sets a semi-functional $RDK_{ID|_{k-1},T} = (RSK_{HIBE,ID|_{k-1}}, RSK_{IBE,T})$. For each $S_{i,j} \in CV_R$, it chooses a random exponent $\delta_{i,j} \in \mathbb{Z}_N$ once for $S_{i,j}$ and converts a normal $SK'_{IBE,S_{i,j}}$ to a semi-functional $SK_{IBE,S_{i,j}}$ with the exponent $\delta_{i,j}$. It outputs a semi-functional update key $UK_{ID|_{k-1},T,R} = (RDK_{ID|_{k-1},T}, CV_R, \{SK_{IBE,S_{i,j}}\}_{S_{i,j} \in CV_R})$.

RHIBE-SD.DeriveKeySF. It first creates a normal decryption key $DK'_{ID|_k,T} = (RSK'_{HIBE,ID|_k}, RSK'_{IBE,T})$ by using MK. It chooses random exponents $a_0, a_1 \in \mathbb{Z}_N$. It converts a normal $RSK'_{HIBE,ID|_k}$ to a semi-functional type-2 $RSK_{HIBE,ID|_k}$ with the exponent a_0. It also converts a normal $RSK'_{IBE,T}$ to a a semi-functional type-2 $RSK_{IBE,T}$ with the exponent a_1. It outputs a semi-functional decryption key $DK_{ID|_k,T} = \big(RSK_{HIBE,ID|_k}, RSK_{IBE,T}\big)$.

RHIBE-SD.EncryptSF. It first creates a normal ciphertext $CT'_{ID|_\ell,T} = (CH'_{HIBE,ID|_\ell}, CH'_{IBE,T}, C')$. It chooses random exponents $c, d_0, d_1 \in \mathbb{Z}_N$. It converts a normal $CH'_{HIBE,ID|_\ell}$ to a semi-functional $CH_{HIBE,ID|_\ell}$ with exponents c, d_0. It also converts a normal $CH'_{IBE,T}$ to a semi-functional $CH_{IBE,T}$ with exponents c, d_1. It outputs a semi-functional ciphertext $CT_{ID|_\ell,T} = \big(CH_{HIBE,ID|_\ell}, CH_{IBE,T}, C'\big)$.

The security proof consists of the sequence of hybrid games $\mathbf{G}_0, \mathbf{G}_1, \ldots, \mathbf{G}_5$. The first game $\mathbf{G}_0$ is the original security game and the last one $\mathbf{G}_5$ is a game such that the adversary has no advantage. We omit the definition of these games since these are the same as those in Theorem 4.2.

Let $\mathbf{Adv}_{\mathcal{A}}^{G_j}$ be the advantage of $\mathcal{A}$ in the game $\mathbf{G}_j$. We have that $\mathbf{Adv}_{RIBE,\mathcal{A}}^{AD\text{-}IND\text{-}CPA}(1^\lambda) = \mathbf{Adv}_{\mathcal{A}}^{G_0}$ and

$\mathbf{Adv}_{\mathcal{A}}^{G_5} = 0$. From the following Lemmas 5.3, 5.4, 5.5, 5.8, and 5.9, we obtain the following equation

$$\begin{aligned}\mathbf{Adv}_{RIBE,\mathcal{A}}^{AD\text{-}IND\text{-}CPA}(1^\lambda) &\leq \sum_{j=1}^{5} \left|\mathbf{Adv}_{\mathcal{A}}^{G_{j-1}} - \mathbf{Adv}_{\mathcal{A}}^{G_j}\right| \\ &\leq O(qL)\mathbf{Adv}_{\mathcal{B}}^{PRF}(1^\lambda) + \mathbf{Adv}_{\mathcal{B}}^{SD}(1^\lambda) + O\big(q^2 \log^2 N_{max} + q^2 r_{max}\big)\mathbf{Adv}_{\mathcal{B}}^{GSD}(1^\lambda) + \\ &\quad \mathbf{Adv}_{\mathcal{B}}^{ComDH}(1^\lambda).\end{aligned}$$

This completes our proof. □

Lemma 5.3. *If the PRF is secure, then no polynomial-time adversary can distinguish* $\mathbf{G}_0$ *from* $\mathbf{G}_1$ *with a non-negligible advantage.*

Lemma 5.4. *If the SD assumption holds, then no polynomial-time adversary can distinguish* $\mathbf{G}_1$ *from* $\mathbf{G}_2$ *with a non-negligible advantage.*

The proofs of Lemmas 5.3 and 5.4 are the same as those of Lemmas 4.3 and 4.4.

Lemma 5.5. *If the GSD assumption holds, then no polynomial-time adversary can distinguish* $\mathbf{G}_2$ *from* $\mathbf{G}_3$ *with a non-negligible advantage.*

Proof. For the proof of this lemma, we cannot use simple hybrid games that change a normal private key (or normal update key) to a semi-functional private key (or semi-functional update key) one by one since the adversary of RHIBE can query a private key for $ID|_k \in \mathbf{Prefix}(ID^*|_\ell)$ and an update key for T^*. Note that these normal keys cannot directly converted to semi-functional keys since an information theoretic argument cannot be used.

To solve this problem, we first divide each private key and update key into small HIBE private keys and IBE private keys. Recall that a private key $SK_{ID|_k}$ consists of many HIBE private keys and an update key $UK_{ID|_{k-1},T,R}$ consists of a randomized decryption key and many IBE private keys where each HIBE private key (or an IBE private key) is associated with a subset $S_{i,j}$ in $\mathcal{BT}_{ID|_{k-1}}$. Next, HIBE private keys and IBE private keys that are related to the same group of a subset $S_{i,j}$ in $\mathcal{BT}_{ID|_{k-1}}$ are grouped together. To uniquely identify the group of a subset $S_{i,j} \in \mathcal{BT}_{ID|_{k-1}}$, we define a group identifier GID of this subset as a string $ID|_{k-1}\|L_i\|d_j$ where $(L_i, L_j) = \mathbf{Label}(S_{i,j})$ and $d_j = \mathbf{Depth}(S_j)$. To prove this lemma, we change normal HIBE private keys and normal IBE private keys that are related to the same group identifier into semi-functional keys by defining additional hybrid games. This additional hybrid games are performed for all group identifiers that are used in the key queries of the adversary.

For additional hybrid games that change HIBE private keys (or IBE private keys) that are related to the same group identifier $GID = ID|_{k-1}\|L_i\|d_j$ from normal keys to semi-functional keys, we need to state additional information of a subset $S_{i,j}$ in $\mathcal{BT}_{ID|_{k-1}}$. Note that an HIBE private key for $S_{i,j}$ and an IBE private key for $S_{i',j'}$ share the same polynomial $f(x)$ if $(L_i = L_{i'}) \wedge (d_j = d_{j'})$ since they belong to the same group. Thus we associate an HIBE private key (or an IBE private key) with an index pair (i_g, i_m, i_c) to state additional information where i_g is a group index, i_m is a member index, and i_c is a counter index.

Suppose that an HIBE private key (or an IBE private key) is related with a subset $S_{i,j}$, Then this key has a group identifier $GID = ID|_{k-1}\|L_i\|d_j$ and a member label L_j. The group index i_g for HIBE private keys (or IBE private keys) is assigned as follows: If the group identifier GID appears first time in queries, then we set i_g as the number of distinct group identifiers in previous queries plus one. If the group identifier GID already appeared before in queries, then we set i_g as the value i'_g of previous HIBE private key (or IBE private key) with the same group identifier GID. The member index i_m for the group index i_g is assigned as follows: If the member label L_j for this group identifier GID appears first time in queries, then we set i_m as the number of distinct members for this group identifier GID in previous queries plus one. If the member label L_j for this group identifier already appeared before in queries, then we set i_m as the value i'_m of previous one. The counter index i_c is assigned as follows: If the group identifier and member label (GID, L_j) appears first time in queries, then we set i_c as one. If the group identifier and member label (GID, L_j) appeared before in queries, then we set i_c as the number of queries with the group identifier and member label (GID, L_j) that appeared before plus one.

For the security proof, we additionally define a sequence of games $\mathbf{G}_{2,1}, \ldots, \mathbf{G}_{2,h}, \ldots, \mathbf{G}_{2,q_g}$ where $\mathbf{G}_2 = \mathbf{G}_{2,0}$ and q_g is the maximum number of group identifiers that are used in private keys and update keys. In the game $\mathbf{G}_{2,h}$ for $1 \leq h \leq q_g$, the challenge ciphertext is semi-functional, HIBE private keys and IBE private keys with a group identifier $i_g \leq h$ are semi-functional, the remaining HIBE private keys and IBE private keys with a group index $i_g > h$ are normal, and all randomized decryption keys are still normal. In the game $\mathbf{G}_3$, the remaining randomized decryption keys are changed to be semi-functional.

Let $\mathbf{Adv}_{\mathcal{A}}^{G_j}$ be the advantage of $\mathcal{A}$ in the game $\mathbf{G}_j$. From the following Lemmas 5.6 and 5.7, we have the following equation

$$\begin{aligned}\mathbf{Adv}_{\mathcal{A}}^{G_2} - \mathbf{Adv}_{\mathcal{A}}^{G_3} &\leq \sum_{h=1}^{q_g} \left|\mathbf{Adv}_{\mathcal{A}}^{G_{2,h-1}} - \mathbf{Adv}_{\mathcal{A}}^{G_{2,h}}\right| + \left|\mathbf{Adv}_{\mathcal{A}}^{G_{2,q_g}} - \mathbf{Adv}_{\mathcal{A}}^{G_3}\right| \\ &\leq O(q^2 \log^2 N_{max} + q^2 r_{max}) \mathbf{Adv}_{\mathcal{B}}^{GSD}(1^\lambda).\end{aligned}$$

This completes the proof. □

Lemma 5.6. *If the GSD assumption holds, then no polynomial-time adversary can distinguish $\mathbf{G}_{2,h-1}$ from $\mathbf{G}_{2,h}$ with a non-negligible advantage.*

Proof. We first divide the behavior of an adversary as two types: Type-I and Type-II. We next show that this lemma holds for two types of the adversary. Let $ID^*|_\ell$ be the challenge hierarchical identity and T^* be the challenge time respectively. The two types of adversaries are formally defined as follows:

Type-I. An adversary is Type-I if it queries on a hierarchical identity $ID|_k \in \mathbf{Prefix}(ID^*|_\ell)$ for at least one HIBE private key with the group index h, or it queries on time $T = T^*$ for at least one IBE private key with the group index h. More specifically, this adversary can be divided as follows:

- Type-I-A. This adversary queries on a hierarchical identity $ID|_k \notin \mathbf{Prefix}(ID^*|_\ell)$ for all HIBE private keys with the group index h, and it queries on time $T = T^*$ for at least one IBE private key with the group index h.
- Type-I-B. This adversary queries on time $T \neq T^*$ for all IBE private keys with h, and it queries on a hierarchical identity $ID|_k \in \mathbf{Prefix}(ID^*|_\ell)$ for at least one HIBE private key with h.
- Type-I-C. This adversary queries on a hierarchical identity $ID|_k \in \mathbf{Prefix}(ID^*|_\ell)$ for at least one HIBE private key with h, and it queries on time $T = T^*$ for at least one IBE private key with h.

Type-II. An adversary is Type-II if it queries on a hierarchical identity $ID|_k \notin \mathbf{Prefix}(ID^*|_\ell)$ for all HIBE private keys with the group index h, and it queries on time $T \neq T^*$ for all IBE private keys with the group index h.

Let's assume that a group index h for this game is defined in $\mathcal{BT}_{ID|_{k-1}}$. Let CV_{R^*} be the covering set of an update key for the challenge time T^* and the revoked set R^* at time T^*, and $PV_{ID^*|_k}$ be the private set of an private key for an hierarchical identity $ID^*|_k \in \mathbf{Prefix}(ID^*|_\ell)$. Let h_m^* be a member index of the group index h such that the HIBE private key for $ID^*|_k$ or the IBE private key for T^* belong to the member index h_m^*. If the adversary is Type-I-A, then there is only one member index h_m^* since CV_{R^*} is a partition. If the adversary is In Type-I-B, then there is only one member index h_m^* since $PV_{ID^*|_k}$ is related with a path. If the adversary is Type-I-C, the member index h_m^* of CV_{R^*} with the group index h should be the same as that of $PV_{ID^*|_k}$ with the same group index h in the SD scheme if $ID^*|_k \in R^*$. If the adversary is Type-II, then there is no member index h_m^* since the adversary does not request a key query for $ID^*|_k$ or T^*.

For the Type-I adversary $\mathcal{A}_I$, we define hybrid games $\mathbf{H}_{(1,1),1}, \mathbf{H}_{(1,1),2}, \ldots, \mathbf{H}_{(q_m,q_c),1}, \mathbf{H}_{(q_m,q_c),2} = \mathbf{H}'_{(q_m,q_c),2}$, $\mathbf{H}'_{(q_m,q_c),1}, \ldots, \mathbf{H}'_{(1,1),2}, \mathbf{H}'_{(1,1),1}, \mathbf{H}'_{(1,0),2}, \mathbf{H}''$ where $\mathbf{G}_{2,h-1} = \mathbf{H}_{(1,0),2}$, $\mathbf{H}'' = \mathbf{G}_{2,h}$, q_m is the maximum number of distinct member subsets of the group index h, and q_c is the maximum number of queries for one member subset. The games are formally defined as follows:

Game $\mathbf{H}_{(h_m,h_c),1}$. This game $\mathbf{H}_{(h_m,h_c),1}$ for $1 \le h_m \le q_m$ and $1 \le h_c \le q_c$ is almost the same as $\mathbf{G}_{2,h-1}$ except the generation of HIBE private keys and IBE private keys with the group index h. These HIBE private keys and IBE private keys with indexes $(i_g = h, i_m, i_c)$ are generated as follows:

- **Case** $i_g < h$: The keys (HIBE private keys and IBE private keys) are generated as semi-functional.
- **Case** $i_g = h$: The keys are generated as follows:
  - $(i_m \ne h_m^*) \wedge (i_m < h_m)$ or $(i_m \ne h_m^*) \wedge (i_m = h_m) \wedge (i_c < h_c)$:
    If this is an HIBE private key query, then it generates a normal $SK'_{HIBE,S_{i,j}}$ and converts the key to a semi-functional type-2 $SK_{HIBE,S_{i,j}}$ by selecting a new random exponent $a_0 \in \mathbb{Z}_N$.
    If this is an IBE private key query, then it generates a normal $SK'_{IBE,S_{i,j}}$ and converts the key to a semi-functional type-2 $SK_{IBE,S_{i,j}}$ by selecting a new random exponent $a_1 \in \mathbb{Z}_N$.
  - $(i_m \ne h_m^*) \wedge (i_m = h_m) \wedge (i_c = h_c)$:
    If this is an HIBE private key query, then it generates a normal $SK'_{HIBE,S_{i,j}}$ and converts the key to a semi-functional type-1 $SK_{HIBE,S_{i,j}}$ by selecting new random exponents $a_0, b_0, \{z_i\} \in \mathbb{Z}_N$.
    If this is an IBE private key query, then it generates a normal $SK'_{IBE,S_{i,j}}$ and converts the key to a semi-functional type-1 $SK_{IBE,S_{i,j}}$ by selecting new random exponents $a_1, b_1 \in \mathbb{Z}_N$.
  - $(i_m \ne h_m^*) \wedge (i_m = h_m) \wedge (h_c < i_c)$ or $(i_m \ne h_m^*) \wedge (h_m < i_m)$: It simply creates a normal key.
  - $(i_m = h_m^*)$: It simply creates a normal key.
- **Case** $i_g > h$: The keys are generated as normal.

Recall that if $a_0 + \sum_{i=k+1}^{\ell} b_0 z_i I_i = b_0 d_0$, then this HIBE private key is *nominally* semi-functional type-1. Similarly, if $a_1 = b_1 d_1$, then this IBE private key is *nominally* semi-functional type-1.

Game $\mathbf{H}_{(h_m,h_c),2}$. This game $\mathbf{H}_{(h_m,h_c),2}$ is almost the same as $\mathbf{H}_{(h_m,h_c),1}$ except that the HIBE private key (or the IBE private key) with the indexes $(i_g = h, i_m, i_c)$ such that $(i_m \ne h_m^*) \wedge (i_m = h_m) \wedge (i_c = h_c)$ is generated with $b_0 = b_1 = 0$. In the game $\mathbf{H}_{(q_m,q_c),2}$, all HIBE private keys and IBE private keys with the group index h are semi-functional type-2 except that HIBE private keys and IBE private keys with the member index h_m^* are normal.

Game $\mathbf{H}'_{(h_m,h_c),1}$. This game $\mathbf{H}'_{(h_m,h_c),1}$ is almost the same as $\mathbf{H}_{(h_m,h_c),1}$ except the generation of an HIBE private key (or an IBE private key) with the indexes $(i_g = h, i_m, i_c)$ such that $(i_m \ne h_m^*) \wedge (i_m = h_m) \wedge (h_c \le i_c)$ or $(i_m \ne h_m) \wedge (h_m < i_m)$. These HIBE private keys (or IBE private keys) are generated as follows:

- $(i_m \neq h_m^*) \wedge (i_m = h_m) \wedge (i_c = h_c)$: Let $\delta_{i,j}$ be a random exponent in $\mathbb{Z}_N$ that is fixed for this member subset $S_{i,j}$.
  If this is an HIBE private key query, then it generates $SK''_{HIBE,S_{i,j}} = (K''_0, K''_1, \{K''_{2,i}\})$ as the same as $\mathbf{H}_{(h_m,h_c),1}$ and creates a semi-functional HIBE private key $SK_{ID,S_{i,j}} = \big(K_0 = K''_0 g_2^{\delta_{i,j}}, K_1 = K''_1, \{K_{2,i} = K''_{2,i}\}\big)$.
  If this is an IBE private key query, then it generates $SK''_{IBE,S_{i,j}} = (K''_0, K''_1)$ as the same as $\mathbf{H}_{(h_m,h_c),1}$ and creates a semi-functional IBE private key $SK_{IBE,S_{i,j}} = \big(K_0 = K''_0 g_2^{\delta_{i,j}}, K_1 = K''_1\big)$.
- $(i_m \neq h_m^*) \wedge (i_m = h_m) \wedge (h_c < i_c)$ or $(i_m \neq h_m^*) \wedge (h_m < i_m)$: It creates a semi-functional key by using the fixed $\delta_{i,j}$ for this member subset $S_{i,j}$.

Game $\mathbf{H}'_{(h_m,h_c),2}$. This game $\mathbf{H}'_{(h_m,h_c),2}$ is almost the same as $\mathbf{H}'_{(h_m,h_c),1}$ except that the HIBE private key or IBE private key with the indexes $(i_g = h, i_m, i_c)$ such that $(i_m \neq h_m^*) \wedge (i_m = h_m) \wedge (i_c = h_c)$ is generated with $b_0 = b_1 = 0$. The modification is similar to the game $\mathbf{H}'_{(h_m,h_c),1}$. In the game $\mathbf{H}'_{(1,0),2}$, all HIBE private keys and all IBE private keys with the group index h except the keys with the member index h_m^* are semi-functional where a fixed $\delta_{i,j}$ is used for each member.

Game $\mathbf{H}''$. This game $\mathbf{H}''$ is the same as $\mathbf{G}_{2,h}$. Compared to the game $\mathbf{H}'_{(1,0),2}$, the remaining HIBE private keys and IBE private keys with the member index h_m^* are changed to be semi-functional by using a fixed $\delta_{i,j}$ for this member subset $S_{i,j}$.

Let $\mathbf{Adv}_{\mathcal{A}_I}^{H_i}$ be the advantage of $\mathcal{A}_I$ in a game $\mathbf{H}_i$. From the following Lemmas 5.10, 5.11, 5.12, 5.13, and 5.14, we obtain the following equation

$$\begin{aligned}
&\mathbf{Adv}_{\mathcal{A}_I}^{H_{(1,0),2}} - \mathbf{Adv}_{\mathcal{A}_I}^{H''} \\
&\leq \sum_{h_m=1}^{q_m}\sum_{h_c=1}^{q_c} \big|\mathbf{Adv}_{\mathcal{A}_I}^{H_{(h_m,h_c-1),2}} - \mathbf{Adv}_{\mathcal{A}_I}^{H_{(h_m,h_c),1}}\big| + \sum_{h_m=1}^{q_m}\sum_{h_c=1}^{q_c} \big|\mathbf{Adv}_{\mathcal{A}_I}^{H_{(h_m,h_c),1}} - \mathbf{Adv}_{\mathcal{A}_I}^{H_{(h_m,h_c),2}}\big| + \\
&\quad \sum_{h_m=1}^{q_m}\sum_{h_c=1}^{q_c} \big|\mathbf{Adv}_{\mathcal{A}_I}^{H'_{(h_m,h_c),2}} - \mathbf{Adv}_{\mathcal{A}_I}^{H'_{(h_m,h_c),1}}\big| + \sum_{h_m=1}^{q_m}\sum_{h_c=1}^{q_c} \big|\mathbf{Adv}_{\mathcal{A}_I}^{H'_{(h_m,h_c),1}} - \mathbf{Adv}_{\mathcal{A}_I}^{H'_{(h_m,h_c-1),2}}\big| + \\
&\quad \big|\mathbf{Adv}_{\mathcal{A}_I}^{H'_{(1,0),2}} - \mathbf{Adv}_{\mathcal{A}_I}^{H''}\big| \\
&\leq O(q)\sum_{h_m=1}^{q_m}\sum_{h_c=1}^{q_c} \mathbf{Adv}_{\mathcal{B}}^{GSD}(1^\lambda).
\end{aligned}$$

For the Type-II adversary $\mathcal{A}_{II}$, we define hybrid games $\mathbf{I}_{(1,1),1}, \mathbf{I}_{(1,1),2}, \ldots, \mathbf{I}_{(q_m,q_c),1}, \mathbf{I}_{(q_m,q_c),2} = \mathbf{I}'_{(q_m,q_c),2}, \mathbf{I}'_{(q_m,q_c),1}, \ldots, \mathbf{I}'_{(1,1),2}, \mathbf{I}'_{(1,1),1}, \mathbf{I}'_{(1,0),2}, \mathbf{I}''$ where $\mathbf{G}_{2,h-1} = \mathbf{I}_{(1,0),2}$ and $\mathbf{I}'_{(1,0),2} = \mathbf{I}'' = \mathbf{G}_{2,h}$. The games are formally defined as follows:

Game $\mathbf{I}_{(h_m,h_c),1}$. This game $\mathbf{I}_{(h_m,h_c),1}$ is almost the same as $\mathbf{H}_{(h_m,h_c),1}$ except that there is no case $i_m = h_m^*$ since the adversary is Type-II.

Game $\mathbf{I}_{(h_m,h_c),2}$. This game $\mathbf{I}_{(h_m,h_c),2}$ is almost the same as $\mathbf{H}_{(h_m,h_c),2}$ except that there is no case $i_m = h_m^*$ since the adversary is Type-II.

Game $\mathbf{I}'_{(h_m,h_c),1}$. This game $\mathbf{I}'_{(h_m,h_c),1}$ is almost the same as $\mathbf{H}'_{(h_m,h_c),1}$ except that there is no case $i_m = h_m^*$ since the adversary is Type-II.

Game $\mathbf{I}'_{(h_m,h_c),2}$. This game $\mathbf{I}'_{(h_m,h_c),1}$ is almost the same as $\mathbf{H}'_{(h_m,h_c),1}$ except that there is no case $i_m = h_m^*$ since the adversary is Type-II. In the game $\mathbf{I}'_{(1,0),2}$, all HIBE private keys and all IBE private keys with the group index h are semi-functional where a fixed $\delta_{i,j}$ is used for each member.

Let $\mathbf{Adv}^{I_i}_{\mathcal{A}_{II}}$ be the advantage of $\mathcal{A}_{II}$ in a game $\mathbf{I}_i$. From the following Lemmas 5.15, 5.16, 5.17, and 5.18, we can obtain the equation $\mathbf{Adv}^{I_{(1,0),2}}_{\mathcal{A}_{II}} - \mathbf{Adv}^{I''}_{\mathcal{A}_{II}} \leq O(q)\sum_{h_m=1}^{q_m}\sum_{h_c=1}^{q_c}\mathbf{Adv}^{GSD}_{\mathcal{B}}(1^\lambda)$.

Let E_I, E_{II} be the event such that an adversary behave like the Type-I, Type-II adversary respectively. From the above three inequalities for three types, we have the following inequality

$$\begin{aligned}\mathbf{Adv}^{G_{2,h-1}}_{\mathcal{A}} - \mathbf{Adv}^{G_{2,h}}_{\mathcal{A}} &\leq \Pr[E_I](\mathbf{Adv}^{H_{(1,0),2}}_{\mathcal{A}_I} - \mathbf{Adv}^{H''}_{\mathcal{A}_I}) + \Pr[E_{II}](\mathbf{Adv}^{I_{(1,0),2}}_{\mathcal{A}_{II}} - \mathbf{Adv}^{I''}_{\mathcal{A}_{II}}) \\ &\leq O(q)\sum_{h_m=1}^{q_m}\sum_{h_c=1}^{q_c}\mathbf{Adv}^{GSD}_{\mathcal{B}}(1^\lambda).\end{aligned}$$

This completes our proof. □

Lemma 5.7. *If the GSD assumption holds, then no polynomial-time adversary can distinguish $\boldsymbol{G}_{2,q_g}$ from $\boldsymbol{G}_3$ with a non-negligible advantage.*

Lemma 5.8. *If the GSD assumption holds, then no polynomial-time adversary can distinguish $\boldsymbol{G}_3$ from $\boldsymbol{G}_4$ with a non-negligible advantage.*

Lemma 5.9. *If the ComDH assumption holds, then no polynomial-time adversary can distinguish $\boldsymbol{G}_4$ from $\boldsymbol{G}_5$ with a non-negligible advantage.*

The proofs of Lemmas 5.7, 5.8 and 5.9 are the same as those of Lemmas 4.7, 4.8 and 4.11.

5.3.1 Type-I Adversary

Lemma 5.10. *If the GSD assumption holds, then no polynomial-time Type-I adversary can distinguish $\boldsymbol{H}_{(h_m,h_c-1),2}$ from $\boldsymbol{H}_{(h_m,h_c),1}$ with a non-negligible advantage.*

Proof. Suppose there exists an adversary $\mathcal{A}_I$ that distinguishes $\mathbf{H}_{(h_m,h_c-1),2}$ from $\mathbf{H}_{(h_m,h_c),1}$ with a non-negligible advantage. A simulator $\mathcal{B}$ that solves the GSD assumption using $\mathcal{A}_I$ is given: a challenge tuple $D = ((N,\mathbb{G},\mathbb{G}_T,e), g_{p_1}, g_{p_3}, X_1R_1, R_2Y_1)$ and Z where $Z = Z_0 = X_2Y_2$ or $Z = Z_1 = X_2R_3Y_2$. Then $\mathcal{B}$ that interacts with $\mathcal{A}_I$ is described as follows:

Setup: $\mathcal{B}$ first chooses random exponents $h', u'_1, \ldots, u'_L, v', w', \alpha \in \mathbb{Z}_N$ and random elements $\Lambda_1, \Lambda_2 \in \mathbb{G}_{p_3}$. Next, it builds $PP_{HIBE} = \big((N,\mathbb{G},\mathbb{G}_T,e), g = g_{p_1}, Y = g_{p_3}, h = g_{p_1}^{h'}, u_1 = g_{p_1}^{u'_1}, \ldots, u_L = g_{p_1}^{u'_L}, \Lambda_1\big)$ and $PP_{IBE} = \big((N,\mathbb{G},\mathbb{G}_T,e), g = g_{p_1}, Y = g_{p_3}, v = g_{p_1}^{v'}, w = g_{p_1}^{w'}, \Lambda_2\big)$. It sets $MK = \alpha$ and publishes $PP = \big(PP_{HIBE}, PP_{IBE}, \Omega = e(g,g)^\alpha\big)$.

Phase 1: Let h_m^* be a member index of the group index h such that the HIBE private key for $ID^*|_k$ or the IBE private key for T^* belong to the member index h_m^* such that $1 \leq h_m^* \leq q_m$ where q_m is the maximum number of members in the group index h. As mentioned before, there is only one index h_m^* in the Type-I adversary. By randomly selecting an index, $\mathcal{B}$ can correctly guess h_m^* with the probability of $1/q_m$. Note that $q_m \leq q_{sk} + q_{uk} \leq q$ since the private set of a private key is related with a path and the covering set of an update key is a partition where q_{sk} is the number of private key queries and q_{uk} is the number of update key queries of the adversary.

For each query, $\mathcal{B}$ proceeds as follows: If this is a decryption key query, then it creates a normal one since it knows MK. If this is an HIBE private key or an IBE private key query with indexes (i_g, i_m, i_c), then it handles this query as follows:

- **Case** $i_g < h$: It first builds a normal key by using MK and converts the key to a semi-functional one with a fixed random exponent $\delta'_{i,j}$ for the subset $S_{i,j}$ by using R_2Y_1.
- **Case** $i_g = h$: It generates the key as follows:
  - $(i_m \neq h^*_m) \wedge (i_m < h_m)$ or $(i_m \neq h^*_m) \wedge (i_m = h_m) \wedge (i_c < h_c)$:
    If this is an HIBE private key query, then it builds a normal key and converts the key to a semi-functional type-2 $SK_{HIBE,S_{i,j}}$ with a new random exponent $a'_0 \in \mathbb{Z}_N$ by using R_2Y_1.
    If this is an IBE private key query, then it builds a normal key and converts the key to a semi-functional type-2 $SK_{IBE,S_{i,j}}$ with a new random exponent $a'_1 \in \mathbb{Z}_N$ by using R_2Y_1.
  - $(i_m \neq h^*_m) \wedge (i_m = h_m) \wedge (i_c = h_c)$:
    If this is an HIBE private key query, then it chooses random elements $Y'_0, Y'_1, \{Y'_{2,i}\} \in \mathbb{G}_{p_3}$ and creates an HIBE private key $SK_{HIBE,S_{i,j}} = \big(K_0 = g^{f_{GL}(L_j)}(Z)^{h'+\sum_{i=1}^{k} u'_i I_i} Y'_0,\ K_1 = (Z)^{-1}Y'_1, \{K_{2,i} = (Z)^{u'_i} Y'_{2,i}\}\big)$.
    If this is an IBE private key query, then it chooses random elements $Y'_0, Y'_1 \in \mathbb{G}_{p_3}$ and creates an IBE private key $SK_{IBE,S_{i,j}} = \big(K_0 = g^{f_{GL}(L_j)}(Z)^{v'+w'T} Y'_0,\ K_1 = (Z)^{-1}Y'_1\big)$.
  - $(i_m \neq h^*_m) \wedge (i_m = h_m) \wedge (h_c < i_c)$ or $(i_m \neq h^*_m) \wedge (h_m < i_m)$: It creates a normal key by using MK.
  - $(i_m = h^*_m)$: It creates a normal key by using MK.
- **Case** $i_g > h$: It creates a normal key by using MK.

Challenge: $\mathcal{B}$ first builds $CH_{HIBE,ID^*|_\ell} = \big(C_0 = X_1R_1,\ C_1 = (X_1R_1)^{h'+\sum_{i=1}^{\ell} u'_i I^*_i}\big)$ and $CH_{IBE,T^*} = (C_0 = X_1R_1,\ C_1 = (X_1R_1)^{v'+w'T^*})$. Next, it flips a random coin $\mu \in \{0,1\}$ and creates the semi-functional challenger ciphertext $CT_{ID^*|_\ell,T^*} = (CH_{HIBE,ID^*|_\ell}, CH_{IBE,T^*}, C = e(X_1R_1, g)^\alpha \cdot M^*_\mu)$.
Phase 2: Same as Phase 1.
Guess: $\mathcal{A}$ outputs a guess μ'. If $\mu = \mu'$, then $\mathcal{B}$ outputs 1. Otherwise, it outputs 0.

If $Z = Z_0 = X_2Y_2$, then the simulation is the same as $\mathbf{H}_{(h_m,h_c-1),2}$ since the HIBE private key (or the IBE private key) with $(i_m \neq h^*_m) \wedge (i_m = h_m) \wedge (i_c = h_c)$ and the semi-functional challenge ciphertext are correctly distributed. If $Z = Z_1 = X_2R_3Y_2$, then the simulation is almost the same as $\mathbf{H}_{(h_m,h_c),1}$ except that the HIBE private key (or the IBE private key) with $(i_m \neq h^*_m) \wedge (i_m = h_m) \wedge (i_c = h_c)$ is generated as nominally semi-functional type-1 by implicitly setting $a_0 \equiv \log_{g_2}(R_3)(h' + \sum_{i=1}^{k} u'_i I_i) \mod p_2$ (or $a_1 \equiv \log_{g_2}(R_3)(v' + w'T) \mod p_2$), $b_0 \equiv b_1 \equiv \log_{g_2}(R_3) \mod p_2$, and $z_i \equiv u'_i \mod p_2$. Note that we solve the paradox of dual system encryption by introducing the nominally semi-functional type-1 key.

Next, we should argue that the Type-I adversary cannot distinguish a nominally semi-functional type-1 key from a semi-functional type-1 key. For this argument, we show an information theoretic argument by using the fact that $ID|_k \notin \mathbf{Prefix}(ID^*|_\ell)$ for all HIBE private key queries with indexes $(i_g = h, i_m, i_c)$ such that $i_m \neq h^*_m$, and $T \neq T^*$ for all IBE private key queries with indexes $(i_g = h, i_m, i_c)$ such that $i_m \neq h^*_m$. The analysis of this information theoretic argument is the same as that in Lemma 4.12. This completes our proof. □

Lemma 5.11. *If the GSD assumption holds, then no polynomial-time Type-I adversary can distinguish* $\boldsymbol{H}_{(h_m,h_c),1}$ *from* $\boldsymbol{H}_{(h_m,h_c),2}$ *with a non-negligible advantage.*

Proof. The proof of this lemma is almost the same as that of Lemma 5.10 except the generation of the key with indexes $i_g = h$ and $(i_m \neq h_m^*) \wedge (i_m = h_m) \wedge (i_c = h_c)$. This key with the group index h is generated as follows:

- $(i_m \neq h_m^*) \wedge (i_m = h_m) \wedge (i_c = h_c)$:
  If this is an HIBE private key query, then it chooses random $a_0' \in \mathbb{Z}_N$, $Y_0', Y_1', \{Y_{2,i}'\} \in \mathbb{G}_{p_3}$ and creates an HIBE private key $SK_{HIBE,S_{i,j}} = \big(K_0 = g^{f_{GL}(L_j)}(Z)^{h' + \sum_{i=1}^k u_i' I_i} Y_0' (R_2 Y_1)^{a_0'},\ K_1 = (Z)^{-1} Y_1',\ \{K_{2,i} = (Z)^{u_i'} Y_{2,i}'\}\big)$.
  If this is an IBE private key query, then it chooses random $a_1' \in \mathbb{Z}_N$, $Y_0', Y_1' \in \mathbb{G}_{p_3}$ and creates an IBE private key $SK_{IBE,S_{i,j}} = \big(K_0 = g^{f_{GL}(L_j)}(Z)^{v' + w'T} Y_0' (R_2 Y_1)^{a_1'},\ K_1 = (Z)^{-1} Y_1'\big)$.

Note that this HIBE private key (or IBE private key) is no longer correlated with the challenge ciphertext since K_0 is randomized by $(R_2 Y_1)^{a_0'}$ (or $(R_2 Y_1)^{a_1'}$). □

Lemma 5.12. *If the GSD assumption holds, then no polynomial-time Type-I adversary can distinguish $\boldsymbol{H}'_{(h_m, h_c - 1),2}$ from $\boldsymbol{H}'_{(h_m, h_c),1}$ with a non-negligible advantage.*

Lemma 5.13. *If the GSD assumption holds, then no polynomial-time Type-I adversary can distinguish $\boldsymbol{H}'_{(h_m, h_c),1}$ from $\boldsymbol{H}'_{(h_m, h_c),2}$ with a non-negligible advantage.*

The proofs of Lemmas 5.12 and 5.13 are almost the same as those of Lemmas 5.10 and 5.11 respectively. The only difference is that K_0 of an HIBE private key and K_0 of an IBE private key with indexes $(i_g = h, i_m, i_c)$ such that $i_m \neq h_m^*$ that are generated in Lemmas 5.10 and 5.11 respectively are additionally multiplied by $(R_2 Y_1)^{\delta_{i,j}'}$ where $\delta_{i,j}'$ is a fixed exponent that is related with the member subset $S_{i,j}$. This modification is possible since $R_2 Y_1$ is given in the assumption. We omit the detailed proofs of these lemmas.

Lemma 5.14. *If the GSD assumption holds, then no polynomial-time Type-I adversary can distinguish $\boldsymbol{H}'_{(1,0),2}$ from $\boldsymbol{H}''$ with a non-negligible advantage.*

Proof. The proof of this lemma is the important part of the security proof since it changes the HIBE private key for $ID^*|_k \in \mathbf{Prefix}(ID^*|_\ell)$ and the IBE private key for T^* from a normal type to a semi-functional type. It should be noted that this changes from normal to semi-functional cannot be handled by introducing a nominally semi-functional type since an information theoretic argument for $ID^*|_k$ and T^* cannot be used. Recall that h_m^* be the member index that is related to $ID^*|_k$ and T^*. To solve this problem, we directly change normal keys for h_m^* to semi-functional keys without introducing nominally semi-functional keys, and then we argue that the paradox of dual system encryption can be solved by the property of the Lagrange interpolation method.

Many parts of this proof is similar to that of Lemma 5.10 except the generation of HIBE private keys and IBE private keys with the group index $i_g = h$. These keys with the group index $i_g = h$ are generated as follows:

- **Case** $i_g = h$: Let $\delta_{i,j}'$ be a fixed exponent in $\mathbb{Z}_N$ for each member $S_{i,j}$ in this group index h.
  - $(i_m \neq h_m^*)$:
    If this is an HIBE private key query, then it selects random $r_1 \in \mathbb{Z}_N$, $Y_0', Y_1', \{Y_{2,i}'\} \in \mathbb{G}_{p_3}$ and creates an HIBE private key $SK_{HIBE,S_{i,j}} = \big(K_0 = (Z)^{L_j} g^{\beta_{ID|_{k-1}}} (h \prod_{i=1}^k u_i^{I_i})^{r_1} Y_0' \cdot (R_2 Y_1)^{\delta_{i,j}'},\ K_1 = g^{-r_1} Y_1',\ \{K_{2,i} = u_i^{r_1} Y_{2,i}'\}\big)$.
    If this is an IBE private key query , then it selects random $r_2 \in \mathbb{Z}_N$, $Y_0', Y_1' \in \mathbb{G}_{p_3}$ and creates an IBE private key $SK_{IBE,S_{i,j}} = \big(K_0 = (Z)^{L_j} g^{\beta_{ID|_{k-1}}} (v w^T)^{r_2} Y_0' \cdot (R_2 Y_1)^{\delta_{i,j}'},\ K_1 = g^{-r_2} Y_1'\big)$.

– $(i_m = h^*_m)$:
  If this is an HIBE private key query, then it selects random $r_1 \in \mathbb{Z}_N$, $Y'_0, Y'_1, \{Y'_{2,i}\} \in \mathbb{G}_{p_3}$ and creates an HIBE private key $SK_{HIBE,S_{i,j}} = \big(K_0 = (Z)^{L_j} g^{\beta_{ID|_{k-1}}} (h \prod_{i=1}^{k} u_i^{I_i})^{r_1} Y'_0,\ K_1 = g^{-r_1} Y'_1,\ \{K_{2,i} = u_i^{r_1} Y'_{2,i}\}\big)$.
  If this is an IBE private key query, then it selects random $r_2 \in \mathbb{Z}_N$, $Y'_0, Y'_1 \in \mathbb{G}_{p_3}$ and creates an IBE private key $SK_{IBE,S_{i,j}} = \big(K_0 = (Z)^{L_j} g^{\beta_{ID|_{k-1}}} (vw^T)^{r_2} Y'_0,\ K_1 = g^{-r_2} Y'_1\big)$.

If $Z = Z_0 = X_2 Y_2$, then the simulation is the same as $\mathbf{H}'_{(1,0),2}$ since all HIBE private keys and IBE private keys with the group index h implicitly uses a random polynomial $f_{GL}(x) \equiv \log_g(X_2) \cdot x + \beta_{ID|_{k-1}} \mod p_1$ and it implicitly sets $\delta_{i,j} \equiv \log_{g_{p_2}}(R_2)\delta'_{i,j} \mod p_2$ for each member index $i_m \neq h^*_m$. If $Z = Z_1 = X_2 R_3 Y_2$, then the simulation is the same as $\mathbf{H}''$ since it implicitly sets $\delta_{i,j} = \log_{g_{p_2}}(R_3) L_j \mod p_2$ for the member index h^*_m. As mentioned before, the HIBE private key query for $ID^*|_k$ and the IBE private key query for T^* should belong to the same member index h^*_m by the restriction $ID^*|_k \in R^*$ of the security model.

We now show that the paradox of dual system encryption can be solved. To check whether an HIBE private key for h^*_m and an IBE private key for h^*_m are normal or semi-functional, the simulator may try to decrypt a semi-functional ciphertext by deriving a decryption key from these keys for h^*_m. However, the simulator cannot derive a decryption key from those keys since the Lagrange interpolation method does not work for the same h^*_m since only one point of $f_{GL}(x)$ is revealed. Recall that the Lagrange interpolation method requires two points of $f_{GL}(x)$ to derive $f_{GL}(0)$. Thus, the simulator cannot check whether these two keys for the same h^*_m are normal or semi-functional. This completes our proof. □

5.3.2 Type-II Adversary

Lemma 5.15. *If the GSD assumption holds, then no polynomial-time Type-II adversary can distinguish* $\boldsymbol{I}_{(h_m,h_c-1),2}$ *from* $\boldsymbol{I}_{(h_m,h_c),1}$ *with a non-negligible advantage.*

Lemma 5.16. *If the GSD assumption holds, then no polynomial-time Type-II adversary can distinguish* $\boldsymbol{I}_{(h_m,h_c),1}$ *from* $\boldsymbol{I}_{(h_m,h_c),2}$ *with a non-negligible advantage.*

Lemma 5.17. *If the GSD assumption holds, then no polynomial-time Type-II adversary can distinguish* $\boldsymbol{I}'_{(h_m,h_c-1),2}$ *from* $\boldsymbol{I}'_{(h_m,h_c),1}$ *with a non-negligible advantage.*

Lemma 5.18. *If the GSD assumption holds, then no polynomial-time Type-II adversary can distinguish* $\boldsymbol{I}'_{(h_m,h_c),1}$ *from* $\boldsymbol{I}'_{(h_m,h_c),2}$ *with a non-negligible advantage.*

The proofs of Lemmas 5.15, 5.16, 5.17, and 5.18 are almost the same as those of Lemmas 5.10, 5.11, 5.12, and 5.13 respectively except that there is no case $i_m = h^*_m$ since the Type-II adversary does not request an HIBE private key for $ID^*|_k \in \mathbf{Prefix}(ID^*|_\ell)$ and an IBE private key for T^*. We omit the detailed proofs of these lemmas.

6 Conclusion

In this work, we proposed two RHIBE schemes by combining LW-HIBE and LW-IBE schemes in composite-order bilinear groups, and the CS (or SD) scheme in a modular way, and then we proved the adaptive security of our RHIBE schemes by using the dual system encryption technique. As mentioned before, we carefully re-designed hybrid games to use the dual system encryption technique since a naive approach of dual system encryption does not work. Our RHIBE schemes are the first RHIBE schemes that achieve the adaptive security.

References

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2005.

[2] Josh Benaloh, Melissa Chase, Eric Horvitz, and Kristin E. Lauter. Patient controlled encryption: ensuring privacy of electronic medical records. In Radu Sion and Dawn Song, editors, *ACM Cloud Computing Security Workshop - CCSW 2009*, pages 103–114. ACM, 2009.

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

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

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

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

[7] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271. Springer, 2003.

[8] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2004.

[9] Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In Joan Feigenbaum, editor, *DRM 2002*, volume 2696 of *Lecture Notes in Computer Science*, pages 61–80. Springer, 2002.

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

[11] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

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

[13] Kwangsu Lee. Self-updatable encryption with short public parameters and its extensions. *Designs Codes Cryptogr.*, 79(1):121–161, 2016.

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

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

[16] Kwangsu Lee, Dong Hoon Lee, Jong Hwan Park, and Moti Yung. CCA security for self-updatable encryption: Protecting cloud data when clients read/write ciphertexts. Cryptology ePrint Archive, Report 2015/1202, 2015. `http://eprint.iacr.org/2015/1202`.

[17] Kwangsu Lee and Seunghwan Park. Revocable hierarchical identity-based encryption with shorter private keys and update keys. Cryptology ePrint Archive, Report 2016/460, 2016. `http://eprint.iacr.org/2016/460`.

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

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

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

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

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

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

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

[25] Jae Hong Seo and Keita Emura. Adaptive-id secure revocable hierarchical identity-based encryption. In Keisuke Tanaka and Yuji Suga, editors, *Advances in Information and Computer Security - IWSEC 2015*, volume 9241 of *Lecture Notes in Computer Science*, pages 21–38. Springer, 2015.

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

[27] Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *IEEE Symposium on Security and Privacy - S&P 2007*, pages 350–364. IEEE Computer Society, 2007.

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