Adaptive-ID Secure Revocable Hierarchical Identity-Based Encryption

Jae Hong Seo¹ and Keita Emura²(✉)

¹ Department of Mathematics, Myongji University, Seoul, Korea
jaehongseo@mju.ac.kr

² Security Fundamentals Lab, National Institute of Information and Communications Technology (NICT), Tokyo, Japan
k-emura@nict.go.jp

Abstract. Revocable Hierarchical Identity-Based Encryption (RHIBE) is a variant of Identity-Based Encryption (IBE), which enables the dynamic user management; a Key Generation Center (KGC) of a usual IBE has a key issuing ability. In contrast, in a RHIBE, a KGC can revoke compromised secret keys and even delegate both key issuing ability and revocation ability.

Recently, Seo and Emura proposed the first construction for RHIBE (CT-RSA 2013) and then refined the security model and the construction for RHIBE (CT-RSA 2015). Nevertheless, their constructions achieve only a slightly weaker security notion, called selective-ID security, in the sense that the adversary has to choose and declare the target identity before she receives the system parameter of target RHIBE scheme.

In this paper, we propose the first RHIBE construction that achieves a right security notion, called adaptive-ID security. In particular, our construction still has the advantages of the Seo-Emura RHIBE schemes; that is, it is scalable and achieves history-free update, security against insiders, and short ciphertexts. We employ the dual system encryption methodology.

1 Introduction

Revocable Identity-Based Encryption: Revocation is one of the important issues in the context of Identity-Based Encryption (IBE). Boneh and Franklin [3,4] considered the first (non-scalable) revocation method in the IBE context, where for all non-revoked identities ID at a time period T, secret keys are computed for ID|T. The first scalable IBE with revocation, which we call RIBE, is proposed by Boldyreva, Goyal, and Kumar (BGK) [1] where a Key Generation Center (KGC) broadcasts key update information kuT at time T, and only non-revoked users at T can compute their decryption keys dkID,T by using their (long-term) secret key skID and kuT. It is worth noting that the size of kuT is O(Rlog(N/R)) by using the Naor-Naor-Lotspiech (NNL) framework [8] called Complete Subtree (CS) method, where R is the number of revoked users and N is the total number of users, whereas that of the Boneh-Franklin
scheme is $O(N - R)$. Though the BGK scheme is selective-ID secure, Libert and Vergnaud (LV) [7] proposed the first adaptive-ID secure RIBE scheme. Next, Seo and Emura [10,12] considered a new security threat called decryption key exposure resistance. They show that the Boneh-Franklin RIBE scheme is secure against decryption key exposure attack but the BGK and LV schemes are vulnerable against this attack, and proposed the first scalable RIBE scheme with decryption key exposure resistance.

**RIBE with Hierarchical Structures:** Seo and Emura [9,11] proposed the first Revocable hierarchical IBE (RHIBE) scheme with $O(\ell^2 \log N)$-size secret key, where $\ell$ is the maximum hierarchical level. However, in their construction a low-level user must know the history of key updates performed by ancestors in the current time period. This history-preserving update methodology makes the scheme very complex. Moreover, the ciphertext size depends on the depth of the hierarchy (i.e., $O(\ell)$). In addition to this, the security model only considered outsiders, where an adversary is not allowed to access to state information used by internal users of the RHIBE system.

Recently, Seo and Emura [13] proposed a RHIBE scheme which implements (1) history-free updates where a low-level user does not need to know the history of the key updates of ancestors, (2) security against insiders where an adversary is allowed to obtain internal state information, and (3) short ciphertexts where the size of ciphertext is constant in terms of the hierarchical level. Moreover, they considered decryption key exposure attack [10,12]. Their scheme is based on the Boneh-Boyen-Goh (BBG) HIBE scheme [2], i.e., they proved that the Seo-Emura RHIBE scheme is secure if the BBG HIBE is secure.

**Motivation:** There is a room for improvement of the Seo-Emura RHIBE scheme [13] because it achieves selective-ID security, where an adversary is required to declare the challenge identity before the system setup phase. We remark that the RHIBE scheme in [9,11] also selective-ID secure. Though Tsai et al. [14] proposed an adaptive-ID secure RHIBE scheme, this scheme is not scalable, i.e., the size of key update linearly depends on $N$.

The Seo-Emura construction [13] is a kind of conversion from HIBE to RHIBE since their construction and security proof are based on the (non-revocable) BBG HIBE scheme. Basically, the selective-ID security of the Seo-Emura construction comes from the underlying BBG HIBE scheme, which is also proven in the selective-ID security model only. Therefore, one may think that if one changes the underlying HIBE with the adaptive-ID secure HIBE such as [6], one can directly obtain the first adaptively secure RHIBE construction. Unfortunately, In the security proof of Seo and Emura [13], the reduction algorithm itself requires the challenge identity before the setup phase so that we can easily see this trivial approach ends in failure. Therefore, we need a different technique for the adaptive-ID secure RHIBE scheme.

**Contribution and Methodology:** In this paper, we propose the first scalable and adaptive-ID secure RHIBE scheme which has the advantages of the Seo-Emura RHIBE scheme [13]. In the contrast to the previous approaches [9–13],
we propose a scalable scheme and then directly prove its security. To this end, we modify the Lewko-Waters HIBE scheme [6] appropriately and then directly prove its adaptive security by employing dual system encryption methodology [15].

In the (usual) dual system encryption, a normal ciphertext can be decrypted by using either a normal decryption key or a semi-functional decryption key, but a semi-functional ciphertext cannot be decrypted by using a semi-functional decryption key. Since a decryption key is computed by key update information and a normal/semi-functional secret key, we need to guarantee not only a decryption simulator

\begin{equation}
\Pr[D,T_1=1]-\Pr[D,T_2=1]
\end{equation}

is negligible, where $T_1 \overset{\$}{\leftarrow} G_{p_1p_2}$ and $T_2 \overset{\$}{\leftarrow} G_{p_1p_3}$.

Definition 3 (Assumption 3 [6]). Let $(n, G, G_T, e) \overset{\$}{\leftarrow} G(\lambda), n = p_1p_2p_3$, $\alpha, s \overset{\$}{\leftarrow} Z_n$, $g \overset{\$}{\leftarrow} G_{p_1}$, $X_2, Y_2 \overset{\$}{\leftarrow} G_{p_2}$, and $X_3 \overset{\$}{\leftarrow} G_{p_3}$. Set $D := ((n, G, G_T, e), g, g^\alpha X_2, X_3, g^s Y_2, Z_2)$. We say that Assumption 3 holds if for all PPT adversaries $A$, $|Pr[A(D, T_1) = 1] - Pr[A(D, T_2) = 1]|$ is negligible, where $T_1 \overset{\$}{\leftarrow} G$ and $T_2 \overset{\$}{\leftarrow} G_{p_1p_3}$.
PPT adversaries $\mathcal{A}$, $|\Pr[\mathcal{A}(D,T_1) = 1] - \Pr[\mathcal{A}(D,T_2) = 1]|$ is negligible, where $T_1 = e(g,g)^{\alpha s}$ and $T_2 \leftarrow \mathbb{G}_T$.

Next, we introduce the KUNode algorithm [1] which implements the CS method. If $v$ is a non-leaf node, then $v_l$ denotes the left child of $v$. Similarly, $v_r$ is the right child of $v$. We assume that each user is assigned to a unique leaf node. If a user, which is assigned to $v$, is revoked on time $T$, then $(v,T)$ is added into the revocation list $RL$.

\[
\text{KUNode}(\mathcal{B}T, RL, T) : \\
X, Y \leftarrow \emptyset; \\
\text{For } \forall (v_i, T_i) \in RL, \text{ if } T_i \leq T, \text{ then add Path}(v_i) \text{ to } X; \\
\text{For } \forall v \in X, \text{ if } v_l \not\in X, \text{ then add } v_l \text{ to } Y; \\
\text{if } v_r \not\in X, \text{ then add } v_r \text{ to } Y; \\
\text{If } |RL| = 0, \text{ then add root to } Y; \\
\text{Return } Y;
\]

3 Revocable Hierarchical Identity-Based Encryption

In this section, we introduce the definition of RHIBE given by Seo and Emura [13] as follows. In order to achieve history-free updates, Seo and Emura pointed out that the following two situations are equivalent: (1) a user $ID$ is not revoked at time $T$, and (2) the user can generate the decryption key $dk_{ID,T}$, and re-defined the syntax of RHIBE from that of [9,11].

Definition 1 [13]. RHIBE consists of seven algorithms Setup, SKGen, KeyUp, DKGen, Enc, Dec, and Revoke defined as follows.

\[
\text{Setup}(1^\lambda, N, \ell): \text{ This algorithm takes as input the security parameter } 1^\lambda, \text{ maximum number of users in each level } N, \text{ and maximum hierarchical length } \ell, \text{ and outputs the public system parameter } mpk, \text{ the master secret key } msk, \text{ initial state information } st_0, \text{ and empty revocation list } RL. \text{ We assume that } mpk \text{ contains description of message space } M, \text{ identity space } I, \text{ and time space } T. \text{ For simplicity, we often omit } mpk \text{ in the input of other algorithms.}
\]

\[
\text{SKGen}(st_{ID|k-1}, ID|k): \text{ This algorithm takes as input } st_{ID|k-1} \text{ and an identity } ID|k, \text{ and outputs the secret key } sk_{ID|k}, \text{ and updates } st_{ID|k-1}.
\]

\[
\text{KeyUp}(dk_{ID|k-1,T}, st_{ID|k-1}, RL_{ID|k-1}, T): \text{ This algorithm takes as input the revocation list } RL_{ID|k-1}, \text{ state information } st_{ID|k-1}, \text{ the decryption key } dk_{ID|k-1,T}, \text{ and a time period } T. \text{ For } k = 1, \text{ we set } dk_{ID|k-1,T} \text{ to be msk disregarding } T. \text{ Then, it outputs the key update } ku_{ID|k-1,T}.
\]

\[
\text{DKGen}(sk_{ID|k}, ku_{ID|k-1,T}): \text{ This algorithm takes as input } sk_{ID|k} \text{ of } ID|k \text{ and } ku_{ID|k-1,T}, \text{ and outputs the decryption key } dk_{ID|k,T} \text{ of } ID|k \text{ at time } T \text{ if } ID|k \text{ is not revoked at } T \text{ by the parent.}
\]
Revoke\( (\mathcal{I}_k, T, RL_{\mathcal{I}_k}) \): This algorithm takes as input \( \mathcal{I}_k \) and \( T \), updates \( RL_{\mathcal{I}_k} \) managed by \( \mathcal{I}_{k-1} \), who is the parent user of \( \mathcal{I}_k \), by adding \((\mathcal{I}_k, T)\).

For any output \( \text{Setup}(1^\lambda, N, \ell) \rightarrow (\text{mpk}, \text{msk}) \), any message \( M \in \mathcal{M} \), any identity \( \mathcal{I}_k \in \mathcal{I} \) where \( k \in [1, \ell] \), any time \( T \in T \), all possible states \( \{\text{st}_{\mathcal{I}_i}\}_{i \in [1,k-1]} \), and all possible revocation lists \( \{RL_{\mathcal{I}_i}\}_{i \in [1,k-1]} \) if \( \mathcal{I}_k \) is not revoked on time \( T \), the following probability should be 1; \( i \) is initialized by 1. While \( i \in [1,k] \), repeatedly run

\[
\begin{pmatrix}
\text{SKGen}(\text{st}_{\mathcal{I}_{i-1}}, \mathcal{I}_{i}) \rightarrow \text{sk}_{\mathcal{I}_i}; \\
\text{KeyUp}(\text{dk}_{\mathcal{I}_{i-1}}, T, \text{st}_{\mathcal{I}_{i-1}}, RL_{\mathcal{I}_{i-1}}) \rightarrow \text{ku}_{\mathcal{I}_{i-1}, T}; \\
\text{DKGen}(\text{sk}_{\mathcal{I}_i}, \text{ku}_{\mathcal{I}_{i-1}, T}) \rightarrow \text{dk}_{\mathcal{I}_i, T}; \\
\hat{i} \leftarrow i + 1;
\end{pmatrix}
\]

Finally, compute \( \text{Enc}(M, \mathcal{I}_k, T) \rightarrow \text{CT} \) and \( \text{Dec}(\text{CT}, \text{dk}_{\mathcal{I}_k, T}) \rightarrow M' \). The perfect correctness requires that the probability that \( M = M' \) to be 1, where the probability is taken over the randomness used in all algorithms.

Next, we give the security model for RHIBE defined by Seo and Emura [13]. It is worth noting that the SKGen oracle returns not only \( \text{sk}_{\mathcal{I}} \) but also \( \text{st}_{\mathcal{I}} \).

So, security against insider is considered. They also considered decryption key exposure resistance [10, 12]. In this paper, we additionally consider adaptive-ID security.

SKGen\( (\cdot) \): The oracle takes as input \( \mathcal{I} \), and outputs \( \text{sk}_{\mathcal{I}} \) and \( \text{st}_{\mathcal{I}} \).

DKGen\( (\cdot, \cdot) \): The oracle takes as input \( \mathcal{I} \) and \( T \), and outputs \( \text{dk}_{\mathcal{I}, T} \).

KeyUp\( (\cdot, \cdot) \): The oracle takes as input \( \mathcal{I}_{k-1} \) and \( T \), and outputs \( \text{ku}_{\mathcal{I}_{k-1}, T} \). If \( k = 1 \), it means that \( \mathcal{A} \) asks the key updates for the first level users generated by the KGC.

Revoke\( (\cdot, \cdot) \): The oracle takes as input \( \mathcal{I}_k \) and \( T \), and adds a pair \((\mathcal{I}_k, T)\) into \( RL_{\mathcal{I}_{k-1}} \), where \( \mathcal{I}_{k-1} \) is the parent of \( \mathcal{I}_k \).

Given a RHIBE scheme \( RHIBE = (\text{Setup}, \text{SKGen}, \text{DKGen}, \text{KeyUp}, \text{Enc}, \text{Dec}, \text{Revoke}) \) and an adversary \( \mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\} \), we define an experiment \( \text{Exp}^{\text{IND-RID-CPA}}(1^\lambda, N, \ell; \rho) \), where \( \rho \) is a random tape for all randomness used in the experiment. Let \( \mathcal{O} \) be a set of oracles \( (\text{SKGen}(\cdot), \text{DKGen}(\cdot, \cdot), \text{KeyUp}(\cdot, \cdot), \text{Revoke}(\cdot, \cdot)) \).

\[
\text{Exp}^{\text{IND-RID-CPA}}(1^\lambda, N, \ell; \rho) :
\begin{align*}
(\text{mpk}, \text{msk}) & \leftarrow \text{Setup}(1^\lambda, N, \ell); (\mathcal{I}_k^*, M^*_0, M^*_1, \text{state}) \leftarrow \mathcal{A}_0(\text{mpk}); \\
b & \leftarrow \{0, 1\}; \text{CT}^* & \leftarrow \text{Enc}(M^*_0, \mathcal{I}_k^*, T^*); b' & \leftarrow \mathcal{A}_2(\text{mpk}, \text{CT}^*, \text{state}); \\
\text{Return} & \begin{cases} 
1 & \text{if } b = b' \text{ and the Conditions are satisfied} \\
0 & \text{otherwise}
\end{cases}
\end{align*}
\]
1. $M_0^*$ and $M_1^*$ have the same length.
2. $A$ has to query to $\text{KeyUp}(\cdot, \cdot)$ and $\text{Revoke}(\cdot, \cdot)$ in increasing order of time.
3. $A$ cannot query to $\text{Revoke}(\cdot, \cdot)$ on time $T$ if it already queried to $\text{KeyUp}(\cdot, \cdot)$ on time $T$.
4. If $|\text{ID}|_{k^*}$ is queried to $\text{SKGen}(\cdot)$ on time $T'$, where $k' \leq k^*$ and $T' \leq T$, then $A$ must query to revoke the challenge identity $|\text{ID}|_{k^*}$ or one of its ancestors on time $T'' \leq T' \leq T$.
5. $A$ cannot query decryption keys $d_{|\text{ID}|_{k^*},T^*}$ of the challenge identity or its ancestors on the challenge time $T^*$, where $k' \leq k^*$.

The advantage is defined as

$$\text{Adv}^{\text{IND-RID-CPA}}_{\text{RHIBE}, A}(\lambda, N, \ell) := \left| \Pr_{\rho}[\text{Exp}^{\text{IND-RID-CPA}}_{\text{RHIBE}, A}(\lambda, N, \ell; \rho) \rightarrow 1] - \frac{1}{2} \right|$$

**Definition 2 (IND-RID-CPA).** We say that RHIBE is IND-RID-CPA secure if for any polynomials $N$ and $\ell$ and probabilistic polynomial time algorithm $A$, the function $\text{Adv}^{\text{IND-RID-CPA}}_{\text{RHIBE}, A}(\lambda, N, \ell)$ is a negligible function in the security parameter $\lambda$.

4 Proposed Adaptive-ID Secure RHIBE

In this section, we give the proposed RHIBE scheme based on the Lewko-Waters HIBE scheme. A decryption key $d_{|\text{ID}|_{k^*}}$ can be computed from a (long-term) secret key $sk_{|\text{ID}|_k}$ and $ku_{|\text{ID}|_{k-1},T}$ if $|\text{ID}|_k$ is not revoked at $T$. $sk_{|\text{ID}|_k}$ is the same as that of the Lewko-Waters HIBE scheme, except that $g^\alpha$ is replaced to $P_\theta$ for a history-free construction as in the Seo-Emura RHIBE scheme [13]. That is, the secret key generation algorithm does not require any secret information from the ancestors but $sk_{|\text{ID}|_k}$ is necessary to compute a decryption key, namely $P_\theta$, called $\text{msk}$-shade, plays the role of a delegation key.

In the description, if $k = 1$, then $sk_{|\text{ID}|_{k-1}}$ means $\text{msk}$, $BT_{|\text{ID}|_{k-1}}$ means $BT_0$, and $ku_{|\text{ID}|_{k-1},T}$ means $ku_{0,T}$. We remark that $ku_{0,T}$ in [13] has the special form such that $ku_{0,T} = \{(P_\theta^{-1}g_2^\alpha(u^T h')^{t_\theta},g^{t_\theta})\}_{\theta}$, i.e., no $g^{t_\theta}$ part is contained. We explicitly define $ID_0 = 0$ as the identity of KGC and add the $g^{t_\theta}$ part to $ku_{0,T}$ in order to fix the form of key update information. This is necessary for employing dual system encryption since it is not guaranteed that a decryption key generated by semi-functional $ku_{0,T}$ is also semi-functional if $ku_{0,T}$ has a special form.

**Setup**($1^\lambda, N, \ell$): Run $G(1^\lambda) \rightarrow (n = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e)$. Choose random elements $g, h, v, u_0, u_1, \ldots, u_\ell, u', h' \overset{\$}{\leftarrow} \mathbb{G}_{p_1}, X_3 \overset{\$}{\leftarrow} \mathbb{G}_{p_3}$, and a random integer $\alpha \overset{\$}{\leftarrow} \mathbb{Z}_n$. Publish

$$\text{mpk} = \{g, h, u_0, u_1, \ldots, u_\ell, e(g, g)^\alpha, u', h', X_3\}$$

and keep $\text{msk} = \alpha$ in a secure storage.
SKGen(st_{ID[k−1]}, ID_k): The state information st_{ID[k−1]}, which is kept by ID_k, contains the binary tree BT_{ID[k−1]}. For ID_k = (l_0, l_1, ..., l_k), assign a random leaf node of BT_{ID[k−1]} to ID_k. For each θ in path(ID_k) ⊂ BT_{ID[k−1]}, recall $P_θ$ if it is stored. Otherwise, choose $P_θ \leftarrow \mathbb{G}_{p_1}$ using g, assign and store it in the corresponding node in the tree. For each $θ \in \text{Path}(ID_k)$ choose $R_{3,θ}, R_{k+1,θ}, ..., R_{ℓ,θ} \leftarrow \mathbb{G}_{p_2}$ using $X_3$ and $r_θ \leftarrow \mathbb{Z}_n$. Compute and output $sk_{ID_k}$ defined as

$$\left\{(P_θ(a_0^0 \cdots a_k^k h)^{r_θ} R_{3,θ}^g R_{3,θ}^{g^r} u_{k+1}^{r_θ} R_{k+1,θ}^{r_θ}, \ldots, u_ℓ^{r_θ} R_{ℓ,θ}) \right\}_{θ \in \text{Path}(ID_k)}.$$  

KeyUp(msk, st_0, RL_0, T): The state information st_0 contains the binary tree BT_0. Compute a set KUNode(BT_0, RL_0, T). For each $θ \in \text{KUNode}(BT_0, RL_0, T)$ recall msk-shade $P_θ$ if it is defined. Otherwise, a new msk-shade at $θ$ is chosen such that $P_θ \leftarrow \mathbb{G}_{p_1}$ using g, and store it in the corresponding node in the tree. Finally, the key update $ku_{0,T}$ is generated as follows: Choose $r_θ, t_θ \leftarrow \mathbb{Z}_n$ and $R_{3,θ}, R_{3,θ}', R_{3,θ}'', R_{1,θ}, ..., R_{ℓ,θ}' \leftarrow \mathbb{G}_{p_2}$ using $X_3$ for each $θ \in \text{KUNode}(BT_0, RL_0, T)$ and compute $ku_{0,T}$ as follows.

$$\left\{(P_θ^{-1} g^α(u_0^0 h)^{r_θ} (u^T h')^t R_{3,θ}^g R_{3,θ}'^{g^r} R_{3,θ}''^{g^t} u_1^{r_θ} R_{1,θ}, \ldots, u_ℓ^{r_θ} R_{ℓ,θ}) \right\}_{θ \in \text{KUNode}(BT_0, RL_0, T)},$$

where the set is for every $θ \in \text{KUNode}(BT_0, RL_0, T)$.

DKGen(sk_{ID_k}, ku_{ID[k−1], T}): Let ID_k = (l_0, ..., l_k). Parse ku_{ID[k−1], T} = {a_0, θ, a_1, θ, ..., a_ℓ, θ} and $sk_{ID_k} = \{(a_0, θ, a_1, θ, b_{k+1}, θ, ..., b_ℓ, θ)\}_{θ \in S}$, where $S$ and $S'$ are sets of nodes. If (ID_k, .) $∉ RL_{ID[k−1]}$, then there should be at least one node $θ$ in Path(ID_k) $∩ S$. For such $θ$, compute

$$(A_0, A_1, A_2, B_{k+1}, ..., B_ℓ) = (a_0, θ, a_1, θ, b_{k+1}, θ, ..., b_ℓ, θ).$$

Finally, re-randomize the result and output it as $dk_{ID_k,T}$. We explain how to re-randomize it later.

KeyUp(dk_{ID[k−1], T}, st_{ID[k−1]}, RL_{ID[k−1], T}): For each $θ \in \text{KUNode}(BT_{ID[k−1], T}, RL_{ID[k−1], T})$, recall $P_θ$ if it is stored. Otherwise, choose $P_θ \leftarrow \mathbb{G}_{p_1}$ using g and store it in the corresponding node in the tree. Let $dk_{ID[k−1], T}$ be (A_0, A_1, A_2, B_k, ..., B_ℓ). For each $θ \in \text{KUNode}(BT_{ID[k−1], T}, RL_{ID[k−1], T})$, re-randomize the decryption key with fresh randomness $r_θ, t_θ, s_0, θ, s_1, θ, s_2, θ, s_3, θ, ..., s_ℓ, θ \leftarrow \mathbb{Z}_n$ so that obtain $(a_0, θ, a_1, θ, a_2, θ, b_{k+1}, θ, ..., b_ℓ, θ)$. Finally, $ku_{ID[k−1], T}$ is generated as

$$\left\{(P_θ^{-1} a_0, θ, a_1, θ, a_2, θ, b_{k+1}, θ, ..., b_ℓ, θ) \right\}_{θ \in \text{KUNode}(BT_{ID[k−1], T}, RL_{ID[k−1], T})}.$$  

Enc(M, ID_k, T): Let ID_k = (l_0, ..., l_k). Choose an integer $s \leftarrow \mathbb{Z}_n$ at random and compute

$$CT = (M \cdot e(g, g)^{αs}, g^s, (u_0^0 \cdots u_k^k h)^s, (u^T h')^s).$$
Decryption Key Re-randomization: The decryption key for $ID_k$ is of the above form. decryption key for $ID_k$ is of the above form. The proposed RHIBE scheme is IND-RID-CPA secure under Assumptions 1, 2, and 3.
Before giving the proof, we explain our strategy. In particular, how to apply dual system encryption methodology to R(H)IBE. In dual system encryption, normal ciphertext can be decrypted by using either normal decryption key or semi-functional key, but semi-functional ciphertext cannot be decrypted by using semi-functional decryption key. In R(H)IBE, a decryption key \( \sk_{\text{ID}} \) and key update information \( \sk_{\text{KeyUp}} \) can only create a nominally semi-functional key regarding to the functionality of RHIBE, we further need to consider the cases that key update information is generated by decryption keys.

- From a normal secret key and normal key update information, a (non-revoked) user can compute a normal decryption key.
- From a semi-functional secret key and normal key update information, a (non-revoked) user can compute a semi-functional decryption key.
- From a normal secret key and semi-functional key update information, a (non-revoked) user can compute a semi-functional decryption key.
- From a semi-functional secret key and semi-functional key update information, a (non-revoked) user can compute a semi-functional decryption key.

Due to the functionality of RHIBE, we further need to consider the cases that key update information is generated by decryption keys.

- From a semi-functional decryption key, the \( \text{KeyUp} \) algorithm outputs semi-functional key update information.

**Remark:** It is particularly worth noting that, in the security proof of the Lewko-Waters HIBE scheme, the simulator \( B \) is compelled to be able to only create a “nominally” semi-functional key, which still works for a semi-functional ciphertext, in order to prevent that \( B \) attempts to test itself whether key is semi-functional by creating a semi-functional ciphertext and trying to decrypt. Since there are many routes to create decryption keys in the RHIBE context, we need to guarantee that still \( B \) can only create a nominally semi-functional key regardless of the decryption key generation route.

**Semi-functional Ciphertext:** Let \( g_2 \) be a generator of \( \mathbb{G}_{p_2} \), and let \((C', C'_0, C'_1, C'_2)\) be a ciphertext generated by the encryption algorithm.
Choose \( x, z_c \in \mathbb{Z}_n \), and set \((C'', C_0, C'_1, C'_2) = (C', C'_0g_2^x, C'_1g_2^{z_c}, C'_2)\).

**Semi-functional Secret Key:** Let \( \{a_0', a_1', b'_1, b'_2, \ldots, b'_\ell, \theta\} \) be a secret key \( \sk_{\text{ID}|_{i-1}} \) by the \( \text{SKGen} \) algorithm. Choose \( z_k \in \mathbb{Z}_n \), and for each \( \theta \)
choose \( \gamma_\theta, z_i, z_{i+1}, \ldots, z_\ell, \gamma_\ell \in \mathbb{Z}_n \), and set \( \{(a_0', a_1', k_{i+1}, \ldots, k_{\ell}, \theta) = \{(a_0', a_1, g_2^{\gamma}, k_{i+1}, \ldots, k_{\ell}, \theta) = \theta \} \).

**Semi-functional Key Update Information:** Let \( \{a_0'_1, a_1'_1, a_2, k_{i+1}, \ldots, k_{\ell, \theta}\} \) be key update information \( \sk_{\text{KeyUp}} \) by the \( \text{KeyUp} \) algorithm. Choose \( z_k \in \mathbb{Z}_n \), and for each \( \theta \)
choose \( \gamma_\theta, z_i, z_{i+1}, \ldots, z_\ell, \gamma_\ell \in \mathbb{Z}_n \), and set \( \{(a_0'_1, a_1', a_2, k_{i+1}, \ldots, k_{\ell, \theta}) = \{(a_0', a_1', g_2^{\gamma}, a_2, k_{i+1}, \ldots, k_{\ell, \theta}) = \theta \} \).

Adaptive-ID Secure Revocable Hierarchical Identity-Based Encryption 29
Semi-functional Decryption Key: Let \((A'_0, A'_1, A'_2, B'_{i+1}, \ldots, B'_t)\) be a decryption key generated by the \(\text{DKGen}\) algorithm. Choose 
\[
\gamma, z_k, z_{i+1}, \ldots, z_t \overset{\$}{\leftarrow} \mathbb{Z}_m,
\]
and set \((A_0, A_1, A_2, B_{i+1}, \ldots, B_t) = (A'_0 g_2^\gamma, A'_1 g_2^\gamma, A'_2, B'_{i+1} g_2^{z_{i+1}}, \ldots, B'_t g_2^{z_t})\).

Semi-functional Ciphertext\times Normal Decryption Key: Let a semi-functional ciphertext be \(\text{CT} = (M \cdot e(g, g)^{\alpha s}, g^s g_2^z, (u_0^b \cdots u_i^h)^s g_2^{xz_c}, (u^T h')^s)\) and a normal decryption key be \((g^\alpha (u_0^b \cdots u_i^h)^r (u^T h')^t R_3^c \cdot g^r R_3, g^t R''_3, b_{i+1} R_{i+1}, \ldots, b_t R_t)\). Then, the Dec algorithm computes 
\[
e(g^r R_3, (u_0^b \cdots u_i^h)^s g_2^{xz_c}) e(g^t R''_3, (u^T h')^s) e(g^\alpha (u_0^b \cdots u_i^h)^r (u^T h')^t R_3^c, g^s g_2^z) = \frac{1}{e(g, g)^{\alpha s}}
\]
Therefore, the Dec algorithm normally works.

Semi-functional Ciphertext\times Semi-functional Decryption Key: Let a semi-functional ciphertext be \(\text{CT} = (M \cdot e(g, g)^{\alpha s}, g^s g_2^z, (u_0^b \cdots u_i^h)^s g_2^{xz_c}, (u^T h')^s)\) and a semi-functional decryption key be \((g^\alpha (u_0^b \cdots u_i^h)^r (u^T h')^t R_3^c g_2^{z_c}, g^r R_3 g_2^{\gamma z_{i+1}}, g^t R''_3, b_{i+1} R_{i+1} g_2^{\gamma z_{i+1}}, \ldots, b_t R_t g_2^{z_c})\). Then, the Dec algorithm computes 
\[
e(g^r R_3 g_2^{\gamma}, (u_0^b \cdots u_i^h)^s g_2^{xz_c}) e(g^t R''_3, (u^T h')^s) e(g^\alpha (u_0^b \cdots u_i^h)^r (u^T h')^t R_3^c g_2^{z_c}, g^s g_2^z) = \frac{e(g_2, g_2)^{x\gamma(z_c-z_k)}}{e(g, g)^{\alpha s}}
\]
That is, as in the Lewko–Waters HIBE scheme, when a semi-functional key is used to decrypt a semi-functional ciphertext, the Dec algorithm will compute the blinding factor multiplied by the additional term \(e(g_2, g_2)^{x\gamma(z_c-z_k)}\). If \(z_c = z_k\), then the decryption still works and we say that the decryption key is nominally semi-functional.

Organization of the Proof: In the following security proof, first the challenge ciphertext is changed to be semi-functional. Next, a decryption key is changed to be semi-functional one by one. We remark that key update information generated by a semi-functional decryption key is also semi-functional. That is, we need to consider the following case: \(A\) issues a decryption key query for \(\text{ID}_i\) and the simulator \(B\) sends a normal decryption key. Later, \(A\) issues a decryption key query of its ancestor’s identity, say \(\text{ID}_{i-1}\), and \(B\) sends a semi-functional decryption key. Moreover, \(A\) issues a secret key query of this identity \(\text{ID}_{i-1}\) and \(B\) sends a normal secret key. Then, \(A\) can compute key update information from the semi-functional decryption key, and therefore can compute a decryption key of \(\text{ID}_i\) by myself. Then, a decryption key is semi-functional. So, we need to guarantee that all decryption keys appeared in the games are normal or nominally semi-functional regardless of its generation routes.

Next, key update information is changed to be semi-functional one by one. Though each \(k_{\text{ID}, T}\) contains \(O(R \log (N/R))\)-size subkeys, we replace all subkeys.
simultaneously since the differences of subkeys are randomness part only. As in
the previous games, we need to guarantee that all decryption keys appeared in
the games are normal or nominally semi-functional regardless of its generation
routes.

Next, a secret key is changed to be semi-functional one by one. Though each
sk_{ID} contains O(log N)-size subkeys, we replace all subkeys simultaneously since
the differences of subkeys are randomness part only. Finally, the plaintext of the
challenge ciphertext is replaced as a random value.

As the most different point between HIBE and RHIBE, an adversary is
allowed to obtain the secret key of the challenge identity (or its ancestors).
Moreover, the simulator can choose state information P_{\theta} directly
(which helps us to prove insider security). Therefore the simulator can gener-
ate the secret key of the challenge identity (or its ancestors) directly. So, in the
following all games, B chooses P_{\theta} and stores it as state information.

Let Game_{real} be the real security game. The next game, Game_{restricted}
is the same as Game_{real} except that the adversary is not allowed to issue key generation
queries for identities which are prefixes of the challenge identity modulo
p_{2}.

Lemma 1 ([6]). Suppose there exists an adversary A such that
\text{Game}_{real}^{Adv}A - \text{Game}_{restricted}^{Adv}A = \epsilon Then, we can build an algorithm
B with advantage \geq \epsilon/2 in breaking Assumption 2.

The next game, Game_{0} is the same as Game_{restricted} except that the challenge
ciphertext is semi-functional.

Lemma 2. Suppose there exists an adversary A such that
\text{Game}_{restricted}^{Adv}A - \text{Game}_{0}^{Adv}A = \epsilon Then, we can build an algorithm B
with advantage \epsilon in breaking Assumption 1.

Proof. First, B is given (g, X_{3}, \bar{T}). B chooses \alpha, a_{0}, \ldots, a_{\ell}, b \$ \mathbb{Z}_{n} and sets g = g,
u_{i} = g^{a_{i}} (i \in [0, \ell]), and h = g^{b}. Moreover, B guesses T^{*} (with success probability
at least 1/|T|), and chooses c, d \$ \mathbb{Z}_{n}, and sets u' = g^{c} and h' = u'^{-T^{*}} g^{d}. B
sends public parameters \text{mpk}_{mLW} = \{u, g, h, u_{0}, \ldots, u_{\ell}, e(g, g)^{\alpha}, u', h', X_{3}\} to A.

For a key query \text{ID}|_{j} = (l_{0}, \ldots, l_{j}), B generates a normal secret key as
follows. B chooses r_{\theta}, t_{\theta}, w_{\theta}, v_{j_{1}, \theta}, \ldots, v_{\ell} \$ \mathbb{Z}_{n} for each \theta, and computes
\[\left\{ (P_{\theta}(u_{l_{0}}^{l_{j}} u_{j_{1}, \theta}^{l_{j}} h)^{v_{j_{1}}} X_{3}^{w_{\theta}} g^{r_{\theta}} X_{3}^{t_{\theta}} u_{j_{1}+1}^{r_{\theta}} X_{3}^{v_{j_{1}+1, \theta}} \ldots, u_{l_{\ell}}^{r_{\theta}} X_{3}^{v_{\ell}, \theta}) \right\}_{\theta}.\]
Since B knows \alpha, B can answer all key update information, secret key, and
decryption key queries. B generates normal values for these queries.

A sends two messages M_{0} and M_{1} and the challenge identity \text{ID}|_{k^{*}} = (l_{0}^{*}, \ldots, l_{\ell}^{*}) to B. B chooses b \$ \{0, 1\}, and computes C' = M_{b} \cdot e(T, g)^{\alpha},
C_{0} = \bar{T} a_{0} l_{0}^{*} + \cdots + a_{k} l_{k}^{*} + b, C_{1} = \bar{T}, and C_{2} = \bar{T} d. Here, the G_{p_{1}} part of \bar{T}
is implicitly set g^{s} and the G_{p_{2}} part of \bar{T} d is implicitly set (u'^{-T^{*}} h')^{s} if B's
guessing is correct. If T \in G_{p_{1}p_{2}}, then this ciphertext is semi-functional with
z_c = a_1l_1^* + \cdots + a_kl_k^* + b. If T \in \mathbb{G}_{p_1}, then this ciphertext is normal. So, \mathcal{B}
uses the output of \mathcal{A}
for breaking Assumption 1.

For the number of decryption key queries q_{dk}, Game_{k}^{dk} (k \in [1, q_{dk}]) is the same as that of Game_{k-1}^{dk} except that the first k decryption keys are semi-functional and the rest of the keys are normal. We remark that Game_0^{dk} = Game_0.

**Lemma 3.** Suppose there exists an adversary \mathcal{A} such that Game_{k-1}^{dk} \text{Adv}_{\mathcal{A}} - Game_k^{dk} \text{Adv}_{\mathcal{A}} = \epsilon
Then, we can build an algorithm \mathcal{B}
with advantage \epsilon
in breaking Assumption 2.

**Proof.** First, \mathcal{B}
is given \((g, X_1X_2, X_3, Y_2Y_3, T)
. \mathcal{B}
chooses \(a, a_0, \ldots, a_{\ell}, b \overset{\$}{\leftarrow} \mathbb{Z}_n\)
and sets \(g = g, u_i = g^{a_i} (i \in [0, \ell])\), and \(h = g^b\). Moreover, \mathcal{B}
guesses \(T^*\) (with success probability at least \(1/|T|\)), chooses \(c, d \overset{\$}{\leftarrow} \mathbb{Z}_n\), and sets \(u' = g^c\) and \(h' = u'^{-T^*} g^d\). \mathcal{B}
sends public parameters \(\text{mpk}_n^{\text{W}} = \{n, g, h, u_0, \ldots, u_{\ell}, e(g, g)^a, u', h', X_3\}\) to \mathcal{A}.
Since \mathcal{B}
knows \(a, \mathcal{B}
\) can answer all key update information and secret key queries. \mathcal{B}
generates normal values for these queries.

For the \(i\)-th decryption key query \((\text{ID}|_j, T)\) where \(\text{ID}|_j = (l_0, \ldots, l_j)\) and \(i < k\), \mathcal{B}
generates a semi-functional decryption key. \mathcal{B}
chooses \(r, z, t, w, z_{j+1}, \ldots, z_{\ell} \overset{\$}{\leftarrow} \mathbb{Z}_n\), and computes
\[
\begin{align*}
(g^a (u_0^{l_0} \cdots u_j^{l_j} h)^{r} (u'^T h')^t (Y_2Y_3)^z, g^r (Y_2Y_3)^w, g^t X_3^w, \\
u_{j+1} (Y_2Y_3)^{z_{j+1}}, \ldots, u_{\ell} (Y_2Y_3)^{z_{\ell}}).
\end{align*}
\]
For \(i > k\), \mathcal{B}
generates a normal decryption key by using \(a\). For \(i = k\), \mathcal{B}
sets \(z_k = a_0l_0 + \cdots + a_jl_j + b\), chooses \(t, w, w', w_{j+1}, \ldots, w_{\ell} \overset{\$}{\leftarrow} \mathbb{Z}_n\), and computes \(\text{dk}_{\text{ID}|_j}, T = (g^a \bar{T}^k (u'^T h')^t X_3^w, \bar{T}, g^r X_3^w, \bar{T}^{a_{j+1}} X_3^{w_{j+1}}, \ldots, \bar{T}^{a_{\ell}} X_3^{w_{\ell}}).\) If \(\bar{T} \in \mathbb{G}_{p_1p_3}\), then this key is normal where the \(\mathbb{G}_{p_1}\) part of \(\bar{T}\) is set as \(g^r\). If \(\bar{T} \in \mathbb{G}\), this key is semi-functional.

\(\mathcal{A}\)
sends two messages \(M_0\) and \(M_1\) and the challenge identity \(\text{ID}|_k^* = (l_0^*, \ldots, l_k^*)\) to \(\mathcal{B}\). \mathcal{B}
chooses \(b \overset{\$}{\leftarrow} \{0, 1\}\), and computes \(c' = M_0 \cdot e(X_1X_2, g)^a, C_0 = (X_1X_2)^a l_0^* + \cdots + a_k l_k^* + b, C_1 = X_1X_2,\) and \(C_2 = X_1^d\). Here, \(X_1\) is implicitly set \(g^a\) and \(z_c = a_0l_0^* + \cdots + a_k l_k^* + b\). If \(\bar{T} \in \mathbb{G}_{p_1p_3}\), then \(\mathcal{B}\) has properly simulated Game_{k-1}. If \(\bar{T} \in \mathbb{G}\), then \(\mathcal{B}\) has properly simulated Game_{k}^{dk}.

If \(\mathcal{B}\) tries to test whether the answer of \(k\)-th decryption key query \(\text{dk}_{\text{ID}|_j}, T\) is semi-functional or not by computing a semi-functional ciphertext, \(\mathcal{B}\) can only create a nominally semi-functional decryption key with \(z_k = z_c\). The answer of \(k\)-th decryption key query \(\text{dk}_{\text{ID}|_j}, T\) is semi-functionally since \(z_k = z_c\). So, we have to consider the case that key update information \(\text{ku}_{\text{ID}|_j}, T\) is generated by this \(\text{dk}_{\text{ID}|_j}, T\) (then \(\text{ku}_{\text{ID}|_j}, T\) is semi-functional), and \(\text{dk}_{\text{ID}|_{j+1}}, T\) is computed by \(\text{ku}_{\text{ID}|_j}, T\) and a normal secret key \(\text{sk}_{\text{ID}|_{j+1}}, \text{ku}_{\text{ID}|_j}, T\) is represented as
\[
\{ (P_1^{-1} g^a \bar{T}^{z_c} (u_0^{l_0} \cdots u_j^{l_j} h)^{r_{0,\theta}} (u'^T h')^t_\theta X_3^{w_{0,\theta}}, \bar{T} g^r X_3^{s_{1,\theta}}, g^{t_\theta} X_3^{w_{1,\theta}},
\]
where $r, t, s, \theta, s', s'_{j+1, \theta}, \ldots, s'_{\ell, \theta} \in \mathbb{Z}_n$ are chosen for each $\theta$. Then since a normal secret key $sk_{ID|_{j+1}}$ is represented as

$$
\left\{ (P_0(u_0^0 \ldots u_{j+1}^0 h)^{r_0} R_{3, \theta}, g^{r_0} R_{3, \theta}, u_j^{r_0} R_{j+1, \theta}, \ldots, u_{t_0}^{r_0} R_{t, \theta}) \right\}_\theta,
$$

the first component of $dk_{ID|_{j+1}, T}$ is represented as

$$
g^a T^{z_k + a_j + 1} (u_0^0 \ldots u_{j+1}^0 h)^{r_0 + r'} (u'^T h')^{t + t_0 + t'} (X_3^{\omega_0, \theta + (w_j + s'_{j+1, \theta} + s'_{\ell, \theta})} R_{3, \theta})
$$

where $r', t', s_0 \in \mathbb{Z}_n$ are for re-randomization. As we can see, $dk_{ID|_{j+1}, T}$ preserves the form with $z_k' := z_k + a_j + 1$ and therefore, $dk_{ID|_{j+1}, T}$ is nominally semi-functional.

From the above estimations, we confirmed that even if $B$ tries to test, $B$ can only create a nominally semi-functional decryption key. So, $B$ uses the output of $A$ for breaking Assumption 2. □

For the number of key update information queries $q_{ku}$, $Game^ku_{k}$ ($k \in [1, q_{ku}]$) is the same as that of $Game^ku_{k-1}$ except that the first $k$ key update information are semi-functional and the rest of these are normal. We remark that $Game^ku_0 = Game^{dk}_0$.

**Lemma 4.** Suppose there exists an adversary $A$ such that $Game^ku_{k-1}Adv_A - Game^ku_{k}Adv_A = \epsilon$ Then, we can build an algorithm $B$ with advantage $\epsilon$ in breaking Assumption 2.

**Proof.** First, $B$ is given $(g, X_1, X_2, X_3, Y_2, Y_3, T)$. $B$ chooses $\alpha, a_0, \ldots, a_\ell, b \in \mathbb{Z}_n$ and sets $g = g^a, u_i = g^{a_i}$ ($i \in [0, \ell]$), and $h = g^b$. Moreover, $B$ guesses $T^*$ (with success probability at least $1/|T|$), chooses $c, d \in \mathbb{Z}_n$, and sets $u' = g^c$ and $h' = u'^{-T} g^d$. $B$ sends public parameters $mpk_{ml\mathbb{W}} = \{ n, g, h, u_0, \ldots, u_{\ell}, c(g, g)^a, u', h', X_3 \}$ to $A$. Since $B$ knows $\alpha$, $B$ can answer all secret key and decryption key queries. $B$ generates normal secret keys. Moreover, $B$ computes semi-functional decryption keys of $(ID|_{j}, T)$ where $ID|_{j} = (l_0, \ldots, l_j)$ as follows. $B$ chooses $r, z, t, w, w', z_{j+1, \theta}, \ldots, z_{\ell, \theta} \in \mathbb{Z}_n$, and computes

$$(g^{\alpha} (u_0^0 \ldots u_{j+1}^0 h)^{r_0} (u'^T h')^{t} (Y_2 Y_3)^z, g^{r} (Y_2 Y_3)^w, g^{t} X_3^{w'}, u_{j+1}^{r'} (Y_2 Y_3)^{z_{j+1, \theta}}, \ldots, u_{\ell}^{r'} (Y_2 Y_3)^{z_{\ell, \theta}}).$$

For the $i$-th key update information query $(ID|_{j}, T)$ where $ID|_{j} = (l_0, \ldots, l_j)$ and $i < k$, $B$ generates semi-functional key update information $ku_{ID|_{j}, T}$ as follows. $B$ chooses $r_0, z_0, t_0, w_0, w'_0, z_{j+1, \theta}, \ldots, z_{\ell, \theta} \in \mathbb{Z}_n$ for each $\theta$, and computes

$$
\left\{ (P_0^{-1} g^{\alpha} (u_0^0 \ldots u_{j}^0 h)^{r_0} (u'^T h')^{t_0} (Y_2 Y_3)^{z_0}, g^{r_0} (Y_2 Y_3)^{w_0}, g^{t_0} X_3^{w'_0},
$$

where
For the number of secret key queries $z > 34$. J.H. Seo and K. Emura

decryption key in the case that $\mathcal{G}$ computes Lemma 5.

Suppose there exists an adversary $A$ such that $\mathcal{G} \left( \mathcal{P}_1, \mathcal{P}_2 \right)$ is nominally semi-functional.

As in the case of $\mathcal{G}$, we can easily check $\mathcal{D}_{\mathcal{G}}$ is a nominally semi-functional decryption key in the case that $\mathcal{D}_{\mathcal{G}}$ is computed by $\mathcal{G}$ and a normal secret key $\mathcal{G}_{\mathcal{G}}$. So, $\mathcal{G}$ uses the output of $\mathcal{A}$ for breaking Assumption 2.

For the number of secret key queries $q_{sk}$, $\mathcal{G}_{sk}$ is the same as that of $\mathcal{G}_{sk}$ except that the first $k$ secret keys are semi-functional and the rest of keys are normal. We remark that $\mathcal{G}_{sk} = \mathcal{G}_{sk}$.

Lemma 5. Suppose there exists an adversary $A$ such that $\mathcal{G}_{k}^{sk} A - \mathcal{G}_{k}^{sk} A = \epsilon$. Then, we can build an algorithm $\mathcal{B}$ with advantage $\epsilon$ in breaking Assumption 2.

Proof. First, $\mathcal{B}$ is given $(g, X_1 X_2, X_3, Y_2 Y_3, \hat{T})$. $\mathcal{B}$ chooses $\alpha, a_0, \ldots, a_{\ell}, b \leftarrow \mathbb{Z}_n$ and $g = g, u_i = g^{a_i}$ (i.e., $\hat{T}(\mathcal{P}_1, \mathcal{P}_2)$, and $h = g^b$. Moreover, $\mathcal{B}$ guesses $T^*$ (with success probability at least $1 | [T]$), chooses $c, d \leftarrow \mathbb{Z}_n$, and $u' = g^c$ and $h' = u'^{-T^*} g^d$. $\mathcal{B}$ sends public parameters $\mathcal{B}_{\mathcal{G}}(\mathcal{P}_1, \mathcal{P}_2)$ to $\mathcal{A}$.

For a decryption key query $\mathcal{D}_{\mathcal{G}}(\mathcal{P}_1, \mathcal{P}_2)$, $\mathcal{B}$ computes a semi-functional decryption key $\mathcal{D}_{\mathcal{G}}$ as follows. $\mathcal{B}$ sets $z_k = a_0 l_0 + \cdots + a_j l_j + b$, and chooses $r, z, t, w, w', z+1, \ldots, z_{\ell}, z \leftarrow \mathbb{Z}_n$, and computes

$$g^{\alpha} (u_0^{l_0} \cdots u_j^l) R (u'^{T} h) (Y_2 Y_3)^{z z_k}, g^r (Y_2 Y_3)^{w z_k}, g^{t} X_3^{w'}$$

$$u_{j+1}^{T} (Y_2 Y_3)^{z_{j+1} z_k}, \ldots, u_{\ell}^{T} (Y_2 Y_3)^{z_{\ell} z_k}.$$
That follows. \( B \) sets \( z_k = a_0l_0 + \cdots + a_jl_j + b \), and chooses \( r_\theta, z_\theta, t_\theta, w_\theta, w'_\theta, z_{j+1,\theta}, \ldots, z_{\ell,\theta} \in \mathbb{Z}_n \) for each \( \theta \), and computes
\[
\left\{ (P_\theta^{-1} g^\alpha(u_0^0 \cdots u_j^j h))^{r_\theta}(u^T h')^{t_\theta}(Y_2 Y_3)^{z_\theta z_k}, g^{r_\theta} (Y_2 Y_3)^{w_\theta z_k}, g^{t_\theta} X^w_3, \\
u_j^{r_\theta} (Y_2 Y_3)^{z_j,\theta z_k}, \ldots, u_{\ell}^{r_\theta} (Y_2 Y_3)^{z_\ell,\theta z_k}) \right\}_\theta.
\]
For the \( i \)-th secret key query \( ID|_j \) where \( ID|_j = (l_0, \ldots, l_j) \) and \( i < k \), \( B \) generates a semi-functional secret key \( sk_{ID|_j} \) as follows. \( B \) sets \( z_k = a_0l_0 + \cdots + a_jl_j + b \), chooses \( r_\theta, z_\theta, t_\theta, w_\theta, w'_\theta, z_{j+1,\theta}, \ldots, z_{\ell,\theta} \in \mathbb{Z}_n \) for each \( \theta \), and computes
\[
\left\{ (P_\theta(u_0^0 \cdots u_j^j h))^{r_\theta}(u^T h')^{t_\theta}(Y_2 Y_3)^{z_\theta z_k}, g^{r_\theta} (Y_2 Y_3)^{w_\theta z_k}, g^{t_\theta} X^w_3, \\
u_j^{r_\theta} (Y_2 Y_3)^{z_j,\theta z_k}, \ldots, u_{\ell}^{r_\theta} (Y_2 Y_3)^{z_\ell,\theta z_k}) \right\}_\theta.
\]
For \( i > k \), \( B \) generates normal secret keys. For \( i = k \), \( B \) sets \( z_k = a_0l_0 + \cdots + a_jl_j + b \), chooses \( \gamma_\theta, t_\theta, w_\theta, w'_\theta, z_{j+1,\theta}, \ldots, w_{\ell,\theta} \in \mathbb{Z}_n \), and computes \( sk_{ID|_j} \) as
\[
\left\{ (P_\theta^r T_\theta^{\gamma_\theta z_k}(u^T h')^{t_\theta} X^w_3, T_\theta^{\gamma_\theta t_\theta} X^w_3, g^{t_\theta} X^w_3, T_\theta^{\gamma_\theta a_j+1} X^w_3, \ldots, T_\theta^{\gamma_\theta a_\ell} X^w_3) \right\}_\theta.
\]
If \( T \in \mathbb{G}_{p_1 p_3} \), then this key is normal where the \( \mathbb{G}_{p_1} \) part of \( T^{\gamma_\theta} \) is set as \( g^{r_\theta} \). If \( T \in \mathbb{G} \), this key is semi-functional.

\( A \) sends two messages \( M_0 \) and \( M_1 \) and the challenge identity \( ID|_{k^*} = (l_0^*, \ldots, l_{k^*}) \) to \( B \). \( B \) chooses \( b \in \mathbb{Z}_2 \) and computes \( C' = M_b \cdot e(X_1 X_2, g)^{a} \), \( C_0 = (X_1 X_2)^{a l_0^* + \cdots + a_{k^*} l_{k^*} + b} \), \( C_1 = X_1 X_2 \), and \( C_2 = X_1^d \). Here, \( X_1 \) is implicitly set \( g^s \) and \( z_c = a_0l_0^* + \cdots + a_{k^*} l_{k^*} + b \). If \( T \in \mathbb{G}_{p_1 p_3} \), then \( B \) has properly simulated \( \text{Game}^k_{k-1} \). If \( T \in \mathbb{G} \), then \( B \) has properly simulated \( \text{Game}^k \).

If \( B \) tries to test whether the answer of \( k \)-th secret key query \( sk_{ID|_j} \) is semi-functional or not by computing a semi-functional ciphertext, \( B \) can only create a nominally semi-functional decryption key with \( z_k = z_c \). Let us consider the case that a decryption key \( dk_{ID|_j, T} \) is computed by semi-functional key update information \( ku_{ID|_{j-1},T} \) and \( sk_{ID|_j} \), where \( ku_{ID|_{j-1},T} \) is defined as
\[
\left\{ (P_\theta^{-1} g^\alpha(u_0^0 \cdots u_{j-1}^j h))^{r_\theta}(u^T h')^{t_\theta}(Y_2 Y_3)^{z_\theta z_k}, g^{t_\theta} X^w_3, \\
u_j^{r_\theta} (Y_2 Y_3)^{z_j,\theta z_k}, \ldots, u_{\ell}^{r_\theta} (Y_2 Y_3)^{z_\ell,\theta z_k}) \right\}.
\]
For some \( \theta \), the first component of \( dk_{ID|_j, T} \) output by the \( \text{DKGen} \) algorithm is represented as
\[
g^\alpha T^{\gamma_\theta z_k} (u_0^0 \cdots u_j^j h)^{r_\theta} g^\gamma (u^T h')^{t_\theta} (Y_2 Y_3)^{z_k (z_\theta + z_j, \theta l_j)}
\]
That is, the \( \mathbb{G}_{p_2} \) part of this value can be represented as \( g_2^{\gamma z_k} \) for some \( \gamma \in \mathbb{Z}_n \). Therefore, \( dk_{ID|_j, T} \) is a nominally semi-functional decryption key.
Similarly, we have to care about the case that \( \text{ku}_{ID|j-1,T} \) is computed a semi-functional decryption key \( dk_{ID|j-1,T} \), and \( dk_{ID|j,T} \) is computed by \( \text{ku}_{ID|j-1,T} \) and \( sk_{ID|j} \). Since \( \text{ku}_{ID|j-1,T} \) is computed by multiplying \( P_\theta^{-1} \) to \( dk_{ID|j,T} \) and the re-randomization process is independent of the \( G_{p_2} \) part, \( dk_{ID|j,T} \) is a nominally semi-functional decryption key.

From the above estimations, we confirmed that even if \( B \) tries to test, \( B \) can only create a nominally semi-functional decryption key. So, \( B \) uses the output of \( A \) for breaking Assumption 2.

The next game, \( \text{Game}_{final} \) is the same as \( \text{Game}_{qs} \) except that the challenge ciphertext is semi-functional of a random message (not one of the challenge messages).

**Lemma 6.** Suppose there exists an adversary \( A \) such that \( \text{Game}_{qs}^{sk} \text{Adv}_A = \epsilon \) Then, we can build an algorithm \( B \) with advantage \( \epsilon \) in breaking Assumption 3.

**Proof.** First, \( B \) is given \( (g, g^\alpha X_2, X_3, g^\beta Y_2, Z_2, T) \). \( B \) chooses \( a_0, \ldots, a_\ell, b \leftarrow Z_n \) and sets \( g = g, u_i = g^{a_i} (i \in [0, \ell]), h = g^b, \) and \( e(g, g)\alpha = e(g^\alpha X_2, g) \).

Moreover, \( B \) guesses \( T^* \) (with success probability at least \( 1/|T| \)), chooses \( c, d \leftarrow Z_n \), and sets \( u' = g^c \) and \( h' = u'^{-T^*} g^d \). \( B \) sends public parameters \( \text{mpk}^{mlW} = \{n, g, h, u_0, \ldots, u_\ell, e(g, g)\alpha, w', h', X_3\} \) to \( A \).

For a secret key query \( ID|j = (I_0, \ldots, I_j) \), \( B \) generates a semi-functional secret key as follows. \( B \) chooses \( c_0, r_0, t_0, w_0, z_0, z_{j+1,0}, \ldots, z_{\ell,0}, w_{j+1,0}, \ldots, w_{\ell,0} \leftarrow Z_n \) for each \( \theta \), and computes \( sk_{ID|j} \) as

\[
\left\{ (P_\theta Z_2^{c_0} (u_0^{l_0} \ldots u_j^{l_j})^{r_0} X_3^{w_0}, g^{r_0} Z_2^{z_0} X_3^{t_0}, \\
u_{j+1}^{r_0} Z_2^{z_{j+1,0}} X_3^{w_{j+1,0}} \cdots, u_{\ell}^{r_0} Z_2^{z_{\ell,0}} X_3^{w_{\ell,0}} \right\}_\theta.
\]

For a key update information query \( (ID|j, T) \) where \( ID|j = (I_0, \ldots, I_j) \), \( B \) generates semi-functional key update information as follows. \( B \) chooses \( c_0, r_0, t_0, t_0', w_0, z_0, z_{j+1,0}, \ldots, z_{\ell,0}, w_{j+1,0}, \ldots, w_{\ell,0} \leftarrow Z_n \) for each \( \theta \), and computes \( \text{ku}_{ID|j,T} \) as

\[
\left\{ (P_\theta^{-1} g^\alpha X_2 Z_2^{c_0} (u_0^{l_0} \ldots u_j^{l_j})^{r_0} (u'^T h')^{t_0} X_3^{w_0}, g^{r_0} Z_2^{z_0} X_3^{t_0}, g^{t_0} X_3^{w_0}, \\
u_{j+1}^{r_0} Z_2^{z_{j+1,0}} X_3^{w_{j+1,0}} \cdots, u_{\ell}^{r_0} Z_2^{z_{\ell,0}} X_3^{w_{\ell,0}} \right\}_\theta.
\]

For a decryption key query \( (ID|j, T) \) where \( ID|j = (I_0, \ldots, I_j) \), \( B \) generates semi-functional decryption key as follows. \( B \) chooses \( c, r, t, t', w, w', z, z_{j+1}, \ldots, z_{\ell}, w_{j+1}, \ldots, w_{\ell} \leftarrow Z_n \), and computes \( \text{dk}_{ID|j,T} \) as

\[
(g^\alpha X_2 Z_2^{c}(u_0^{l_0} \ldots u_j^{l_j})^{r}(u'^T h')^t X_3^{w}, g^r Z_2^z X_3^{t'}, g^t X_3^{w}).
\]
\[u_{r+1}^j Z_{j+1}^{w_{j+1}} X_3^{w_{j+1}}, \ldots, u_{r}^* Z_{j}^{w_{j}} X_3^{w_{j}}\).

\(A\) sends two messages \(M_0\) and \(M_1\) and the challenge identity \(ID|_{s^*} = (I^*_0, \ldots, I^*_s)\) to \(B\). \(B\) chooses \(b \leftarrow \{0, 1\}\), and computes \(C' = M_b \cdot \bar{T}, C_0 = (g^s Y_2)^{a_0 I^*_0 + \cdots + a_k I^*_k + b}, C_1 = g^s Y_2,\) and \(C_2 = (g^s Y_2)^d\). If \(\bar{T} = e(g, g)^{\alpha s}\), then \(C' = M_b e(g, g)^{\alpha s}\). Therefore, this is a semi-functional ciphertext of \(M_b\). If \(\bar{T}\) is a random element of \(G_T\), then this is a semi-functional ciphertext of a random message. So, \(B\) uses the output of \(A\) for breaking Assumption 3.

\[\square\]

5 Conclusion

In this paper, we propose the first adaptive-ID secure and scalable RHIBE scheme. Moreover, our construction has the advantages of the Seo-Emura RHIBE scheme [13]; that is, it has the history-free update, security against insiders, short ciphertexts, and decryption key exposure resistance. Since our scheme is constructed over composite order bilinear groups, it is a natural open problem to construct adaptive-ID secure RHIBE scheme over prime order settings. Lewko’s technique for prime-order construction [5] might be useful for this problem.

References
