Adaptive-ID Secure Revocable Hierarchical Identity-Based Encryption

Jae Hong Seo[1] and Keita Emura[2](✉)

[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 $\mathsf{ID}||T$. 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 ku_T at time T, and only non-revoked users at T can compute their decryption keys $\mathsf{dk}_{\mathsf{ID},T}$ by using their (long-term) secret key $\mathsf{sk}_{\mathsf{ID}}$ and ku_T. It is worth noting that the size of ku_T is $O(R\log(N/R))$ 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

K. Tanaka and Y. Suga (Eds.): IWSEC 2015, LNCS 9241, pp. 21–38, 2015.
DOI: 10.1007/978-3-319-22425-1_2

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 ℓ 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 contrcution [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 secret key in the RHIBE context, we need to check whether a decryption key is semi-functional or not when it is generated by normal/semi-functional key update information and a normal/semi-functional secret key. Especially, in the security proof of the Lewko-Waters HIBE scheme, in order to prevent that the simulator $\mathcal{B}$ attempts to test itself whether key is semi-functional by creating a semi-functional ciphertext and trying to decrypt, $\mathcal{B}$ is required to compute a "nominally"semi-functional key only which still works for a semi-functional ciphertext. Since there are many routes for generating decryption keys in the RHIBE context, in the security proof we need to guarantee not only a decryption key is nominally semi-functional but also a decryption key generated by key update information and a secret key is nominally semi-functional.

2 Preliminaries

In this section, we give the definitions of complexity assumptions. Let $\mathcal{G}$ be a group generator which takes as input the security parameter λ, and outputs $(n = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ be cyclic groups with order n, p_1, p_2, and p_3 are distinct prime numbers, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is an efficient, non-degenerate bilinear map. We denote the subgroup of $\mathbb{G}$ with order m as $\mathbb{G}_m$. Here, for all $X_i \in \mathbb{G}_{p_i}$ and $X_j \in \mathbb{G}_{p_j}$ $(i, j \in \{1, 2, 3\})$, $e(X_i, X_j) = 1_T$ holds if $i \neq j$.

Definition 1 (Assumption 1 [6]). *Let* $(n, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(1^\lambda)$*,* $n = p_1p_2p_3$*,* $g \xleftarrow{\$} \mathbb{G}_{p_1}$*, and* $X_3 \xleftarrow{\$} \mathbb{G}_{p_3}$*. Set* $D := ((n, \mathbb{G}, \mathbb{G}_T, e), g, X_3)$*. We say that Assumption 1 holds if for all probabilistic polynomial-time (PPT) adversaries* $\mathcal{A}$*,* $|\Pr[\mathcal{A}(D, T_1) = 1] - \Pr[\mathcal{A}(D, T_2) = 1]|$ *is negligible, where* $T_1 \xleftarrow{\$} \mathbb{G}_{p_1p_2}$ *and* $T_2 \xleftarrow{\$} \mathbb{G}_{p_1}$*.*

Definition 2 (Assumption 2 [6]). *Let* $(n, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(1^\lambda)$*,* $n = p_1p_2p_3$*,* $g, X_1 \xleftarrow{\$} \mathbb{G}_{p_1}$*,* $X_2, Y_2 \xleftarrow{\$} \mathbb{G}_{p_2}$*, and* $X_3, Y_3 \xleftarrow{\$} \mathbb{G}_{p_3}$*. Set* $D := ((n, \mathbb{G}, \mathbb{G}_T, e), g, X_1X_2, X_3, Y_2Y_3)$*. We say that Assumption 2 holds if for all PPT adversaries* $\mathcal{A}$*,* $|\Pr[\mathcal{A}(D, T_1) = 1] - \Pr[\mathcal{A}(D, T_2) = 1]|$ *is negligible, where* $T_1 \xleftarrow{\$} \mathbb{G}$ *and* $T_2 \xleftarrow{\$} \mathbb{G}_{p_1p_3}$*.*

Definition 3 (Assumption 3 [6]). *Let* $(n, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(1^\lambda)$*,* $n = p_1p_2p_3$*,* $\alpha, s \xleftarrow{\$} \mathbb{Z}_n$*,* $g \xleftarrow{\$} \mathbb{G}_{p_1}$*,* $X_2, Y_2, Z_2 \xleftarrow{\$} \mathbb{G}_{p_2}$*, and* $X_3 \xleftarrow{\$} \mathbb{G}_{p_3}$*. Set* $D := ((n, \mathbb{G}, \mathbb{G}_T, e), g, g^\alpha X_2, X_3, g^s Y_2, Z_2)$*. We say that Assumption 3 holds if for all*

PPT adversaries $\mathcal{A}$, $|\Pr[\mathcal{A}(D,T_1)=1]-\Pr[\mathcal{A}(D,T_2)=1]|$ *is negligible, where* $T_1=e(g,g)^{\alpha s}$ *and* $T_2 \xleftarrow{\$} \mathbb{G}_T$.

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.

$\mathsf{KUNode}(\mathsf{BT},RL,T)$:

$\mathtt{X},\mathtt{Y}\leftarrow\emptyset$;

For $\forall(\mathsf{v}_i,T_i)\in RL$, if $T_i\le T$, then add $\mathsf{Path}(\mathsf{v}_i)$ to $\mathtt{X}$;

For $\forall\mathsf{v}\in\mathtt{X}$, if $\mathsf{v}_l\notin\mathtt{X}$, then add v_l to $\mathtt{Y}$;

if $\mathsf{v}_r\notin\mathtt{X}$, then add v_r to $\mathtt{Y}$;

If $|RL|=0$, then add $\mathtt{root}$ to $\mathtt{Y}$;

Return $\mathtt{Y}$;

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 $\mathsf{dk}_{\mathsf{ID},T}$, 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.*

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

$\mathsf{SKGen}(\mathsf{st}_{ID|_{k-1}},ID|_k)$: *This algorithm takes as input* $\mathsf{st}_{ID|_{k-1}}$ *and an identity* $ID|_k$*, outputs the secret key* $\mathsf{sk}_{ID|_k}$*, and updates* $\mathsf{st}_{ID|_{k-1}}$.

$\mathsf{KeyUp}(\mathsf{dk}_{ID|_{k-1},T},\mathsf{st}_{ID|_{k-1}},RL_{ID|_{k-1}},T)$: *This algorithm takes as input the revocation list* $RL_{ID|_{k-1}}$*, state information* $\mathsf{st}_{ID|_{k-1}}$*, the decryption key* $\mathsf{dk}_{ID|_{k-1},T}$*, and a time period* T*. For* $k=1$*, we set* $\mathsf{dk}_{ID|_{k-1},T}$ *to be* msk *disregarding* T*. Then, it outputs the key update* $\mathsf{ku}_{ID|_{k-1},T}$.

$\mathsf{DKGen}(\mathsf{sk}_{ID|_k},\mathsf{ku}_{ID|_{k-1},T})$: *This algorithm takes as input* $\mathsf{sk}_{ID|_k}$ *of* $ID|_k$ *and* $\mathsf{ku}_{ID|_{k-1},T}$*, and outputs the decryption key* $\mathsf{dk}_{ID|_k,T}$ *of* $ID|_k$ *at time* T *if* $ID|_k$ *is not revoked at* T *by the parent.*

$\mathsf{Enc}(M, \mathit{ID}, T)$: *This algorithm takes as input a message M, ID and the current time T and outputs the ciphertext* CT.

$\mathsf{Dec}(\mathsf{CT}, \mathsf{dk}_{\mathit{ID},T})$: *This algorithm takes as input* CT *and* $\mathsf{dk}_{\mathit{ID},T}$*, and outputs the message.*

$\mathsf{Revoke}(\mathit{ID}|_k, T, RL_{\mathit{ID}|_{k-1}})$: *This algorithm takes as input $\mathit{ID}|_k$ and T, updates $RL_{\mathit{ID}|_{k-1}}$ managed by $\mathit{ID}|_{k-1}$, who is the parent user of $\mathit{ID}|_k$, by adding $(\mathit{ID}|_k, T)$.*

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

$$\begin{pmatrix} \mathsf{SKGen}(\mathsf{st}_{\mathsf{ID}|_{i-1}}, \mathsf{ID}|_i) \to \mathsf{sk}_{\mathsf{ID}|_i}; \\ \mathsf{KeyUp}(\mathsf{dk}_{\mathsf{ID}|_{i-1},T}, \mathsf{st}_{\mathsf{ID}|_{i-1}}, RL_{\mathsf{ID}|_{i-1}}) \to \mathsf{ku}_{\mathsf{ID}|_{i-1},T}; \\ \mathsf{DKGen}(\mathsf{sk}_{\mathsf{ID}|_i}, \mathsf{ku}_{\mathsf{ID}|_{i-1},T}) \to \mathsf{dk}_{\mathsf{ID}|_i,T}; \\ i \leftarrow i+1; \end{pmatrix}$$

Finally, compute $\mathsf{Enc}(M, \mathsf{ID}|_k, T) \to \mathsf{CT}$ and $\mathsf{Dec}(\mathsf{CT}, \mathsf{dk}_{\mathsf{ID}|_k,T}) \to M'$. 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 $\mathsf{sk}_{\mathsf{ID}}$ but also $\mathsf{st}_{\mathsf{ID}}$. So, security against insider is considered. They also considered decryption key exposure resistance [10,12]. In this paper, we additionally consider adaptive-ID security.

$\mathsf{SKGen}(\cdot)$: The oracle takes as input ID, and outputs $\mathsf{sk}_{\mathsf{ID}}$ and $\mathsf{st}_{\mathsf{ID}}$.

$\mathsf{DKGen}(\cdot,\cdot)$: The oracle takes as input ID and T, and outputs $\mathsf{dk}_{\mathsf{ID},T}$.

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

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

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

$$\begin{aligned}
&\mathbf{Exp}^{\text{IND-RID-CPA}}_{\mathcal{RHIBE},\mathcal{A}}(\lambda, N, \ell; \rho): \\
&(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, N, \ell); (\mathsf{ID}|^*_{k^*}, M^*_0, M^*_1, \mathit{state}) \leftarrow \mathcal{A}^{\mathcal{O}}_1(\mathsf{mpk}); \\
&b \xleftarrow{\$} \{0,1\}; \mathsf{CT}^* \leftarrow \mathsf{Enc}(M^*_b, \mathsf{ID}|^*_{k^*}, T^*); b' \leftarrow \mathcal{A}^{\mathcal{O}}_2(\mathsf{mpk}, \mathsf{CT}^*, \mathit{state}); \\
&\text{Return} \begin{cases} 1 \text{ if } b = b' \text{ and the Conditions are satisfied} \\ 0 \text{ otherwise} \end{cases}
\end{aligned}$$

1. M_0^* and M_1^* have the same length.
2. $\mathcal{A}$ has to query to $\mathsf{KeyUp}(\cdot,\cdot)$ and $\mathsf{Revoke}(\cdot,\cdot)$ in increasing order of time.
3. $\mathcal{A}$ cannot query to $\mathsf{Revoke}(\cdot,\cdot)$ on time T if it already queried to $\mathsf{KeyUp}(\cdot,\cdot)$ on time T.
4. If $\mathsf{ID}|_{k'}^*$ is queried to $\mathsf{SKGen}(\cdot)$ on time T', where $k' \leq k^*$ and $T' \leq T$, then $\mathcal{A}$ must query to revoke the challenge identity $\mathsf{ID}|_{k^*}^*$ or one of its ancestors on time $T' \leq T'' \leq T$.
5. $\mathcal{A}$ cannot query decryption keys $\mathsf{dk}_{\mathsf{ID}|_{k'}^*,T^*}$ of the challenge identity or its ancestors on the challenge time T^*, where $k' \leq k^*$.

The advantage is defined as

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

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

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 $\mathsf{dk}_{\mathsf{ID}|_k,T}$ can be computed from a (long-term) secret key $\mathsf{sk}_{\mathsf{ID}|_k}$ and $\mathsf{ku}_{\mathsf{ID}|_{k-1},T}$ if $\mathsf{ID}|_k$ is not revoked at T. $\mathsf{sk}_{\mathsf{ID}|_k}$ is the same as that of the Lewko-Waters HIBE scheme, except that g^α is replaced to P_θ 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 $\mathsf{sk}_{\mathsf{ID}|_k}$ is necessary to compute a decryption key, namely P_θ, called msk-shade, plays the role of a delegation key.

In the description, if $k = 1$, then $\mathsf{sk}_{\mathsf{ID}|_{k-1}}$ means msk, $\mathsf{BT}_{\mathsf{ID}|_{k-1}}$ means BT_0, and $\mathsf{ku}_{\mathsf{ID}|_{k-1},T}$ means $\mathsf{ku}_{0,T}$. We remark that $\mathsf{ku}_{0,T}$ in [13] has the special form such that $\mathsf{ku}_{0,T} = \{(P_\theta^{-1} g_2^\alpha (u'^T h')^{t_\theta}, g^{t_\theta})\}_\theta$, i.e., no g^{r_θ} part is contained. We explicitly define $\mathsf{ID}_0 = \mathsf{I}_0$ as the identity of KGC and add the g^{r_θ} part to $\mathsf{ku}_{0,T}$ 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 $\mathsf{ku}_{0,T}$ is also semi-functional if $\mathsf{ku}_{0,T}$ has a special form.

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

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

and keep $\mathsf{msk} = \alpha$ in a secure storage.

$\mathsf{SKGen}(\mathsf{st}_{\mathsf{ID}|_{k-1}}, \mathsf{ID}|_k)$: The state information $\mathsf{st}_{\mathsf{ID}|_{k-1}}$, which is kept by $\mathsf{ID}|_{k-1}$, contains the binary tree $\mathsf{BT}_{\mathsf{ID}|_{k-1}}$. For $\mathsf{ID}|_k = (\mathsf{I}_0, \mathsf{I}_1, \ldots, \mathsf{I}_k)$, assign a random leaf node of $\mathsf{BT}_{\mathsf{ID}|_{k-1}}$ to $\mathsf{ID}|_k$. For each θ in $\mathsf{path}(\mathsf{ID}|_k) \subset \mathsf{BT}_{\mathsf{ID}|_{k-1}}$, recall P_θ if it is stored. Otherwise, choose $P_\theta \xleftarrow{\$} \mathbb{G}_{p_1}$ using g, assign and store it in the corresponding node in the tree. For each $\theta \in \mathsf{Path}(\mathsf{ID}|_k)$ choose $R'_{3,\theta}, R_{3,\theta}, R_{k+1,\theta}, \ldots, R_{\ell,\theta} \xleftarrow{\$} \mathbb{G}_{p_3}$ using X_3 and $r_\theta \xleftarrow{\$} \mathbb{Z}_n$. Compute and output $\mathsf{sk}_{\mathsf{ID}|_k}$ defined as

$$\left\{(P_\theta(u_0^{\mathsf{I}_0} \cdots u_k^{\mathsf{I}_k} h)^{r_\theta} R'_{3,\theta}, g^{r_\theta} R_{3,\theta}, u_{k+1}^{r_\theta} R_{k+1,\theta}, \ldots, u_\ell^{r_\theta} R_{\ell,\theta})\right\}_{\theta \in \mathsf{Path}(\mathsf{ID}|_k)}.$$

$\mathsf{KeyUp}(\mathsf{msk}, \mathsf{st}_0, RL_0, T)$: The state information st_0 contains the binary tree BT_0. Compute a set $\mathsf{KUNode}(\mathsf{BT}_0, RL_0, T)$. For each $\theta \in \mathsf{KUNode}(\mathsf{BT}_0, RL_0, T)$ recall msk-shade P_θ if it is defined. Otherwise, a new msk-shade at θ is chosen such that $P_\theta \xleftarrow{\$} \mathbb{G}_{p_1}$ using g, and store it in the corresponding node in the tree. Finally, the key update $\mathsf{ku}_{0,T}$ is generated as follows: Choose $r_\theta, t_\theta \xleftarrow{\$} \mathbb{Z}_n$ and $\bar{R}_{3,\theta}, \bar{R}'_{3,\theta}, \bar{R}''_{3,\theta}, \bar{R}_{1,\theta}, \ldots, \bar{R}_{\ell,\theta} \xleftarrow{\$} \mathbb{G}_{p_3}$ using X_3 for each $\theta \in \mathsf{KUNode}(\mathsf{BT}_0, RL_0, T)$ and compute $\mathsf{ku}_{0,T}$ as follows.

$$\left\{(P_\theta^{-1} g^\alpha (u_0^{\mathsf{I}_0} h)^{r_\theta} (u'^T h')^{t_\theta} \bar{R}_{3,\theta}, g^{r_\theta} \bar{R}'_{3,\theta}, g^{t_\theta} \bar{R}''_{3,\theta}, u_1^{r_\theta} \bar{R}_{1,\theta}, \ldots, u_\ell^{r_\theta} \bar{R}_{\ell,\theta})\right\}_\theta,$$

where the set is for every $\theta \in \mathsf{KUNode}(\mathsf{BT}_0, RL_0, T)$.

$\mathsf{DKGen}(\mathsf{sk}_{\mathsf{ID}|_k}, \mathsf{ku}_{\mathsf{ID}|_{k-1},T})$: Let $\mathsf{ID}|_k = (\mathsf{I}_0, \ldots, \mathsf{I}_k)$. Parse $\mathsf{ku}_{\mathsf{ID}|_{k-1},T} = \{\tilde{a}_{0,\theta}, \tilde{a}_{1,\theta}, \tilde{a}_{2,\theta}, \tilde{b}_{k,\theta}, \ldots, \tilde{b}_{\ell,\theta}\}_{\theta \in \mathsf{S}}$ and $\mathsf{sk}_{\mathsf{ID}|_k} = \{(a_{0,\theta}, a_{1,\theta}, b_{k+1,\theta}, \ldots, b_{\ell,\theta})\}_{\theta \in \mathsf{S}'}$, where S and S' are sets of nodes. If $(\mathsf{ID}|_k, \cdot, \cdot) \notin RL_{\mathsf{ID}|_{k-1}}$, then there should be at least one node θ in $\mathsf{Path}(\mathsf{ID}|_k) \cap \mathsf{S}$. For such θ, compute

$$\begin{aligned} &(A_0,\ A_1,\ A_2,\ B_{k+1}, \ldots, B_\ell) \\ =\ &(a_{0,\theta} \cdot \tilde{a}_{0,\theta} \cdot \tilde{b}_{k,\theta}^{\mathsf{I}_k},\ \ a_{1,\theta} \cdot \tilde{a}_{1,\theta},\ \tilde{a}_{2,\theta},\ b_{k+1,\theta} \cdot \tilde{b}_{k+1,\theta}, \ldots, b_{\ell,\theta} \cdot \tilde{b}_{\ell,\theta}). \end{aligned}$$

Finally, re-randomize the result and output it as $\mathsf{dk}_{\mathsf{ID}|_k,T}$. We explain how to re-randomize it later.

$\mathsf{KeyUp}(\mathsf{dk}_{\mathsf{ID}|_{k-1},T}, \mathsf{st}_{\mathsf{ID}|_{k-1}}, RL_{\mathsf{ID}|_{k-1}}, T)$:
For each $\theta \in \mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}|_{k-1}}, RL_{\mathsf{ID}|_{k-1}}, T)$, recall P_θ if it is stored. Otherwise, choose $P_\theta \xleftarrow{\$} \mathbb{G}_{p_1}$ using g and store it in the corresponding node in the tree. Let $\mathsf{dk}_{\mathsf{ID}|_{k-1},T}$ be $(A_0, A_1, A_2, B_k, \ldots, B_\ell)$. For each $\theta \in \mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}|_{k-1}}, RL_{\mathsf{ID}|_{k-1}}, T)$, re-randomize the decryption key with fresh randomness $r_\theta, t_\theta, s_{0,\theta}, s_{1,\theta}, s_{2,\theta}, s'_{k,\theta}, \ldots, s'_{\ell,\theta} \xleftarrow{\$} \mathbb{Z}_n$ so that obtain $(a_{0,\theta}, a_{1,\theta}, a_{2,\theta}, b_{k,\theta}, \ldots, b_{\ell,\theta})$. Finally, $\mathsf{ku}_{\mathsf{ID}|_{k-1},T}$ is generated as

$$\left\{(P_\theta^{-1} a_{0,\theta},\ a_{1,\theta},\ a_{2,\theta},\ b_{k,\theta}, \ldots, b_{\ell,\theta})\right\}_{\theta \in \mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}|_{k-1}}, RL_{\mathsf{ID}|_{k-1}}, T)}.$$

$\mathsf{Enc}(M, \mathsf{ID}|_k, T)$: Let $\mathsf{ID}|_k = (\mathsf{I}_0, \ldots, \mathsf{I}_k)$. Choose an integer $s \xleftarrow{\$} \mathbb{Z}_n$ at random and compute

$$\mathsf{CT} = (M \cdot e(g,g)^{\alpha s},\ g^s,\ (u_0^{\mathsf{I}_0} \cdots u_k^{\mathsf{I}_k} h)^s,\ (u'^T h')^s).$$

$\mathsf{Dec}(\mathsf{CT}, \mathsf{dk}_{\mathsf{ID}|_k,T})$: Let $\mathsf{ID}|_k = (\mathsf{I}_0, \ldots, \mathsf{I}_k)$. Parse $\mathsf{CT} = (C', C_0, C_1, C_2)$ and $\mathsf{dk}_{\mathsf{ID}|_k,T} = (A_0, A_1, A_2, B_{k+1}, \ldots, B_\ell)$. Output $C' \cdot \frac{e(A_1,C_1)\cdot e(A_2,C_2)}{e(A_0,C_0)}$

$\mathsf{Revoke}(\mathsf{ID}|_k, T, RL_{\mathsf{ID}|_{k-1}}, \mathsf{st}_{\mathsf{ID}|_{k-1}})$: Find a leaf node $\zeta_{\mathsf{ID}|_k}$, associating with $\mathsf{ID}|_k$, in $\mathsf{BT}_{\mathsf{ID}|_{k-1}}$. Update $RL_{\mathsf{ID}|_{k-1}}$ by adding a triple $(\mathsf{ID}|_k, T, \zeta_{\mathsf{ID}|_k})$.

Correctness: From the shape of the ciphertext for $\mathsf{ID}|_k$ and T, we can easily see that if $\mathsf{dk}_{\mathsf{ID}|_k,T}$ is of the form $(g^\alpha(u_0^{\mathsf{I}_0}\cdots u_k^{\mathsf{I}_k}h)^r(u'^T h')^t R'_3$, $g^r R_3, g^t R''_3, b_{k+1}R_{k+1}, \ldots, b_\ell R_\ell)$, then the DKGen algorithm correctly outputs the plaintext with probability 1. We show that $\mathsf{dk}_{\mathsf{ID}|_k,T}$, which is obtained by k times repeatedly performing SKGen, KeyUp and DKGen, is of the desired form $(g^\alpha(u_0^{\mathsf{I}_0}\cdots u_k^{\mathsf{I}_k}h)^r(u'^T h')^t R'_3,\ g^r R_3,\ g^t R''_3,\ u_{k+1}^r R_{k+1}, \ldots, u_\ell^r R_\ell)$ for some r and t. Then, it is sufficient for the perfect correctness. To this end, we will use the mathematical induction; for the first level ancestor, $\mathsf{sk}_{\mathsf{ID}|_1}$ and $\mathsf{ku}_{0,T}$ are respectively defined as

$$\Big\{(P_\theta(u_0^{\mathsf{I}_0}u_1^{\mathsf{I}_1}h)^{r_\theta}R'_{3,\theta}, g^{r_\theta}R_{3,\theta}, u_2^{r_\theta}R_{2,\theta}, \ldots, u_\ell^{r_\theta}R_{\ell,\theta})\Big\}_{\theta\in\mathsf{Path}(\mathsf{ID}|_1)} \text{ and}$$

$$\Big\{(P_\theta^{-1}g^\alpha(u_0^{\mathsf{I}_0}h)^{r'_\theta}(u'^T h')^{t_\theta}\bar{R}_{3,\theta}, g^{r'_\theta}\bar{R}'_{3,\theta}, g^{t_\theta}\bar{R}''_{3,\theta}, u_1^{r'_\theta}\bar{R}_{1,\theta}, \ldots, u_\ell^{r'_\theta}\bar{R}_{\ell,\theta})\Big\}_\theta,$$

where $\theta \in \mathsf{KUNode}(\mathsf{BT}_0, RL_0, T)$. For $\theta \in \mathsf{Path}(\mathsf{ID}|_1) \cap \mathsf{KUNode}(\mathsf{BT}_0, RL_0, T)$, $\mathsf{dk}_{\mathsf{ID}|_1}$ is equal to

$$\Big(g^\alpha(u_0^{\mathsf{I}_0}u_1^{\mathsf{I}_1}h)^{r_\theta+r'_\theta}(u'^T h')^{t_\theta}(R'_{3,\theta}\bar{R}'_{3,\theta}\bar{R}^{\mathsf{I}_1}_{1,\theta}),\ g^{r_\theta+r'_\theta}R_{3,\theta},\ g^{t_\theta}\bar{R}_{3,\theta},$$

$$u_2^{r_\theta+r'_\theta}(R_{2,\theta}\bar{R}_{2,\theta}), \ldots, u_\ell^{r_\theta+r'_\theta}(R_{\ell,\theta}\bar{R}_{\ell,\theta})\Big)$$

and later it is re-randomized. Thus, $\mathsf{dk}_{\mathsf{ID}|_1}$ is of the desired form. As in the similar confirmation, we can check the case of $k \geq 2$ under the assumption that $\mathsf{dk}_{\mathsf{ID}|_{k-1}}$ is of the above form.

Decryption Key Re-randomization: The decryption key for $\mathsf{ID}|_k = (\mathsf{I}_0, \ldots, \mathsf{I}_k)$ at T has the form $(g^\alpha(u_0^{\mathsf{I}_0}\cdots u_k^{\mathsf{I}_k}h)^r(u'^T h')^t R'_3$, $g^r R_3$, $g^t R''_3,\ u_{k+1}^r R_{k+1}, \ldots, u_\ell^r R_\ell)$. Since u_i's, h, g, u', h' and X_3 are publicly available, anyone can re-randomize $\mathsf{dk}_{\mathsf{ID}|_k,T} := (a_0, a_1, a_2, b_{k+1}, \ldots, b_\ell)$ by computing

$$\Big(a_0(u_0^{\mathsf{I}_0}\cdots u_k^{\mathsf{I}_k}h)^{r'}(u'^T h')^{t'}X_3^{s_0},\ a_1 g^{r'}X_3^{s_1},\ a_2 g^{t'}X_3^{s_2},$$

$$b_{k+1}u_{k+1}^{r'}X_3^{s'_{k+1}}, \ldots, b_\ell u_\ell^{r'}X_3^{s'_\ell}\Big)$$

with randomly chosen $r', t', s_0, s_1, s_2, s'_{k+1}, \ldots, s'_\ell \xleftarrow{\$} \mathbb{Z}_n$.

Theorem 1. *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 $\mathsf{dk}_{\mathsf{ID},T}$ is computed from a secret key $\mathsf{sk}_{\mathsf{ID}}$ and key update information ku_T. So, in our strategy, we newly define normal key update information/secret key and semi-functional key update information/secret key.

- 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.

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 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 $\mathcal{B}$ 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 $\mathcal{B}$ 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 $\mathcal{B}$ 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 $x, z_c \xleftarrow{\$} \mathbb{Z}_n$, and set $(C'', C'_0, C'_1, C'_2) = (C', C'_0 g_2^x, C'_1 g_2^{x z_c}, C'_2)$.

Semi-functional Secret Key: Let $\{(a'_{0,\theta}, a'_{1,\theta}, b'_{i+1,\theta}, \ldots, b'_{\ell,\theta})\}_\theta$ be a secret key $\mathsf{sk}_{\mathsf{ID}|_{i-1}}$ by the SKGen algorithm. Choose $z_k \xleftarrow{\$} \mathbb{Z}_n$ and for each θ choose $\gamma_\theta, z_{i+1,\theta}, \ldots, z_{\ell,\theta} \xleftarrow{\$} \mathbb{Z}_n$, and set $\{(a_{0,\theta}, a_{1,\theta}, k_{i+1,\theta}, \ldots, k_{\ell,\theta})\}_\theta = \{(a_{0,\theta} g_2^{\gamma_\theta z_k}, a_{1,\theta} g_2^{\gamma_\theta}, k_{i+1,\theta} g_2^{\gamma_\theta z_{i+1,\theta}}, \ldots, k_{\ell,\theta} g_2^{\gamma_\theta z_{\ell,\theta}})\}_\theta$.

Semi-functional Key Update Information: Let $\{(\tilde{a}'_{0,\theta}, \tilde{a}'_{1,\theta}, \tilde{a}'_{2,\theta}, \tilde{k}'_{i+1,\theta}, \ldots, \tilde{k}'_{\ell,\theta})\}_\theta$ be key update information $\mathsf{ku}_{\mathsf{ID}|_{i-1},T}$ by the KeyUp algorithm. Choose $z_k \xleftarrow{\$} \mathbb{Z}_n$ and for each θ choose $\gamma_\theta, z_{i+1,\theta}, \ldots, z_{\ell,\theta} \xleftarrow{\$} \mathbb{Z}_n$, and set $\{\tilde{a}_{0,\theta}, \tilde{a}_{1,\theta}, \tilde{a}_{2,\theta}, \tilde{k}_{i+1,\theta}, \ldots, \tilde{k}_{\ell,\theta}\}_\theta = \{(\tilde{a}'_{0,\theta} g_2^{\gamma_\theta z_k}, \tilde{a}'_{1,\theta} g_2^{\gamma_\theta}, \tilde{a}'_{2,\theta}, \tilde{k}'_{i+1,\theta} g_2^{\gamma_\theta z_{i+1,\theta}}, \ldots, \tilde{k}'_{\ell,\theta} g_2^{\gamma_\theta z_{\ell,\theta}})\}_\theta$.

Semi-functional Decryption Key: Let $(A_0', A_1', A_2', B_{i+1}', \ldots, B_\ell')$ be a decryption key generated by the DKGen algorithm. Choose $\gamma, z_k, z_{i+1}, \ldots, z_\ell \xleftarrow{\$} \mathbb{Z}_n$, and set $(A_0, A_1, A_2, B_{i+1}, \ldots, B_\ell) = (A_0' g_2^{\gamma z_k}, A_1' g_2^{\gamma}, A_2', B_{i+1}' g_2^{\gamma z_{i+1}}, \ldots, B_\ell' g_2^{\gamma z_\ell})$.

Semi-functional Ciphertext×Normal Decryption Key: Let a semi-functional ciphertext be $\mathsf{CT} = (M \cdot e(g,g)^{\alpha s},\ g^s g_2^x,\ (u_0^{\mathsf{I}_0} \cdots u_i^{\mathsf{I}_i} h)^s g_2^{x z_c},$ $(u'^T h')^s)$ and and a normal decryption key be $(g^\alpha (u_0^{\mathsf{I}_0} \cdots u_i^{\mathsf{I}_i} h)^r$ $(u'^T h')^t R_3', g^r R_3, g^t R_3'', b_{i+1} R_{i+1}, \ldots, b_\ell R_\ell)$. Then, the Dec algorithm computes

$$\frac{e(g^r R_3, (u_0^{\mathsf{I}_0} \cdots u_i^{\mathsf{I}_i} h)^s g_2^{x z_c}) e(g^t R_3'', (u'^T h')^s)}{e(g^\alpha (u_0^{\mathsf{I}_0} \cdots u_i^{\mathsf{I}_i} h)^r (u'^T h')^t R_3', g^s g_2^x)} = \frac{1}{e(g,g)^{\alpha s}}$$

Therefore, the Dec algorithm normally works.

Semi-functional Ciphertext×Semi-functional Decryption Key: Let a semi-functional ciphertext be $\mathsf{CT} = (M \cdot e(g,g)^{\alpha s},\ g^s g_2^x,\ (u_0^{\mathsf{I}_0} \cdots u_i^{\mathsf{I}_i} h)^s g_2^{x z_c},$ $(u'^T h')^s)$ and a semi-functional decryption key be $(g^\alpha (u_0^{\mathsf{I}_0} \cdots u_i^{\mathsf{I}_i} h)^r$ $(u'^T h')^t R_3' g_2^{\gamma z_k}, g^r R_3 g_2^{\gamma}, g^t R_3'', b_{i+1} R_{i+1} g_2^{\gamma z_{i+1}}, \ldots, b_\ell R_\ell g_2^{\gamma z_\ell})$. Then, the Dec algorithm computes

$$\frac{e(g^r R_3 g_2^{\gamma}, (u_0^{\mathsf{I}_0} \cdots u_i^{\mathsf{I}_i} h)^s g_2^{x z_c}) e(g^t R_3'', (u'^T h')^s)}{e(g^\alpha (u_0^{\mathsf{I}_0} \cdots u_i^{\mathsf{I}_i} h)^r (u'^T h')^t R_3' g_2^{\gamma z_k}, g^s g_2^x)} = \frac{e(g_2, g_2)^{x\gamma(z_c - z_k)}}{e(g,g)^{\alpha s}}$$

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: $\mathcal{A}$ issues a decryption key query for $\mathsf{ID}|_i$ and the simulator $\mathcal{B}$ sends a normal decryption key. Later, $\mathcal{A}$ issues a decryption key query of its ancestor's identity, say $\mathsf{ID}|_{i-1}$, and $\mathcal{B}$ sends a semi-functional decryption key. Moreover, $\mathcal{A}$ issues a secret key query of this identity $\mathsf{ID}|_{i-1}$ and $\mathcal{B}$ sends a normal secret key. Then, $\mathcal{A}$ can compute key update information from the semi-functional decryption key, and therefore can compute a decryption key of $\mathsf{ID}|_i$ 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 $\mathsf{ku}_{\mathsf{ID},T}$ 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 $\mathsf{sk}_{\mathsf{ID}}$ 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). In the scheme, for history-free update, no ancestor's secret value is contained in secret keys. Moreover, the simulator can choose state information P_θ directly (which helps us to prove insider security). Therefore the simulator can generate the secret key of the challenge identity (or its ancestors) directly. So, in the following all games, $\mathcal{B}$ chooses P_θ and stores it as state information.

Let Game_{real} be the real security game. The next game, $\text{Game}_{restricted}$ 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 $\mathcal{A}$ such that $Game_{real}\mathbf{Adv}_{\mathcal{A}} - Game_{restricted}\mathbf{Adv}_{\mathcal{A}} = \epsilon$ Then, we can build an algorithm $\mathcal{B}$ with advantage $\geq \epsilon/2$ in breaking Assumption 2.*

The next game, Game_0 is the same as $\text{Game}_{restricted}$ except that the challenge ciphertext is semi-functional.

Lemma 2. *Suppose there exists an adversary $\mathcal{A}$ such that $Game_{restricted}\mathbf{Adv}_{\mathcal{A}} - Game_0\mathbf{Adv}_{\mathcal{A}} = \epsilon$ Then, we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 1.*

Proof. First, $\mathcal{B}$ is given $(g, X_3, \bar{T})$. $\mathcal{B}$ chooses $\alpha, a_0, \ldots, a_\ell, b \xleftarrow{\$} \mathbb{Z}_n$ 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 $1/|\mathcal{T}|$), and chooses $c, d \xleftarrow{\$} \mathbb{Z}_n$, and sets $u' = g^c$ and $h' = u'^{-T^*} g^d$. $\mathcal{B}$ sends public parameters $\mathsf{mpk}^{\mathsf{mLW}} = \{n, g, h, u_0, \ldots, u_\ell, e(g,g)^\alpha, u', h', X_3\}$ to $\mathcal{A}$.

For a key query $\mathsf{ID}|_j = (\mathsf{I}_0, \ldots, \mathsf{I}_j)$, $\mathcal{B}$ generates a normal secret key key as follows. $\mathcal{B}$ chooses $r_\theta, t_\theta, w_\theta, v_{j_1,\theta}, \ldots, v_\ell \xleftarrow{\$} \mathbb{Z}_n$ for each θ, and computes

$$\Big\{(P_\theta(u_0^{\mathsf{I}_0} \cdots u_j^{\mathsf{I}_j} h)^{r_\theta} X_3^{w_\theta}, g^{r_\theta} X_3^{t_\theta}, u_{j+1}^{r_\theta} X_3^{v_{j+1,\theta}}, \ldots, u_\ell^{r_\theta} X_3^{v_{\ell,\theta}}\Big\}_\theta.$$

Since $\mathcal{B}$ knows α, $\mathcal{B}$ can answer all key update information, secret key, and decryption key queries. $\mathcal{B}$ generates normal values for these queries.

$\mathcal{A}$ sends two messages M_0 and M_1 and the challenge identity $\mathsf{ID}|_{k^*}^* = (\mathsf{I}_0^*, \ldots, \mathsf{I}_{k^*}^*)$ to $\mathcal{B}$. $\mathcal{B}$ chooses $b \xleftarrow{\$} \{0,1\}$, and computes $C' = M_b \cdot e(\bar{T}, g)^\alpha$, $C_0 = \bar{T}^{a_0 \mathsf{I}_0^* + \cdots + a_{k^*} \mathsf{I}_{k^*}^* + b}$, $C_1 = \bar{T}$, and $C_2 = \bar{T}^d$. Here, the $\mathbb{G}_{p_1}$ part of $\bar{T}$ is implicitly set g^s and the $\mathbb{G}_{p_1}$ part of $\bar{T}^d$ is implicitly set $(u'^{T^*} h')^s$ if $\mathcal{B}$'s guessing is correct. If $T \in \mathbb{G}_{p_1 p_2}$, then this ciphertext is semi-functional with

$z_c = a_1 \mathsf{I}_1^* + \cdots + a_{k^*} \mathsf{I}_{k^*}^* + b$. 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. $\square$

For the number of decryption key queries q_{dk}, Game$_k^{\mathsf{dk}}$ ($k \in [1, q_{\mathsf{dk}}]$) is the same as that of Game$_{k-1}^{\mathsf{dk}}$ except that the first k decryption keys are semi-functional and the rest of the keys are normal. We remark that Game$_0^{\mathsf{dk}}$ = Game$_0$.

Lemma 3. *Suppose there exists an adversary $\mathcal{A}$ such that Game$_{k-1}^{\mathsf{dk}}$$\mathbf{Adv}_{\mathcal{A}}$ − Game$_k^{\mathsf{dk}}$$\mathbf{Adv}_{\mathcal{A}} = \epsilon$ Then, we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 2.*

Proof. First, $\mathcal{B}$ is given $(g, X_1X_2, X_3, Y_2Y_3, \bar{T})$. $\mathcal{B}$ chooses $\alpha, a_0, \ldots, a_\ell, b \xleftarrow{\$} \mathbb{Z}_n$ 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 $1/|\mathcal{T}|$), chooses $c, d \xleftarrow{\$} \mathbb{Z}_n$, and sets $u' = g^c$ and $h' = u'^{-T^*} g^d$. $\mathcal{B}$ sends public parameters $\mathsf{mpk}^{\mathsf{mLW}} = \{n, g, h, u_0, \ldots, u_\ell, e(g,g)^\alpha, u', h', X_3\}$ to $\mathcal{A}$. Since $\mathcal{B}$ knows α, $\mathcal{B}$ 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 $(\mathsf{ID}|_j, T)$ where $\mathsf{ID}|_j = (\mathsf{I}_0, \ldots, \mathsf{I}_j)$ and $i < k$, $\mathcal{B}$ generates a semi-functional decryption key. $\mathcal{B}$ chooses $r, z, t, w, w', z_{j+1}, \ldots, z_\ell \xleftarrow{\$} \mathbb{Z}_n$, and computes

$$(g^\alpha (u_0^{\mathsf{I}_0} \cdots u_j^{\mathsf{I}_j} h)^r (u'^T h')^t (Y_2Y_3)^z,\ g^r (Y_2Y_3)^w,\ g^t X_3^{w'},$$

$$u_{j+1}^r (Y_2Y_3)^{z_{j+1}}, \ldots, u_\ell^r (Y_2Y_3)^{z_\ell}).$$

For $i > k$, $\mathcal{B}$ generates a normal decryption key by using α. For $i = k$, $\mathcal{B}$ sets $z_k = a_0 \mathsf{I}_0 + \cdots + a_j \mathsf{I}_j + b$, chooses $t, w, w', w_{j+1}, \ldots, w_\ell \xleftarrow{\$} \mathbb{Z}_n$, and computes $\mathsf{dk}_{\mathsf{ID}|_j, T} = (g^\alpha \bar{T}^{z_k} (u'^T h')^t X_3^w, \bar{T}, g^t X_3^{w'}, \bar{T}^{a_{j+1}} X_3^{w_{j+1}}, \ldots, \bar{T}^{a_\ell} X_3^{w_\ell})$. 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 $\mathsf{ID}|_{k^*}^* = (\mathsf{I}_0^*, \ldots, \mathsf{I}_{k^*}^*)$ to $\mathcal{B}$. $\mathcal{B}$ chooses $b \xleftarrow{\$} \{0,1\}$, and computes $C' = M_b \cdot e(X_1X_2, g)^\alpha$, $C_0 = (X_1X_2)^{a_0 \mathsf{I}_0^* + \cdots + a_{k^*} \mathsf{I}_{k^*}^* + b}$, $C_1 = X_1X_2$, and $C_2 = X_1^d$. Here, X_1 is implicitly set g^s and $z_c = a_0 \mathsf{I}_0^* + \cdots + a_{k^*} \mathsf{I}_{k^*}^* + b$. If $\bar{T} \in \mathbb{G}_{p_1p_3}$, then $\mathcal{B}$ has properly simulated Game$_{k-1}^{\mathsf{dk}}$. If $\bar{T} \in \mathbb{G}$, then $\mathcal{B}$ has properly simulated Game$_k^{\mathsf{dk}}$.

If $\mathcal{B}$ tries to test whether the answer of k-th decryption key query $\mathsf{dk}_{\mathsf{ID}|_j, T}$ 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 $\mathsf{dk}_{\mathsf{ID}|_j, T}$ is nominally semi-functional since $z_k = z_c$. So, we have to consider the case that key update information $\mathsf{ku}_{\mathsf{ID}|_j, T}$ is generated by this $\mathsf{dk}_{\mathsf{ID}|_j, T}$ (then $\mathsf{ku}_{\mathsf{ID}|_j, T}$ is semi-functional), and $\mathsf{dk}_{\mathsf{ID}|_{j+1}, T}$ is computed by $\mathsf{ku}_{\mathsf{ID}|_j, T}$ and a normal secret key $\mathsf{sk}_{\mathsf{ID}|_{j+1}}$. $\mathsf{ku}_{\mathsf{ID}|_j, T}$ is represented as

$$\Big\{ (P_\theta^{-1} g^\alpha \bar{T}^{z_k} (u_0^{\mathsf{I}_0} \cdots u_j^{\mathsf{I}_j} h)^{r_\theta} (u'^T h')^{t + t_\theta} X_3^{w + s_{0,\theta}},\ \bar{T} g^{r_\theta} X_3^{s_{1,\theta}},\ g^{t + t_\theta} X_3^{w' + s_{2,\theta}},$$

$$\bar{T}^{a_{j+1}} u_{j+1}^{r_\theta} X_3^{w_{j+1}+s'_{j+1,\theta}}, \ldots, \bar{T}^{a_\ell} u_\ell^{r_\theta} X_3^{w_\ell+s'_{\ell,\theta}})\Big\}_\theta$$

where $r_\theta, t_\theta, s_{0,\theta}, s_{1,\theta}, s_{2,\theta}, s'_{j+1,\theta}, \ldots, s'_{\ell,\theta} \xleftarrow{\$} \mathbb{Z}_n$ are chosen for each θ. Then since a normal secret key $\mathsf{sk}_{\mathsf{ID}|_{j+1}}$ is represented as

$$\Big\{(P_\theta(u_0^{\mathsf{I}_0} \cdots u_{j+1}^{\mathsf{I}_{j+1}} h)^{r'_\theta} R'_{3,\theta}, g^{r'_\theta} R_{3,\theta}, u_{j+1}^{r'_\theta} R_{j+1,\theta}, \ldots, u_\ell^{r'_\theta} R_{\ell,\theta})\Big\}_\theta,$$

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

$$g^\alpha \bar{T}^{z_k + a_{j+1}\mathsf{I}_{j+1}} (u_0^{\mathsf{I}_0} \cdots u_{j+1}^{\mathsf{I}_{j+1}} h)^{r_\theta + r'_\theta + r'} (u'^T h')^{t+t_\theta+t'} (X_3^{s_{0,\theta}+(w_{j+1}+s'_{j+1,\theta})\mathsf{I}_{j+1}+s_0} R'_{3,\theta})$$

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

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

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

Lemma 4. *Suppose there exists an adversary $\mathcal{A}$ such that Game$_{k-1}^{\mathsf{ku}}$*$\mathbf{Adv}_\mathcal{A}$ − *Game$_k^{\mathsf{ku}}$*$\mathbf{Adv}_\mathcal{A} = \epsilon$ *Then, we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 2.*

Proof. First, $\mathcal{B}$ is given $(g, X_1X_2, X_3, Y_2Y_3, \bar{T})$. $\mathcal{B}$ chooses $\alpha, a_0, \ldots, a_\ell, b \xleftarrow{\$} \mathbb{Z}_n$ 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 $1/|\mathcal{T}|$), chooses $c, d \xleftarrow{\$} \mathbb{Z}_n$, and sets $u' = g^c$ and $h' = u'^{-T^*} g^d$. $\mathcal{B}$ sends public parameters $\mathsf{mpk}^{\mathsf{mLW}} = \{n, g, h, u_0, \ldots, u_\ell, e(g,g)^\alpha, u', h', X_3\}$ to $\mathcal{A}$. Since $\mathcal{B}$ knows α, $\mathcal{B}$ can answer all secret key and decryption key queries. $\mathcal{B}$ generates normal secret keys. Moreover, $\mathcal{B}$ computes semi-functional decryption keys of $(\mathsf{ID}|_j, T)$ where $\mathsf{ID}|_j = (\mathsf{I}_0, \ldots, \mathsf{I}_j)$ as folows. $\mathcal{B}$ chooses $r, z, t, w, w', z_{j+1,\theta}, \ldots, z_{\ell,\theta} \xleftarrow{\$} \mathbb{Z}_n$, and computes

$$(g^\alpha (u_0^{\mathsf{I}_0} \cdots u_j^{\mathsf{I}_j} h)^r (u'^T h')^t (Y_2Y_3)^z,\ g^r (Y_2Y_3)^w,\ g^t X_3^{w'},$$
$$u_{j+1}^r (Y_2Y_3)^{z_{j+1}}, \ldots, u_\ell^r (Y_2Y_3)^{z_\ell}).$$

For the i-th key update information query $(\mathsf{ID}|_j, T)$ where $\mathsf{ID}|_j = (\mathsf{I}_0, \ldots, \mathsf{I}_j)$ and $i < k$, $\mathcal{B}$ generates semi-functional key update information $\mathsf{ku}_{\mathsf{ID}|_j,T}$ as follows. $\mathcal{B}$ chooses $r_\theta, z_\theta, t_\theta, w_\theta, w'_\theta, z_{j+1,\theta}, \ldots, z_{\ell,\theta} \xleftarrow{\$} \mathbb{Z}_n$ for each θ, and computes

$$\Big\{(P_\theta^{-1} g^\alpha (u_0^{\mathsf{I}_0} \cdots u_j^{\mathsf{I}_j} h)^{r_\theta} (u'^T h')^{t_\theta} (Y_2Y_3)^{z_\theta},\ g^{r_\theta} (Y_2Y_3)^{w_\theta},\ g^{t_\theta} X_3^{w'_\theta},$$

$$u_{j+1}^{r_\theta}(Y_2Y_3)^{z_{j+1,\theta}}, \ldots, u_\ell^{r_\theta}(Y_2Y_3)^{z_{\ell,\theta}})\Big\}_\theta.$$

For $i > k$, $\mathcal{B}$ generates normal key update information by using α. For $i = k$, $\mathcal{B}$ sets $z_k = a_0\mathsf{I}_0 + \cdots + a_j\mathsf{I}_j + b$, chooses $\gamma_\theta, t_\theta, w_\theta, w'_\theta, w_{j+1,\theta}, \ldots, w_{\ell,\theta} \xleftarrow{\$} \mathbb{Z}_n$, and computes $\mathsf{ku}_{\mathsf{ID}|_j,T}$ as

$$\Big\{(P_\theta^{-1}g^\alpha \bar{T}^{\gamma_\theta z_k}(u'^T h')^{t_\theta} X_3^{w_\theta},\ \bar{T}^{\gamma_\theta},\ g^{t_\theta}X_3^{w'_\theta},$$

$$\bar{T}^{\gamma_\theta a_{j+1}}X_3^{w_{j+1,\theta}}, \ldots, \bar{T}^{\gamma_\theta a_\ell}X_3^{w_{\ell,\theta}})\Big\}_\theta.$$

If $\bar{T} \in \mathbb{G}_{p_1p_3}$, then this key is normal where the $\mathbb{G}_{p_1}$ part of $\bar{T}^{\gamma_\theta}$ is set as g^{r_θ}. 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 $\mathsf{ID}|^*_{k^*} = (\mathsf{I}^*_0, \ldots, \mathsf{I}^*_{k^*})$ to $\mathcal{B}$. $\mathcal{B}$ chooses $b \xleftarrow{\$} \{0,1\}$, and computes $C' = M_b \cdot e(X_1X_2, g)^\alpha$, $C_0 = (X_1X_2)^{a_0\mathsf{I}^*_0+\cdots+a_{k^*}\mathsf{I}^*_{k^*}+b}$, $C_1 = X_1X_2$, and $C_2 = X_1^d$. Here, X_1 is implicitly set g^s and $z_c = a_0\mathsf{I}^*_0+\cdots+a_{k^*}\mathsf{I}^*_{k^*}+b$. If $\bar{T} \in \mathbb{G}_{p_1p_3}$, then $\mathcal{B}$ has properly simulated Game$^{\mathsf{ku}}_{k-1}$. If $\bar{T} \in \mathbb{G}$, then $\mathcal{B}$ has properly simulated Game$^{\mathsf{ku}}_k$.

As in Game$^{\mathsf{ku}}_k$, we can easily check $\mathsf{dk}_{\mathsf{ID}|_{j+1},T}$ is a nominally semi-functional decryption key in the case that $\mathsf{dk}_{\mathsf{ID}|_{j+1},T}$ is computed by $\mathsf{ku}_{\mathsf{ID}|_j,T}$ and a normal secret key $\mathsf{sk}_{\mathsf{ID}|_{j+1}}$. So, $\mathcal{B}$ uses the output of $\mathcal{A}$ for breaking Assumption 2. □

For the number of secret key queries q_{sk}, Game$^{\mathsf{sk}}_k$ ($k \in [1, q_{\mathsf{sk}}]$) is the same as that of Game$^{\mathsf{sk}}_{k-1}$ except that the first k secret keys are semi-functional and the rest of keys are normal. We remark that Game$^{\mathsf{sk}}_0$ = Game$^{\mathsf{ku}}_{q_{\mathsf{ku}}}$.

Lemma 5. *Suppose there exists an adversary $\mathcal{A}$ such that Game$^{\mathsf{sk}}_{k-1}$$\mathbf{Adv}_\mathcal{A}$ − Game$^{\mathsf{sk}}_k$$\mathbf{Adv}_\mathcal{A}$ = ϵ Then, we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 2.*

Proof. First, $\mathcal{B}$ is given $(g, X_1X_2, X_3, Y_2Y_3, \bar{T})$. $\mathcal{B}$ chooses $\alpha, a_0, \ldots, a_\ell, b \xleftarrow{\$} \mathbb{Z}_n$ 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 $1/|\mathcal{T}|$), chooses $c, d \xleftarrow{\$} \mathbb{Z}_n$, and sets $u' = g^c$ and $h' = u'^{-T^*}g^d$. $\mathcal{B}$ sends public parameters $\mathsf{mpk}^{\mathsf{mLW}} = \{n, g, h, u_0, \ldots, u_\ell, e(g,g)^\alpha, u', h', X_3\}$ to $\mathcal{A}$.

For a decryption key query $(\mathsf{ID}|_j, T)$ where $\mathsf{ID}|_j = (\mathsf{I}_0, \ldots, \mathsf{I}_j)$, $\mathcal{B}$ computes a semi-functional decryption key $\mathsf{dk}_{\mathsf{ID}|_j,T}$ as follows. $\mathcal{B}$ sets $z_k = a_0\mathsf{I}_0+\cdots+a_j\mathsf{I}_j+b$, and chooses $r, z, t, w, w', z_{j+1,\theta}, \ldots, z_{\ell,\theta} \xleftarrow{\$} \mathbb{Z}_n$, and computes

$$(g^\alpha(u_0^{\mathsf{I}_0}\cdots u_j^{\mathsf{I}_j}h)^r(u'^T h')^t(Y_2Y_3)^{zz_k},\ g^r(Y_2Y_3)^{wz_k},\ g^tX_3^{w'},$$

$$u_{j+1}^r(Y_2Y_3)^{z_{j+1}z_k}, \ldots, u_\ell^r(Y_2Y_3)^{z_\ell z_k}).$$

Similarly, for a key update information query $(\mathsf{ID}|_j, T)$ where $\mathsf{ID}|_j = (\mathsf{I}_0, \ldots, \mathsf{I}_j)$, $\mathcal{B}$ computes semi-functional key update information $\mathsf{ku}_{\mathsf{ID}|_j,T}$ as follows. $\mathcal{B}$ sets $z_k = a_0\mathsf{I}_0 + \cdots + a_j\mathsf{I}_j + b$, and chooses $r_\theta, z_\theta, t_\theta, w_\theta, w'_\theta, z_{j+1,\theta}, \ldots, z_{\ell,\theta} \xleftarrow{\$} \mathbb{Z}_n$ for each θ, and computes

$$\Big\{(P_\theta^{-1} g^\alpha (u_0^{\mathsf{I}_0} \cdots u_j^{\mathsf{I}_j} h)^{r_\theta} (u'^T h')^{t_\theta} (Y_2Y_3)^{z_\theta z_k},\ g^{r_\theta}(Y_2Y_3)^{w_\theta z_k},\ g^{t_\theta} X_3^{w'_\theta},$$

$$u_{j+1}^{r_\theta}(Y_2Y_3)^{z_{j+1,\theta} z_k}, \ldots, u_\ell^{r_\theta}(Y_2Y_3)^{z_{\ell,\theta} z_k})\Big\}_\theta.$$

For the i-th secret key query $\mathsf{ID}|_j$ where $\mathsf{ID}|_j = (\mathsf{I}_0, \ldots, \mathsf{I}_j)$ and $i < k$, $\mathcal{B}$ generates a semi-functional secret key $\mathsf{sk}_{\mathsf{ID}|_j}$ as follows. $\mathcal{B}$ sets $z_k = a_0\mathsf{I}_0 + \cdots + a_j\mathsf{I}_j + b$, chooses $r_\theta, z_\theta, t_\theta, w_\theta, w'_\theta, z_{j+1,\theta}, \ldots, z_{\ell,\theta} \xleftarrow{\$} \mathbb{Z}_n$ for each θ, and computes

$$\Big\{(P_\theta (u_0^{\mathsf{I}_0} \cdots u_j^{\mathsf{I}_j} h)^{r_\theta} (u'^T h')^{t_\theta} (Y_2Y_3)^{z_\theta z_k},\ g^{r_\theta}(Y_2Y_3)^{w_\theta z_k},\ g^{t_\theta} X_3^{w'_\theta},$$

$$u_{j+1}^{r_\theta}(Y_2Y_3)^{z_{j+1,\theta} z_k}, \ldots, u_\ell^{r_\theta}(Y_2Y_3)^{z_{\ell,\theta} z_k})\Big\}_\theta.$$

For $i > k$, $\mathcal{B}$ generates normal secret keys. For $i = k$, $\mathcal{B}$ sets $z_k = a_0\mathsf{I}_0 + \cdots + a_j\mathsf{I}_j + b$, chooses $\gamma_\theta, t_\theta, w_\theta, w'_\theta, w_{j+1,\theta}, \ldots, w_{\ell,\theta} \xleftarrow{\$} \mathbb{Z}_n$, and computes $\mathsf{sk}_{\mathsf{ID}|_j}$ as

$$\Big\{(P_\theta \bar{T}^{\gamma_\theta z_k} (u'^T h')^{t_\theta} X_3^{w_\theta}, \bar{T}^{\gamma_\theta}, g^{t_\theta} X_3^{w'_\theta}, \bar{T}^{\gamma_\theta a_{j+1}} X_3^{w_{j+1,\theta}}, \ldots, \bar{T}^{\gamma_\theta a_\ell} X_3^{w_{\ell,\theta}})\Big\}_\theta.$$

If $\bar{T} \in \mathbb{G}_{p_1p_3}$, then this key is normal where the $\mathbb{G}_{p_1}$ part of $\bar{T}^{\gamma_\theta}$ is set as g^{r_θ}. 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 $\mathsf{ID}|_{k^*}^* = (\mathsf{I}_0^*, \ldots, \mathsf{I}_{k^*}^*)$ to $\mathcal{B}$. $\mathcal{B}$ chooses $b \xleftarrow{\$} \{0,1\}$, and computes $C' = M_b \cdot e(X_1X_2, g)^\alpha$, $C_0 = (X_1X_2)^{a_0\mathsf{I}_0^* + \cdots + a_{k^*}\mathsf{I}_{k^*}^* + b}$, $C_1 = X_1X_2$, and $C_2 = X_1^d$. Here, X_1 is implicitly set g^s and $z_c = a_0\mathsf{I}_0^* + \cdots + a_{k^*}\mathsf{I}_{k^*}^* + b$. If $\bar{T} \in \mathbb{G}_{p_1p_3}$, then $\mathcal{B}$ has properly simulated $\text{Game}_{k-1}^{\mathsf{sk}}$. If $\bar{T} \in \mathbb{G}$, then $\mathcal{B}$ has properly simulated $\text{Game}_k^{\mathsf{sk}}$.

If $\mathcal{B}$ tries to test whether the answer of k-th secret key query $\mathsf{sk}_{\mathsf{ID}|_j}$ is semi-functional or not by computing a semi-functional ciphertext, $\mathcal{B}$ can only create a nominally semi-functional decryption key with $z_k = z_c$. Let us consider the case that a decryption key $\mathsf{dk}_{\mathsf{ID}|_j,T}$ is computed by semi-functional key update information $\mathsf{ku}_{\mathsf{ID}|_{j-1},T}$ and $\mathsf{sk}_{\mathsf{ID}|_j}$, where $\mathsf{ku}_{\mathsf{ID}|_{j-1},T}$ is defined as

$$\Big\{(P_\theta^{-1} g^\alpha (u_0^{\mathsf{I}_0} \cdots u_{j-1}^{\mathsf{I}_{j-1}} h)^{r'_\theta} (u'^T h')^{t'_\theta} (Y_2Y_3)^{z'_\theta z_k},\ g^{t_\theta} X_3^{w'_\theta},$$

$$u_j^{r'_\theta}(Y_2Y_3)^{z_{j,\theta} z_k}, \ldots, u_\ell^{r'_\theta}(Y_2Y_3)^{z_{\ell,\theta} z_k})\Big\}.$$

For some θ, the first component of $\mathsf{dk}_{\mathsf{ID}|_j,T}$ output by the DKGen algorithm is represented as

$$g^\alpha \bar{T}^{\gamma_\theta z_k} (u_0^{\mathsf{I}_0} \cdots u_j^{\mathsf{I}_j} h)^{r'_\theta + r_\theta} (u'^T h')^{t'_\theta + t_\theta} (Y_2Y_3)^{z_k(z'_\theta + z_{j,\theta}\mathsf{I}_j)}$$

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, $\mathsf{dk}_{\mathsf{ID}|_j,T}$ is a nominally semi-functional decryption key.

Similarly, we have to care about the case that $\mathsf{ku}_{\mathsf{ID}|_{j-1},T}$ is computed a semi-functional decryption key $\mathsf{dk}_{\mathsf{ID}|_{j-1},T}$, and $\mathsf{dk}_{\mathsf{ID}|_j,T}$ is computed by $\mathsf{ku}_{\mathsf{ID}|_{j-1},T}$ and $\mathsf{sk}_{\mathsf{ID}|_j}$. Since $\mathsf{ku}_{\mathsf{ID}|_{j-1},T}$ is computed by multiplicating P_θ^{-1} to $\mathsf{dk}_{\mathsf{ID}|_j,T}$ and the re-randomization process is independent of the $\mathbb{G}_{p_2}$ part, $\mathsf{dk}_{\mathsf{ID}|_j,T}$ is a nominally semi-functional decryption key.

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

The next game, Game_{final} is the same as $\mathrm{Game}^{\mathsf{sk}}_{q_{\mathsf{sk}}}$ 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 $\mathcal{A}$ such that $Game^{\mathsf{sk}}_{q_{\mathsf{sk}}}\mathbf{Adv}_{\mathcal{A}} - Game_{final}\mathbf{Adv}_{\mathcal{A}} = \epsilon$ Then, we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption* 3.

Proof. First, $\mathcal{B}$ is given $(g, g^\alpha X_2, X_3, g^s Y_2, Z_2, \bar{T})$. $\mathcal{B}$ chooses $a_0, \ldots, a_\ell, b \xleftarrow{\$} \mathbb{Z}_n$ 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, $\mathcal{B}$ guesses T^* (with success probability at least $1/|\mathcal{T}|$), chooses $c, d \xleftarrow{\$} \mathbb{Z}_n$, and sets $u' = g^c$ and $h' = u'^{-T^*} g^d$. $\mathcal{B}$ sends public parameters $\mathsf{mpk}^{\mathsf{mLW}} = \{n, g, h, u_0, \ldots, u_\ell, e(g,g)^\alpha, u', h', X_3\}$ to $\mathcal{A}$.

For a secret key query $\mathsf{ID}|_j = (\mathsf{I}_0, \ldots, \mathsf{I}_j)$, $\mathcal{B}$ generates a semi-functional secret key as follows. $\mathcal{B}$ chooses $c_\theta, r_\theta, t_\theta, w_\theta, z_\theta, z_{j+1,\theta}, \ldots, z_{\ell,\theta}, w_{j+1,\theta}, \ldots, w_{\ell,\theta} \xleftarrow{\$} \mathbb{Z}_n$ for each θ, and computes $\mathsf{sk}_{\mathsf{ID}|_j}$ as

$$\Big\{(P_\theta Z_2^{c_\theta}(u_0^{\mathsf{I}_0}\cdots u_j^{\mathsf{I}_j} h)^{r_\theta} X_3^{w_\theta},\ g^{r_\theta} Z_2^{z_\theta} X_3^{t_\theta},$$

$$u_{j+1}^{r_\theta} Z_2^{z_{j+1,\theta}} X_3^{w_{j+1,\theta}}, \ldots, u_\ell^{r_\theta} Z_2^{z_{\ell,\theta}} X_3^{w_{\ell,\theta}})\Big\}_\theta.$$

For a key update information query $(\mathsf{ID}|_j, T)$ where $\mathsf{ID}|_j = (\mathsf{I}_0, \ldots, \mathsf{I}_j)$, $\mathcal{B}$ generates semi-functional key update information as follows. $\mathcal{B}$ chooses $c_\theta, r_\theta, t_\theta, t'_\theta, w_\theta, w'_\theta, z_\theta, z_{j+1,\theta}, \ldots, z_{\ell,\theta}, w_{j+1,\theta}, \ldots, w_{\ell,\theta} \xleftarrow{\$} \mathbb{Z}_n$ for each θ, and computes $\mathsf{ku}_{\mathsf{ID}|_j,T}$ as

$$\Big\{(P_\theta^{-1} g^\alpha X_2 Z_2^{c_\theta}(u_0^{\mathsf{I}_0}\cdots u_j^{\mathsf{I}_j} h)^{r_\theta}(u'^T h')^{t_\theta} X_3^{w_\theta},\ g^{r_\theta} Z_2^{z_\theta} X_3^{t'_\theta},\ g^{t_\theta} X_3^{w'_\theta},$$

$$u_{j+1}^{r_\theta} Z_2^{z_{j+1,\theta}} X_3^{w_{j+1,\theta}}, \ldots, u_\ell^{r_\theta} Z_2^{z_{\ell,\theta}} X_3^{w_{\ell,\theta}})\Big\}_\theta.$$

For a decryption key query $(\mathsf{ID}|_j, T)$ where $\mathsf{ID}|_j = (\mathsf{I}_0, \ldots, \mathsf{I}_j)$, $\mathcal{B}$ generates semi-functional decryption key as follows. $\mathcal{B}$ chooses $c, r, t, t', w, w', z, z_{j+1}, \ldots, z_\ell, w_{j+1}, \ldots, w_\ell \xleftarrow{\$} \mathbb{Z}_n$, and computes $\mathsf{dk}_{\mathsf{ID}|_j,T}$ as

$$(g^\alpha X_2 Z_2^c(u_0^{\mathsf{I}_0}\cdots u_j^{\mathsf{I}_j} h)^r(u'^T h')^t X_3^w,\ g^r Z_2^z X_3^{t'},\ g^t X_3^{w'},$$

$$u_{j+1}^{r} Z_2^{z_{j+1}} X_3^{w_{j+1}}, \ldots, u_\ell^{r_\theta} Z_2^{z_\ell} X_3^{w_\ell}).$$

$\mathcal{A}$ sends two messages M_0 and M_1 and the challenge identity $\mathsf{ID}|_{k^*}^* = (\mathsf{I}_0^*, \ldots, \mathsf{I}_{k^*}^*)$ to $\mathcal{B}$. $\mathcal{B}$ chooses $b \xleftarrow{\$} \{0,1\}$, and computes $C' = M_b \cdot \bar{T}$, $C_0 = (g^s Y_2)^{a_0 \mathsf{I}_0^* + \cdots + a_{k^*} \mathsf{I}_{k^*}^* + b}$, $C_1 = g^s Y_2$, 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 $\mathbb{G}_T$, then this is a semi-functional ciphertext of a random message. So, $\mathcal{B}$ uses the output of $\mathcal{A}$ for breaking Assumption 3. □

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

1. Boldyreva, A., Goyal, V., Kumar, V.: Identity-based encryption with efficient revocation. In: ACM CCS, pp. 417–426 (2008)
2. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
3. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
4. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. SIAM J. Comput. **32**(3), 586–615 (2003)
5. Lewko, A.: Tools for simulating features of composite order bilinear groups in the prime order setting. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 318–335. Springer, Heidelberg (2012)
6. Lewko, A., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010)
7. Libert, B., Vergnaud, D.: Adaptive-ID secure revocable identity-based encryption. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 1–15. Springer, Heidelberg (2009)
8. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
9. Seo, J.H., Emura, K.: Efficient delegation of key generation and revocation functionalities in identity-based encryption. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 343–358. Springer, Heidelberg (2013)
10. Seo, J.H., Emura, K.: Revocable identity-based encryption revisited: security model and construction. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 216–234. Springer, Heidelberg (2013)

11. Seo, J.H., Emura, K.: Revocable hierarchical identity-based encryption. Theor. Comput. Sci. **542**, 44–62 (2014)
12. Seo, J.H., Emura, K.: Revocable identity-based cryptosystem revisited: Security models and constructions. IEEE Trans. Inf. Forensics Secur. **9**(7), 1193–1205 (2014)
13. Seo, J.H., Emura, K.: Revocable hierarchical identity-based encryption: history-free update, security against insiders, and short ciphertexts. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 106–123. Springer, Heidelberg (2015)
14. Tsai, T., Tseng, Y., Wu, T.: RHIBE: constructing revocable hierarchical ID-based encryption from HIBE. Informatica Lith. Acad. Sci. **25**(2), 299–326 (2014)
15. Waters, B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)

http://www.springer.com/978-3-319-22424-4

Advances in Information and Computer Security
10th International Workshop on Security, IWSEC 2015,
Nara, Japan, August 26-28, 2015, Proceedings
Tanaka, K.; Suga, Y. (Eds.)
2015, XIII, 357 p. 51 illus., Softcover
ISBN: 978-3-319-22424-4