Generic Construction of Server-Aided Revocable Hierarchical Identity-Based Encryption with Decryption Key Exposure Resistance

Yanyan Liu[1,2], Yiru Sun[1,2]

1 State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
2 School of Cyber Security, University of Chinese Academy of Sciences, Beijing 101408, China
{liuyanyan,sunyiru}@iie.ac.cn

Abstract. In this paper, we extend the notion of server-aided revocable identity-based encryption (SR-IBE) to the hierarchical IBE (HIBE) setting and propose a generic construction of server-aided revocable hierarchical IBE (SR-HIBE) schemes *with* decryption key exposure resistance (DKER) from any (weak) L-level revocable HIBE scheme *without* DKER and $(L+1)$-level HIBE scheme. In order to realize the server-aided revocation mechanism, we use the "double encryption" technique, and this makes our construction has short ciphertext size. Furthermore, when the maximum hierarchical depth is one, we obtain a generic construction of SR-IBE schemes *with* DKER from *any* IBE scheme and two-level HIBE scheme.

Keywords: generic construction, server-aided revocation mechanism, hierarchical identity-based encryption, decryption key exposure resistance

1 Introduction

Identity-based encryption (IBE), which was proposed by Shamir [37] in 1984, provides a public key encryption mechanism that an arbitrary string representing user's identities (e.g., email address, ID number) can be used as public keys. As an extension of IBE, hierarchical identity-based encryption (HIBE) supports key delegation functionality. It has been well studied on (H)IBE [1, 3, 4, 6, 9, 11, 42, 43, 40, 44] since after Boneh and Franklin [5] gave the first IBE scheme in 2001. As for many multi-user cryptosystems, adding an efficient revocation mechanism to the (H)IBE scheme to revoke the malicious user is a necessary problem. Boneh and Franklin [5] gave a naive approach method to realize revocation that every non-revoked users need to update their secret key by communicating with the key generation center (KGC) per time period secretly. This method is too inefficient since the workloads of the KGC is linearly in the number of users N.

In 2008, Boldyreva et al. [2] introduced the notion of revocable IBE (RIBE), and gave an efficient RIBE scheme by combining fuzzy identity-based encryption [31] and the tree-based revocation technique in [25]. In their definition, each user can issue a long-term secret key corresponding to his identity, but only using this secret key cannot decrypt the ciphertext. In order to realize the revocation, the KGC broadcasts the key update per time period, and the non-revoked users can obtain their decryption key by combining the secret key and the key update, while the revoked user gains nothing. In their work, the size of the key update in each time period is logarithmic in N (i.e., $\mathcal{O}(r \log \frac{N}{r})$, where r is the number of the revoked users). Followed by the Boldyreva et al.'s framework, there emerged numerous works on R(H)IBE constructions [7, 10, 14, 15, 20, 16, 18, 19, 27, 30, 33, 34, 36, 41]. Nevertheless, in their model [2], the non-revoked users still need to communicate with the KGC per time period to update their decryption key, and this leads to a heavy workload on every user.

In order to resolve the above problem, Qin et al. [28] introduced server-aided RIBE (SR-IBE) and gave a pairing-based instantiation of SR-IBE scheme. In the SR-IBE scheme, almost all the workloads of users are outsourced to an untrusted server, and the users can compute their decryption keys at any time period by themselves. The server can be unstrusted since it does not keep any secret, and the only requirement is computing correctly. More specifically, the SR-IBE scheme works as follows. When a user registers to the system, the KGC generates a "public key" which corresponding to the long-term secret key in RIBE scheme, and sends it to the server through public channel. The key updates are only sent to the server rather than to all users. The ciphertexts are also forwarded to the server, and the latter transforms them to "partial decrypted ciphertexts" which are sent to the intended recipients. The recipients can recover the message using the decryption key generated from their private key. Later on, Nguyen et al. [26] gave the first lattice-based SR-IBE scheme by combining the ideas of Chen et al.'s RIBE scheme [7] and the HIBE scheme in [1]. In 2018, Hu et al. [12] gave a non-black-box construction of SR-IBE from computational Diffie-Hellman assumption.

Considering decryption key leakage that caused by unexpected human errors, Seo and Emura introduced a security notion called decryption key exposure resistance (DKER) to RIBE [33] and RHIBE [36], respectively. This security guarantees that an exposure of a user's decryption key at some time will not compromise the confidentiality of ciphertexts that are encrypted for different time period. After the notion of DKER proposed, many works that viewing it as a default security requirements for R(H)IBE scheme emerged [10, 14–16, 18, 19, 41, 30, 36, 38].

Observe that in the RHIBE scheme, the secret key of a user can be divided into two parts: one part is its own secret key which used to realize key revocation by combining with the key update information, the second part is used to realize key delegation for its descendants. Thus, the workloads of users in RHIBE are much heavier than that in RIBE. So it has practical interest to add a server-aided revocation mechanism to the RHIBE setting to reduce the user's heavy workload. Note that Ma and Lin's generic construction of RIBE scheme with DKER is natural to be server-aided [23]. In their works, the server gains and sends a random chosen message to the recipient. While the server may be untrusted, if he sends a random string which is not what he obtained, then the recipient cannot obtain the real message and may have no idea of that. Inspired by the above observation and the works of [28],[26] and [23], we focus on giving a generic construction of SR-HIBE scheme that can guarantee both the integrity and privacy of message. For security aspect, we also consider the DKER property on our generic construction of SR-HIBE.

Note that RHIBE should support both key revocation and key delegation functionalities, so we cannot transform all the computations of users to the server when considering the sever-aided revocation mechanism in RHIBE. While it is still meaningful to add a server-aided revocation mechanism to RHIBE since a parent node may manage a large number of the users, and the workloads of the secret key are also very heavy for the user.

Our contributions. In this paper, we add a server-aided revocation mechanism to the HIBE scheme and propose a generic construction of SR-HIBE scheme with DKER. Our contributions are as follows:

First, we extend the notion of SR-IBE to the SR-HIBE case, and give a formal definition of SR-HIBE scheme. Additionally, we also give a rigorous security definition of SR-HIBE by modifying that of RHIBE in [15]. To the best of our knowledge, it is the first time to realize the server-aided revocation mechanism in the HIBE setting.

Second, we propose a generic construction of SR-HIBE scheme with DKER from any L-level RHIBE without DKER and $(L+1)$-level HIBE. In our construction, the decryption key size is equal to that of the underlying HIBE scheme, and the ciphertext size is the same as that of the underlying RHIBE scheme. Our generic construction possesses DKER property since the decryption key is a secret key of the HIBE scheme which has a property that disclosure of the secret key of a user ID will not compromise his ancestors' secret key. To show the advantages of our approach, we give a detailed comparison of our construction with some non-server-aided revocable HIBE schems [32, 35] in Table 1.

Third, our construction implies a generic transformation from any IBE and two-level HIBE to SR-IBE scheme with DKER by combining with Ma and Lin's work that the generic construction of RIBE from any IBE. Compared with the SR-IBE scheme with

Table 1. Comparison with non-server-aided revocable HIBE schemes.

Schemes	PP size	PK size	SK size	KU size	CT size	DKER resis.	Dec. cost	Assumption
SE[32]	$\mathcal{O}(L)$	-	$\mathcal{O}(L^2 \log N)$	$\mathcal{O}(Lr \log \frac{N}{r})^{\dagger}$	$\mathcal{O}(L)$	$\times$	$\mathcal{O}(L)$	static
SE[35]	$\mathcal{O}(L)$	-	$\mathcal{O}(L \log N)$	$\mathcal{O}(Lr \log \frac{N}{r})^{\dagger}$	$\mathcal{O}(1)$	✓	$\mathcal{O}(1)$	q-type
Ours	$\mathcal{O}(L)$	$\mathcal{O}(L^2 \log N)$	$\mathcal{O}(L)$	$\mathcal{O}(Lr \log \frac{N}{r})^{\ddagger}$	$\mathcal{O}(L)$	✓	$\mathcal{O}(L)$	static

Let N be the maximum number of users in each level, L the maximum hierarchical level and r the number of revoked users. † means that this item is sent to the users, and ‡ represents that this item is sent to the server. "-" denotes that this item does not considered in the scheme. We initialize our generic construction of SR-HIBE by using the RHIBE scheme in [32] and the HIBE scheme derived from the Waters IBE scheme [42].

DKER in [23], our construction has shorter ciphertext size and can guarantee both the integrity and privacy of messages. If the server sends something that is different from what he obtained, then the recipient cannot decrypt and thus can detect this dishonest behavior.

Technique overview. At a high level, in our generic construction, we use an RHIBE scheme with L levels and a $(L+1)$-level HIBE scheme as building blocks, together with the "double encryption" technique.

However, looking into the details, it is not straightforward to obtain our generic construction from those two building blocks. As we mentioned before, the secret key of a user in RHIBE scheme contains two parts: one part is its own secret key, the other one is used to realize key delegation for its descendants, so we cannot send the whole secret key as a public key to the server just as in SR-IBE. We should be careful to deal with the secret key in RHIBE so that our construction can work smoothly.

In our work, we overcome those above problems as follows. Firstly, we use the stateful version of the definition of RHIBE scheme used in [15] and separate the secret key $\mathsf{r.sk_{ID}}$ generated by $\mathsf{r.GenSK}$ into two part: $(\mathsf{r.sk_{ID}}, \mathsf{r.msk_{ID}})$, where the first part is a secret key while the other one is a master secret key for ID which is used to generate the master secret key for its descendants. Note that this modification does not change the syntax definition of RHIBE scheme since it is just a form change, while this is very useful for our generic construction. So in our SR-HIBE construction, users only sends $\mathsf{r.sk_{ID}}$ to the server as public key while keeping $\mathsf{r.msk_{ID}}$ as secret. Secondly, we use a $(L+1)$-level HIBE scheme that any user ID with secret key $\mathsf{h.sk_{ID}}$ can delegate a decryption key $\mathsf{h.sk_{ID,t}}$ for $(\mathsf{ID}, \mathsf{t})$.Thirdly, using "double encryption" technique makes our construction can work on all settings, guarantee both the integrity and privacy of messages and has shorter ciphertext size.

Construction scratch. At the setup stage, the KGC runs the setup algorithm of RHIBE and HIBE schemes to generate the public parameter $\mathsf{mpk_{kgc}} := (\mathsf{r.mpk_{kgc}}, \mathsf{h.mpk_{kgc}})$ and a master secret key $\mathsf{msk_{kgc}} := (\mathsf{r.msk_{kgc}}, \mathsf{h.msk_{kgc}})$, where $(\mathsf{r.mpk_{kgc}}, \mathsf{r.msk_{kgc}})$ and $(\mathsf{h.mpk_{kgc}}, \mathsf{h.msk_{kgc}})$ denote the public parameter and the master secret key of RHIBE scheme and the HIBE scheme, respectively. When a user ID registers to join the system, the "delegated KGC" – the user's direct parent $\mathsf{pa(ID)}$ generates the public key $\mathsf{pk_{ID}} := \mathsf{r.sk_{ID}}$ and the secret key $\mathsf{sk_{ID}} := (\mathsf{r.msk_{ID}}, \mathsf{h.sk_{ID}})$, and sends $\mathsf{pk_{ID}}$ to the server through the public channel and $\mathsf{sk_{ID}}$ to the user with ID by secret channel. At each time period, the user $\mathsf{pa(ID)}$ generates and sends the update key $\mathsf{uk_{pa(ID),t}}$ to the server, and the latter can obtain the transform key $\mathsf{tk_{ID,t}}$ which is corresponding to the decryption key in RHIBE scheme by using $\mathsf{pk_{ID}}$ and $\mathsf{uk_{pa(ID),t}}$. When encrypting a message, the sender runs $\mathsf{h.Enc}(\mathsf{h.mpk}, (\mathsf{ID}, \mathsf{t}), M) \to \mathsf{h.ct_{ID,t}}$ and $\mathsf{r.Enc}(\mathsf{r.mpk}, \mathsf{ID}, \mathsf{t}, \mathsf{h.ct_{ID,t}}) \to \mathsf{r.ct_{ID,t}}$, and sends $\mathsf{r.ct_{ID,t}}$ to the server. The latter uses $\mathsf{tk_{ID,t}}$ to transform this ciphertext into a "partial decrypted ciphertext" – a HIBE ciphertext $\mathsf{h.ct'_{ID,t}}$, and sends it to the intended recipient. The recipient uses $\mathsf{h.sk_{ID}}$ to generate his decryption key $\mathsf{dk_{ID,t}} := \mathsf{h.sk_{ID,t}}$, and use it to recover the message.

Related works. Revocation mechanism has been considered in many multi-user cryptosystems [7, 13, 17, 22, 24, 39, 40] to revoke the malicious user. While most schemes suffer one problem: the workloads of the user is too heavy, so server-aided revocation mechanism was also introduced to such schemes. For example, server-aided revocation mechanism was studied in the setting of attribute-based encryption by Cui et al. [8] and they also considered the decryption key exposure on the transform keys. Later on, Qin et al. [29] considered server-aided RABE scheme against local decryption key exposure attacks. In 2018, Ling et al. [21] introduced the server-aided revocation mechanism to predicate encryption.

Roadmap. The rest of this paper is organized as follows. Section 2 provides definitions of HIBE, RHIBE and SR-HIBE schemes, and the strategy-dividing Lemma which will be used in the security proof. Our generic construction of SR-HIBE scheme and its security proof are presented in Section 3. We summarize our results in Section 4.

2 Preliminaries

Notations. Let λ be the security parameter, $\mathsf{negl}(\lambda)$ represents a negligible function. For positive integer $n \in \mathbb{N}$, $[n]$ represents the set $\{1, \cdots, n\}$. PPT is the abbreviation for probabilistic polynomial time. In (R)HIBE, $\mathsf{ID} = (\mathsf{id}_1, \cdots, \mathsf{id}_\ell)$, $\mathsf{id}_i \in \mathcal{ID}$, denotes a ℓ-level user with identity ID, where id_i and $\mathcal{ID}$ are called as element identity and element identity space. For $\ell \in \mathbb{N}$, define $(\mathcal{ID})^{\leq \ell} := \bigcup_{i \in [\ell]} (\mathcal{ID})^i$ and the hierarchical identity space $\mathcal{ID}_h :=$

$(\mathcal{ID})^{\leq L}$, where L is the maximum depth of the hierarchy. The level-0 user is denoted as kgc, i.e., the key generation center. For a ℓ-level identity $\mathsf{ID} = (\mathsf{id}_1, \cdots, \mathsf{id}_\ell)$, set $\mathsf{ID}_{[i]} := (\mathsf{id}_1, \cdots, \mathsf{id}_i)$ represents the length-i prefix of ID, $i \in [\ell]$, define $\mathsf{pa}(\mathsf{ID}) := (\mathsf{id}_1, \cdots, \mathsf{id}_{\ell-1})$ as the direct ancestor of ID and $\mathsf{prefix}(\mathsf{ID}) := \{\mathsf{ID}_{[1]}, \cdots, \mathsf{ID}_{[\ell]} = \mathsf{ID}\}$. Furthermore, $\mathsf{ID}\|\mathcal{ID} \subseteq (\mathcal{ID})^{\ell+1}$ represents the subset that contains all the nodes who have ID as its direct ancestor.

2.1 Hierarchical Identity-Based Encryption

In this paper, we extend the syntax definition of two-level HIBE scheme in [15] to the general case. Assume that the plaintext space $\mathcal{M}$, the element identity space $\mathcal{ID}$, and the hierarchical identity space $\mathcal{ID}_h := (\mathcal{ID})^{\leq L}$. The syntax definition of HIBE scheme $\mathcal{HIBE} = (\mathsf{Setup}, \mathsf{Delegate}, \mathsf{Enc}, \mathsf{Dec})$ is defined as follows.

Definition 1 (Hierarchical Identity-Based Encryption, HIBE).

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk}_{\mathsf{kgc}})$. *On input the security parameter* 1^λ*, output a public parameter* mpk *and the master secret key* $\mathsf{msk}_{\mathsf{kgc}}$.
- $\mathsf{Delegate}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{pa(ID)}}, \mathsf{ID}) \to \mathsf{sk}_{\mathsf{ID}}$[1]. *This algorithm is run by the user* $\mathsf{pa}(\mathsf{ID})$ *where* $\mathsf{ID} \in \mathcal{ID}_h$*. It takes the public parameter* mpk*, a secret key* $\mathsf{sk}_{\mathsf{pa(ID)}}$ *and an identity* ID*, outputs the secret key* $\mathsf{sk}_{\mathsf{ID}}$.
- $\mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}, M) \to \mathsf{ct}$. *On input the public parameter* mpk*, the recipient's identity* $\mathsf{ID} \in \mathcal{ID}_h$ *and a message* M*, outputs a ciphertext* ct.
- $\mathsf{Dec}(\mathsf{mpk}, \mathsf{ct}, \mathsf{sk}_{\mathsf{ID}}) \to M/\perp$. *It takes the public parameter* mpk*, a ciphertext* ct *and a secret key* $\mathsf{sk}_{\mathsf{ID}}$ *as inputs, outputs a message* M *or a symbol* $\perp$.

Correctness. For all λ, $(\mathsf{mpk}, \mathsf{msk}_{\mathsf{kgc}}) \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{ID} \in \mathcal{ID}_h$, $\mathsf{sk}_{\mathsf{ID}} \leftarrow \mathsf{Delegate}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{pa(ID)}}, \mathsf{ID})$, $M \in \mathcal{M}$, $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}, M)$, it holds that $\mathsf{Dec}(\mathsf{mpk}, \mathsf{ct}, \mathsf{sk}_{\mathsf{ID}}) = M$.

Security definition. We also extend the security definition in [15]. The following game is played between an adversary $\mathcal{A}$ and the challenger $\mathcal{C}$.

Initial. $\mathcal{A}$ announces the challenge identity $\mathsf{ID}^* \in \mathcal{ID}_h$ and sends it to $\mathcal{C}$.

Setup. $\mathcal{C}$ runs $(\mathsf{mpk}, \mathsf{msk}_{\mathsf{kgc}}) \leftarrow \mathsf{Setup}(1^\lambda)$, and prepares a table T which initially contains $(\mathsf{kgc}, \mathsf{msk}_{\mathsf{kgc}})$ to store the generated identity/secret key pair $(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}})$ during the game. Then $\mathcal{C}$ sends mpk to $\mathcal{A}$.

Level-i secret key generation query: When $\mathcal{A}$ issues a query $\mathsf{ID} \in (\mathcal{ID})^i$, $i \in [L]$, $\mathcal{C}$ first checks if $(\mathsf{ID}, *) \notin \mathrm{T}$ and $(\mathsf{pa}(\mathsf{ID}), \mathsf{sk}_{\mathsf{pa(ID)}}) \in \mathrm{T}$ for some $\mathsf{pa}(\mathsf{ID})$. If not, return $\perp$. Otherwise, $\mathcal{C}$ runs $\mathsf{sk}_{\mathsf{ID}} \leftarrow \mathsf{Delegate}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{pa(ID)}}, \mathsf{ID})$ and stores $(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}})$ into the table T but returns nothing to $\mathcal{A}$.

[1] We combine the GenSK and $\mathsf{Delegate}$ algorithms in [15] into one $\mathsf{Delegate}$ algorithm. When $\mathsf{pa}(\mathsf{ID}) = \mathsf{kgc}$, $\mathsf{sk}_{\mathsf{pa(ID)}} := \mathsf{msk}_{\mathsf{kgc}}$.

Level-i secret key reveal query: When $\mathcal{A}$ queries on $\mathsf{ID} \in (\mathcal{ID})^i$, $i \in [L]$, $\mathcal{C}$ first checks if $\mathsf{ID} \notin \mathsf{prefix}(\mathsf{ID}^*)$ and $(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}}) \in \mathrm{T}$. If not, return $\perp$. Otherwise, $\mathcal{C}$ returns $\mathsf{sk}_{\mathsf{ID}}$ to $\mathcal{A}$.
Level-$(i, i+1)$ secret key reveal query: When $\mathcal{A}$ queries on $(\mathsf{ID}, \mathsf{id}_{i+1}) \in \mathcal{ID}^{i+1}$, $\mathcal{C}$ first checks if $\mathsf{ID} \notin \mathsf{prefix}(\mathsf{ID}^*)$ and $(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}}) \in \mathrm{T}$. If not, return $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ executes $\mathsf{sk}_{\mathsf{ID}||\mathsf{id}_{i+1}} \leftarrow \mathsf{Delegate}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{id}_{i+1})$ and returns $\mathsf{sk}_{\mathsf{ID}||\mathsf{id}_{i+1}}$ to $\mathcal{A}$.
Challenge phase: Once $\mathcal{A}$ submits two messages M_0, M_1 with equal length, $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$, and sends the challenge ciphertext $\mathsf{ct}_b^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}^*, M_b)$ to $\mathcal{A}$.
Guess. $\mathcal{A}$ make a guess b' for b and wins the game if $b' = b$.

Definition 2 (Selective-identity security). *A HIBE scheme $\mathcal{HIBE}$ is selective-identity secure if for any PPT adversary $\mathcal{A}$, its advantage denoted as $\mathbf{Adv}^{\mathsf{HIBE-sel}}_{\mathcal{HIBE},\mathcal{A}}(1^\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$ is negligible in λ.*

Adaptive-identity security. A HIBE scheme $\mathcal{HIBE}$ is called adaptive-identity secure if $\mathcal{A}$ announces ID^* and two equal length messages M_0, M_1 at the **Challenge phase** with the restrictions that $\mathcal{A}$ has never issued the secret key generation query on ID^* and the secret key reveal query on any of $\mathsf{ID} \in \mathsf{prefix}(\mathsf{ID}^*)$.

2.2 Revocable Hierarchical Identity-Based Encryption

In this paper, we make a little change to the syntax definition of the RHIBE scheme in [15], in order that we can obtain our generic construction of SR-HIBE scheme smoothly. As mentioned in technique overview in introduction section, we separate the secret key $\mathsf{sk}_{\mathsf{ID}}$ into two parts: $(\mathsf{sk}_{\mathsf{ID}}, \mathsf{msk}_{\mathsf{ID}})$. Note that this modification does not change the syntax definition of RHIBE scheme since it is just a change in form. Assume that the plaintext space $\mathcal{M}$, the time period space $\mathcal{T}$, the element identity space $\mathcal{ID}$, and the hierarchical identity space $\mathcal{ID}_h := (\mathcal{ID})^{\leq L}$, the syntax definition of the RHIBE scheme $\mathcal{RHIBE} = (\mathsf{Setup}, \mathsf{GenSK}, \mathsf{KeyUp}, \mathsf{GenDK}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rev})$ is as follows.

Definition 3 (Revocable Hierarchical Identity-Based Encryption, RHIBE [15]).

- $\mathsf{Setup}(1^\lambda, L) \rightarrow (\mathsf{mpk}, \mathsf{msk}_{\mathsf{kgc}}, \mathsf{RL}_{\mathsf{kgc}}, \mathsf{st}_{\mathsf{kgc}})$. *This algorithm is run by the KGC. On input the security parameter 1^λ and the maximum depth of the hierarchy $L \in \mathbb{N}$, output a public parameter mpk, the KGC's secret key $\mathsf{msk}_{\mathsf{kgc}}$ (also called the master secret key), the initial state st_{kgc} and an empty revocation list $\mathsf{RL}_{\mathsf{kgc}}$.*
- $\mathsf{GenSK}(\mathsf{mpk}, \mathsf{msk}_{\mathsf{pa(ID)}}, \mathsf{ID}, \mathsf{st}_{\mathsf{pa(ID)}}) \rightarrow (\mathsf{sk}_{\mathsf{ID}}, \mathsf{msk}_{\mathsf{ID}}, \mathsf{RL}_{\mathsf{ID}}, \mathsf{st}_{\mathsf{ID}}, \mathsf{st}'_{\mathsf{pa(ID)}})$.[2] *This algorithm is run by a parent user $\mathsf{pa}(\mathsf{ID})$. On input the public parameter mpk, the parent's master secret key $\mathsf{msk}_{\mathsf{pa(ID)}}$, an identity $\mathsf{ID} \in \mathcal{ID}_h$ and the state $\mathsf{st}_{\mathsf{pa(ID)}}$, output a secret*

[2] When ID is in level-1, then $\mathsf{pa}(\mathsf{ID})$ is the KGC, i.e., $\mathsf{sk}_{\mathsf{pa(ID)}} = \mathsf{sk}_{\mathsf{kgc}}$. When $L = 1$, i.e., the RIBE case, $\mathsf{GenSK}(\mathsf{mpk}, \mathsf{msk}_{\mathsf{kgc}}, \mathsf{ID}, \mathsf{st}_{\mathsf{kgc}}) \rightarrow (\mathsf{sk}_{\mathsf{ID}}, \perp, \perp, \perp, \mathsf{st}'_{\mathsf{kgc}})$.

key $\mathsf{sk}_{\mathsf{ID}}$*, the delegated master secret key* $\mathsf{msk}_{\mathsf{ID}}$ *which was used when* ID *worked as a "delegate KGC" for its children, an empty revocation list* $\mathsf{RL}_{\mathsf{ID}}$ *and the state* $\mathsf{st}_{\mathsf{ID}}$ *that held by* ID*, and the updated state* $\mathsf{st}'_{\mathsf{pa(ID)}}$.

- $\mathsf{KeyUp}(\mathsf{mpk}, \mathsf{msk}_{\mathsf{ID}}, \mathsf{t}, \mathsf{RL}_{\mathsf{ID,t}}, \mathsf{ku}_{\mathsf{pa(ID),t}}, \mathsf{st}_{\mathsf{ID}}) \rightarrow (\mathsf{ku}_{\mathsf{ID,t}}, \mathsf{st}'_{\mathsf{ID}})$. *This algorithm is run by the user* $\mathsf{ID} \in \mathcal{ID}_h$*. It takes the public parameter* mpk*, a master secret key* $\mathsf{msk}_{\mathsf{ID}}$*, the time period* t*, the revocation list* $\mathsf{RL}_{\mathsf{ID,t}} \subseteq \mathsf{ID} \parallel \mathcal{ID}$*, a parent's key update* $\mathsf{ku}_{\mathsf{pa(ID),t}}$ *and the state* $\mathsf{st}_{\mathsf{ID}}$ *as inputs, outputs the key update information* $\mathsf{ku}_{\mathsf{ID,t}}$ *and the updated state* $\mathsf{st}'_{\mathsf{ID}}$.
- $\mathsf{GenDK}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ku}_{\mathsf{pa(ID),t}}) \rightarrow \mathsf{dk}_{\mathsf{ID,t}}$. *This algorithm is run by the receiver* $\mathsf{ID} \in \mathcal{ID}_h$*. On input the public parameter* mpk*, a secret key* $\mathsf{sk}_{\mathsf{ID}}$ *and a parent's key update information* $\mathsf{ku}_{\mathsf{pa(ID),t}}$*, output a short-term decryption key* $\mathsf{dk}_{\mathsf{ID,t}}$ *for time period* t.
- $\mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}, \mathsf{t}, M) \rightarrow \mathsf{ct}_{\mathsf{ID,t}}$. *This algorithm is run by the sender. It takes the public parameter* mpk*, the recipient's identity* $\mathsf{ID} \in \mathcal{ID}_h$*, a time period* t *and a message* M*, outputs a ciphertext* $\mathsf{ct}_{\mathsf{ID,t}}$.
- $\mathsf{Dec}(\mathsf{mpk}, \mathsf{ct}_{\mathsf{ID,t}}, \mathsf{dk}_{\mathsf{ID,t}}) \rightarrow M/\perp$. *This algorithm is run by the receiver* $\mathsf{ID} \in (\mathcal{ID})^{\leq L}$*. It takes the public parameter* mpk*, a ciphertext* $\mathsf{ct}_{\mathsf{ID,t}}$ *and a short-term decryption key* $\mathsf{dk}_{\mathsf{ID,t}}$*, outputs a message* M *or a symbol* $\perp$.
- $\mathsf{Rev}((\mathsf{ID}, \mathsf{t}), \mathsf{RL}_{\mathsf{pa(ID),t}}, \mathsf{st}_{\mathsf{pa(ID)}}) \rightarrow \mathsf{RL}_{\mathsf{pa(ID),t}}$. *This algorithm is run by a parent user with* $\mathsf{pa(ID)}$*. On input the identity* $\mathsf{ID} \in \mathcal{ID}_h$ *and the time* t*, the revocation list* $\mathsf{RL}_{\mathsf{pa(ID),t}}$ *managed by* $\mathsf{pa(ID)}$ *and the state* $\mathsf{st}_{\mathsf{pa(ID)}}$*, output the updated revocation list* $\mathsf{RL}_{\mathsf{pa(ID),t}}$ *by adding* ID *as a revoked user at time* t.

Correctness. It required that for all $\lambda \in \mathbb{N}$, $L \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk}_{\mathsf{kgc}}, \mathsf{RL}_{\mathsf{kgc}}, \mathsf{st}_{\mathsf{kgc}}) \leftarrow \mathsf{Setup}(1^\lambda, L)$, $\ell \in [L]$, $\mathsf{ID} \in \mathcal{ID}_h$, $\mathsf{t} \in \mathcal{T}$, $\mathsf{M} \in \mathcal{M}$, $\mathsf{RL}_{\mathsf{ID}_{[\ell-1]},\mathsf{t}} \subseteq \mathsf{ID}_{[\ell-1]} \parallel \mathcal{ID}$, if $\mathsf{ID}' \notin \mathsf{RL}_{\mathsf{pa(ID')},\mathsf{t}}$ holds for all $\mathsf{ID}' \in \mathsf{prefix(ID)}$, then we require $M' = M$ to hold after does the following steps:

(1) $(\mathsf{ku}_{\mathsf{kgc,t}}, \mathsf{st}_{\mathsf{kgc}}) \leftarrow \mathsf{KeyUp}(\mathsf{mpk}, \mathsf{msk}_{\mathsf{kgc}}, \mathsf{t}, \mathsf{RL}_{\mathsf{kgc,t}}, \perp, \mathsf{st}_{\mathsf{kgc}})$.
(2) For all $\mathsf{ID}' \in \mathsf{prefix(ID)}$ (in the short to long order), execute (2.1) and (2.2):
  (2.1) $(\mathsf{sk}_{\mathsf{ID}'}, \mathsf{msk}_{\mathsf{ID}'}, \mathsf{RL}_{\mathsf{ID}'}, \mathsf{st}_{\mathsf{ID}'}, \mathsf{st}'_{\mathsf{pa(ID')}}) \leftarrow \mathsf{GenSK}(\mathsf{mpk}, \mathsf{msk}_{\mathsf{pa(ID')}}, \mathsf{ID}', \mathsf{st}_{\mathsf{pa(ID')}})$
  (2.2) $(\mathsf{ku}_{\mathsf{ID}',\mathsf{t}}, \mathsf{st}'_{\mathsf{ID}'}) \leftarrow \mathsf{KeyUp}(\mathsf{mpk}, \mathsf{msk}_{\mathsf{ID}'}, \mathsf{t}, \mathsf{RL}_{\mathsf{ID}',\mathsf{t}}, \mathsf{ku}_{\mathsf{pa(ID')},\mathsf{t}}, \mathsf{st}_{\mathsf{ID}'})$[3].
(3) $\mathsf{dk}_{\mathsf{ID,t}} \leftarrow \mathsf{GenDK}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ku}_{\mathsf{pa(ID),t}})$
(4) $\mathsf{ct}_{\mathsf{ID,t}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}, \mathsf{t}, M)$.
(5) $M' \leftarrow \mathsf{Dec}(\mathsf{mpk}, \mathsf{ct}_{\mathsf{ID,t}}, \mathsf{dk}_{\mathsf{ID,t}})$.

Security definition. We adopt the stateful version of the security definition for RHIBE in [15]. We also introduce the "current time period" $\mathsf{t}_{\mathsf{cu}} \in \mathcal{T}$ and initialize it to 1 as in [15]. The following game is played between an adversary $\mathcal{A}$ and the challenger $\mathcal{C}$.

[3] If $|\mathsf{ID}'| = L$, then this step is skipped.

Initial. $\mathcal{A}$ announces the challenge identity/time period pair $(\mathsf{ID}^*, \mathsf{t}^*) \in \mathcal{ID}_h \times \mathcal{T}$ and sends them to $\mathcal{C}$.

Setup. $\mathcal{C}$ first runs $(\mathsf{mpk}, \mathsf{msk}_{\mathsf{kgc}}, \mathsf{RL}_{\mathsf{kgc}}, \mathsf{st}_{\mathsf{kgc}}) \leftarrow \mathsf{Setup}(1^\lambda, L)$, and prepares a table T to store the generated identity/master secret key/secret key tuple $(\mathsf{ID}, \mathsf{msk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}})$ during the game. The table is initialized to $(\mathsf{kgc}, \mathsf{msk}_{\mathsf{kgc}}, \perp)$. Then $\mathcal{C}$ executes $(\mathsf{ku}_{\mathsf{kgc},1}, \mathsf{st}_{\mathsf{kgc}}) \leftarrow \mathsf{KeyUp}(\mathsf{mpk}, \mathsf{msk}_{\mathsf{kgc}}, \mathsf{t}_{\mathsf{cu}} = 1, \mathsf{RL}_{\mathsf{kgc},1} = \emptyset, \perp, \mathsf{st}_{\mathsf{kgc}})$, and sends the public parameter mpk and the key update $\mathsf{ku}_{\mathsf{kgc},1}$ to $\mathcal{A}$.

$\mathcal{A}$ may adaptively issue any of the following queries to $\mathcal{C}$:

Secret key generation query: When $\mathcal{A}$ issues a query $\mathsf{ID} \in \mathcal{ID}_h$, $\mathcal{C}$ first checks if $(\mathsf{ID}, *, *) \notin \mathrm{T}$ and $(\mathsf{pa}(\mathsf{ID}), \mathsf{msk}_{\mathsf{pa}(\mathsf{ID})}, \mathsf{sk}_{\mathsf{pa}(\mathsf{ID})}) \in \mathrm{T}$ for some $\mathsf{pa}(\mathsf{ID})$. If not, return $\perp$. Otherwise, $\mathcal{C}$ runs $(\mathsf{sk}_{\mathsf{ID}}, \mathsf{msk}_{\mathsf{ID}}, \mathsf{RL}_{\mathsf{ID}}, \mathsf{st}_{\mathsf{ID}}, \mathsf{st}'_{\mathsf{pa}(\mathsf{ID})}) \leftarrow \mathsf{GenSK}(\mathsf{mpk}, \mathsf{msk}_{\mathsf{pa}(\mathsf{ID})}, \mathsf{ID}, \mathsf{st}_{\mathsf{pa}(\mathsf{ID})})$, and stores $(\mathsf{ID}, \mathsf{msk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}})$ into the table T. If $\mathsf{ID} \in (\mathcal{ID})^{\leq L-1}$, $\mathcal{C}$ also need to execute $(\mathsf{ku}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}}, \mathsf{st}'_{\mathsf{ID}}) \leftarrow \mathsf{KeyUp}(\mathsf{mpk}, \mathsf{msk}_{\mathsf{ID}}, \mathsf{t}_{\mathsf{cu}}, \mathsf{RL}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}} = \emptyset, \mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}_{\mathsf{cu}}}, \mathsf{st}_{\mathsf{ID}})$. Then, $\mathcal{C}$ returns the key update information $\mathsf{ku}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}}$ to $\mathcal{A}$ when $\mathsf{ID} \in (\mathcal{ID})^{\leq L-1}$, or return nothing to $\mathcal{A}$ if $\mathsf{ID} \in (\mathcal{ID})^L$.

In the following, we require all identities ID in the following queries must have been queried in this query, namely, $(\mathsf{ID}, \mathsf{msk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}}) \in \mathrm{T}$.

Secret key reveal query: When $\mathcal{A}$ queries on $\mathsf{ID} \in \mathcal{ID}_h$, $\mathcal{C}$ first checks whether the condition: if $\mathsf{t}_{\mathsf{cu}} \geq \mathsf{t}^*$ and $\mathsf{ID} \in \mathsf{predix}(\mathsf{ID}^*)$, then $\mathsf{ID} \in \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}^*}$, is satisfied. If not, return $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ obtains the entry $(\mathsf{ID}, \mathsf{msk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}})$ and returns the secret key $(\mathsf{msk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}})$ to $\mathcal{A}$.

Revoke & key update query: When $\mathcal{A}$ queries on $\mathsf{RL} \subseteq \mathcal{ID}_h$ (which denotes the set of identities that will be revoked), $\mathcal{C}$ first checks the following conditions:

- $\mathsf{RL}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}} \subseteq \mathsf{RL}$ for all $\mathsf{ID} \in (\mathcal{ID})^{\leq L-1} \cup \{\mathsf{kgc}\}$ that appear in table T.[4]
- For all identities ID such that $(\mathsf{ID}, *, *) \in \mathrm{T}$ and $\mathsf{ID}' \in \mathsf{prefix}(\mathsf{ID})$, if $\mathsf{ID}' \in \mathsf{RL}$, then $\mathsf{ID} \in \mathsf{RL}$.[5]
- If $\mathsf{t}_{\mathsf{cu}} = \mathsf{t}^* - 1$ and $\mathcal{A}$ has already issued the secret key reveal queries on some $\mathsf{ID}' \in \mathsf{prefix}(\mathsf{ID}^*)$, then $\mathsf{ID}' \in \mathsf{RL}$.

If those are all not satisfied, return $\perp$. Otherwise, $\mathcal{C}$ increments $\mathsf{t}_{\mathsf{cu}} \leftarrow \mathsf{t}_{\mathsf{cu}} + 1$. Then, $\mathcal{C}$ executes the following two steps for all identities that have been issued a secret key generation queries and not been revoked, i.e., $\mathsf{ID} \in (\mathcal{ID})^{\leq L-1} \cup \{\mathsf{kgc}\}$, $(\mathsf{ID}, *, *) \in \mathrm{T}$ and $\mathsf{ID} \notin \mathsf{RL}$, in the identity hierarchy order:

1. Set $\mathsf{RL}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}} \leftarrow \mathsf{RL} \cap (\mathsf{ID} \| \mathcal{ID})$, where $\mathsf{kgc} \| \mathcal{ID} := \mathcal{ID}$.

[4] This check ensures that the identities that have already been revoked will remain revoked in the next time period.

[5] This check ensures that if some ID is revoked, then all of its descendants are also revoked.

2. Run $(\mathsf{ku}_{\mathsf{ID},\mathsf{t_{cu}}}, \mathsf{st}'_{\mathsf{ID}}) \leftarrow \mathsf{KeyUp}(\mathsf{mpk}, \mathsf{msk}_{\mathsf{ID}}, \mathsf{t_{cu}}, \mathsf{RL}_{\mathsf{ID},\mathsf{t_{cu}}}, \mathsf{ku}_{\mathsf{pa(ID)},\mathsf{t_{cu}}}, \mathsf{st}_{\mathsf{ID}})$. When $\mathsf{ID} = \mathsf{kgc}$, $\mathsf{ku}_{\mathsf{pa(kgc)},\mathsf{t_{cu}}} := \perp$.

Finally, $\mathcal{C}$ sends all the key update information $\{\mathsf{ku}_{\mathsf{ID},\mathsf{t_{cu}}}\}_{(\mathsf{ID},*,*)\in \mathrm{T}}$ to $\mathcal{A}$.

Decryption key reveal query: When $\mathcal{A}$ issues a query $(\mathsf{ID}, \mathsf{t}) \in \mathcal{ID}_h \times \mathcal{T}$, $\mathcal{C}$ first checks if the following conditions holds:

– $\mathsf{t} \leq \mathsf{t_{cu}}$, $\mathsf{ID} \notin \mathsf{RL}_{\mathsf{pa(ID)},\mathsf{t}}$, $(\mathsf{ID}, \mathsf{t}) \neq (\mathsf{ID}^*, \mathsf{t}^*)$.

If not, return $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ obtains the tuple $(\mathsf{ID}, \mathsf{msk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}})$ from the table T, runs $\mathsf{dk}_{\mathsf{ID},\mathsf{t}} \rightarrow \mathsf{GenDK}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ku}_{\mathsf{pa(ID)},\mathsf{t}})$ and returns $\mathsf{dk}_{\mathsf{ID},\mathsf{t}}$ to $\mathcal{A}$.

Challenge phase: Once $\mathcal{A}$ submits two messages M_0, M_1 with equal length, $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$, and sends the challenge ciphertext $\mathsf{ct}^*_b \leftarrow \mathsf{Enc}(\mathsf{ID}^*, \mathsf{t}^*, M_b)$ to $\mathcal{A}$.

Guess. $\mathcal{A}$ make a guess b' for b and wins the game if $b' = b$.

Definition 4 (Selective-identity security). *A RHIBE scheme $\mathcal{RHIBE}$ is selective-identity secure if for any PPT adversary $\mathcal{A}$, its advantage denoted as* $\mathbf{Adv}^{\mathsf{RHIBE-sel}}_{\mathcal{RHIBE},\mathcal{A}}(1^\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$ *is negligible in λ.*

Adaptive-identity security. A RHIBE scheme $\mathcal{RHIBE}$ is called adaptive-identity secure if $\mathcal{A}$ announces the challenge identity/time period pair $(\mathsf{ID}^*, \mathsf{t}^*)$ and two equal length messages M_0, M_1 at the **Challenge phase** with the following restrictions that

– If $\mathsf{t}^* \leq \mathsf{t_{cu}}$, then $\mathcal{A}$ has not submitted a decryption key reveal query on $(\mathsf{ID}^*, \mathsf{t}^*)$,
– If $\mathcal{A}$ has been issued a secret key reveal query for any $\mathsf{ID} \in \mathsf{prefix}(\mathsf{ID}^*)$, then $\mathsf{ID} \in \mathsf{RL}_{\mathsf{pa(ID)},\mathsf{t}^*}$ and thus, $\mathsf{ID}^* \in \mathsf{RL}_{\mathsf{pa(ID)},\mathsf{t}^*}$.

Note that those restrictions related to ID^*, t^* are only work in the challenge phase, while it is not considered before the challenge query.

Weak selective/adaptive-identity secure. A RHIBE scheme $\mathcal{RHIBE}$ is called weak security (i.e., security without DKER) if $\mathcal{A}$ is not allowed to make any decryption key reveal query. The *weak selective-identity* (resp. *weak adaptive-identity*) security advantage of $\mathcal{A}$ is denoted by $\mathbf{Adv}^{\mathsf{RHIBE-sel-weak}}_{\mathcal{RHIBE},\mathcal{A}}(1^\lambda)$ (resp. $\mathbf{Adv}^{\mathsf{RHIBE-adap-weak}}_{\mathcal{RHIBE},\mathcal{A}}(1^\lambda)$).

2.3 Server-Aided Revocable Hierarchical Identity-Based Encryption

In this section, we give a definition of the SR-HIBE scheme. Qin et al. [28] proposed and formalized the definition of SR-IBE scheme which contains ten algorithms: $\mathsf{Setup}, \mathsf{PubKG}, \mathsf{KeyUp}, \mathsf{TranKG}, \mathsf{PrivKG}, \mathsf{DecKG}, \mathsf{Enc}, \mathsf{Transform}, \mathsf{Dec}$ and Rev. Note that in their definition, $\mathsf{PubKG}(\mathsf{msk}_{\mathsf{kgc}}, \mathsf{ID}, \mathsf{st}) \rightarrow (\mathsf{pk}_{\mathsf{ID}}, \mathsf{st}')$ and $\mathsf{PrivKG}(\mathsf{msk}_{\mathsf{kgc}}, \mathsf{ID}) \rightarrow \mathsf{sk}_{\mathsf{ID}}$ are all executed by the KGC for the user ID. Thus we combine these two algorithms into one UerKG algorithm

where $\mathsf{UerKG}(\mathsf{msk_{kgc}}, \mathsf{ID}, \mathsf{st}) \to (\mathsf{pk_{ID}}, \mathsf{sk_{ID}}, \mathsf{st}')$, namely, when a user join the system, the KGC generates a public/secret key pair for the user. Note that this is just a change in form, the combined definition of SR-IBE is equivalent to that in [28]. We extend this combined definition of SR-IBE to the HIBE setting to obtain the definition of SR-HIBE. Assume that the plaintext space $\mathcal{M}$, the time period space $\mathcal{T}$, the element identity space $\mathcal{ID}$, and the hierarchical identity space $\mathcal{ID}_{\mathsf{h}} := (\mathcal{ID})^{\leq L}$. The definition of SR-HIBE scheme $\mathcal{SR}\text{-}\mathcal{HIBE} = (\mathsf{Setup}, \mathsf{UserKG}, \mathsf{UpdKG}, \mathsf{TranKG}, \mathsf{DecKG}, \mathsf{Enc}, \mathsf{Transform}, \mathsf{Dec}, \mathsf{Rev})$ is as follows.

Definition 5 (Server-Aided Revocable Hierarchical Identity-Based Encryption, SR-HIBE).

- $\mathsf{Setup}(1^\lambda, L) \to (\mathsf{mpk}, \mathsf{msk_{kgc}}, \mathsf{st_{kgc}}, \mathsf{RL_{kgc}})$. *This algorithm is run by the KGC. On input the security parameter* 1^λ *and the maximum depth of the hierarchy* $L \in \mathbb{N}$*, output a public parameter* mpk*, the KGC's secret key* $\mathsf{msk_{kgc}}$ *(also called the master secret key), the initial state* st_{kgc} *and an empty revocation list* $\mathsf{RL_{kgc}}$.
- $\mathsf{UserKG}(\mathsf{mpk}, \mathsf{sk_{pa(ID)}}, \mathsf{ID}, \mathsf{st_{pa(ID)}}) \to (\mathsf{pk_{ID}}, \mathsf{sk_{ID}}, \mathsf{st}'_{\mathsf{pa(ID)}}, \mathsf{RL_{ID}}, \mathsf{st_{ID}})$.[6] *This algorithm is run by a parent user* $\mathsf{pa(ID)}$ *when a user* $\mathsf{ID} \in \mathcal{ID}_h$ *registers to the system. On input the public parameter* mpk*, the parent's secret key* $\mathsf{sk_{pa(ID)}}$*, an identity* ID *and the state* $\mathsf{st_{pa(ID)}}$*, output a public key* $\mathsf{pk_{ID}}$*, a secret key* $\mathsf{sk_{ID}}$*, the updated state* $\mathsf{st}'_{\mathsf{pa(ID)}}$*, an empty revocation list* $\mathsf{RL_{ID}}$ *and an initial state* $\mathsf{st_{ID}}$ *managed by* ID*. The public key* $\mathsf{pk_{ID}}$ *is sent to the server through public channel and the secret key* $\mathsf{sk_{ID}}$ *is sent to the user* ID *by secret channel.*
- $\mathsf{UpdKG}(\mathsf{mpk}, \mathsf{sk_{ID}}, \mathsf{t}, \mathsf{RL_{ID,t}}, \mathsf{uk_{pa(ID),t}}, \mathsf{st_{ID}}) \to (\mathsf{uk_{ID,t}}, \mathsf{st}'_{\mathsf{ID}})$. *This algorithm is run by the user with* $\mathsf{ID} \in \mathcal{ID}_h$*. It takes the public parameter* mpk*, a secret key* $\mathsf{sk_{ID}}$*, the time period* t*, the revocation list* $\mathsf{RL_{ID,t}} \subseteq \mathsf{ID} \parallel \mathcal{ID}$*, a parent's update key* $\mathsf{uk_{pa(ID),t}}$ *and the state* $\mathsf{st_{ID}}$ *as inputs, and outputs an update key* $\mathsf{uk_{ID,t}}$ *and the updated state* $\mathsf{st}'_{\mathsf{ID}}$*. The update key* $\mathsf{uk_{ID,t}}$ *also is sent to the server by public channel.*
- $\mathsf{TranKG}(\mathsf{mpk}, \mathsf{pk_{ID}}, \mathsf{uk_{pa(ID),t}}) \to \mathsf{tk_{ID,t}}$. *This algorithm is run by the server. It takes the public parameter* mpk*, a public key* $\mathsf{pk_{ID}}$ *of a user with* $\mathsf{ID} \in \mathcal{ID}_h$ *and a parent's update key* $\mathsf{uk_{pa(ID),t}}$ *as inputs, outputs a short-term transform key* $\mathsf{tk_{ID,t}}$ *for time period* t.
- $\mathsf{DecKG}(\mathsf{mpk}, \mathsf{sk_{ID}}, \mathsf{t}) \to \mathsf{dk_{ID,t}}$. *This algorithm is run by the recipient. On input the public parameter* mpk*, a secret key* $\mathsf{sk_{ID}}$ *of user with* $\mathsf{ID} \in \mathcal{ID}_h$ *and a time period* t*, output a short-term decryption key* $\mathsf{dk_{ID,t}}$ *for time period* t.
- $\mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}, \mathsf{t}, M) \to \mathsf{ct_{ID,t}}$. *This algorithm is run by the sender. It takes the public parameter* mpk*, the recipient's identity* $\mathsf{ID} \in \mathcal{ID}_h$*, a time period* t *and a message* M*, outputs a ciphertext* $\mathsf{ct_{ID,t}}$*, which is sent to the server.*

[6] When $L = 1$, i.e. for the RIBE case, $\mathsf{sk_{pa(ID)}} = \mathsf{msk_{kgc}}$, thus $\mathsf{UserKG}(\mathsf{mpk}, \mathsf{msk_{kgc}}, \mathsf{ID}, \mathsf{st_{kgc}}) \to (\mathsf{pk_{ID}}, \mathsf{sk_{ID}}, \mathsf{st}'_{\mathsf{kgc}}, \perp, \perp)$.

- $\mathsf{Transform}(\mathsf{mpk}, \mathsf{ct}_{\mathsf{ID},\mathsf{t}}, \mathsf{tk}_{\mathsf{ID},\mathsf{t}}) \rightarrow \mathsf{ct}'_{\mathsf{ID},\mathsf{t}}$. *This algorithm is run by the server. It takes the public parameter* mpk*, a ciphertext* $\mathsf{ct}_{\mathsf{ID},\mathsf{t}}$ *and a transformation key* $\mathsf{tk}_{\mathsf{ID},\mathsf{t}}$*, outputs a partially decrypted ciphertext* $\mathsf{ct}'_{\mathsf{ID},\mathsf{t}}$*, which is sent to the recipient by the public channel.*
- $\mathsf{Dec}(\mathsf{mpk}, \mathsf{ct}'_{\mathsf{ID},\mathsf{t}}, \mathsf{dk}_{\mathsf{ID},\mathsf{t}}) \rightarrow M/\perp$. *This algorithm is run by the recipient* $\mathsf{ID} \in \mathcal{ID}_h$*. It takes the public parameter* mpk*, a partially decrypted ciphertext* $\mathsf{ct}'_{\mathsf{ID},\mathsf{t}}$ *and a decryption key* $\mathsf{dk}_{\mathsf{ID},\mathsf{t}}$*, outputs a message* M *or a symbol* $\perp$*.*
- $\mathsf{Rev}((\mathsf{ID}, \mathsf{t}), \mathsf{RL}_{\mathsf{pa(ID)},\mathsf{t}}, \mathsf{st}_{\mathsf{pa(ID)}}) \rightarrow \mathsf{RL}'_{\mathsf{pa(ID)},\mathsf{t}}$. *This algorithm is run by a parent user with* $\mathsf{pa(ID)}$ *where* $\mathsf{ID} \in \mathcal{ID}_h$*. On input the identity* ID *and the time* t*, the revocation list* $\mathsf{RL}_{\mathsf{pa(ID)},\mathsf{t}}$ *managed by* $\mathsf{pa(ID)}$ *and the state* $\mathsf{st}_{\mathsf{pa(ID)}}$*, output the updated revocation list* $\mathsf{RL}'_{\mathsf{pa(ID)},\mathsf{t}}$ *by adding* ID *as a revoked user at time* t*.*

Correctness. It needs that for all $\lambda \in \mathbb{N}$, $L \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk}_{\mathsf{kgc}}, \mathsf{st}_{\mathsf{kgc}}, \mathsf{RL}_{\mathsf{kgc}}) \leftarrow \mathsf{Setup}(1^\lambda, L)$, $\ell \in [L]$, $\mathsf{ID} \in \mathcal{ID}_h$, $\mathsf{t} \in \mathcal{T}$, $\mathsf{M} \in \mathcal{M}$, $\mathsf{RL}_{\mathsf{ID}_{[\ell-1]},\mathsf{t}} \subseteq \mathsf{ID}_{[\ell-1]} \parallel \mathcal{ID}$, if $\mathsf{ID}' \notin \mathsf{RL}_{\mathsf{pa(ID')},\mathsf{t}}$ holds for all $\mathsf{ID}' \in \mathsf{prefix(ID)}$, then $M' = M$ holds after executing the following steps:

(1) $(\mathsf{uk}_{\mathsf{kgc},\mathsf{t}}, \mathsf{st}_{\mathsf{kgc}}) \leftarrow \mathsf{UpdKG}(\mathsf{mpk}, \mathsf{msk}_{\mathsf{kgc}}, \mathsf{t}, \mathsf{RL}_{\mathsf{kgc},\mathsf{t}}, \perp, \mathsf{st}_{\mathsf{kgc}})$.
(2) For all $\mathsf{ID}' \in \mathsf{prefix(ID)}$ (in the short to long order), execute (2.1) and (2.2):
  (2.1) $(\mathsf{pk}_{\mathsf{ID}'}, \mathsf{sk}_{\mathsf{ID}'}, \mathsf{st}'_{\mathsf{pa(ID')}}, \mathsf{RL}_{\mathsf{ID}'}, \mathsf{st}_{\mathsf{ID}'}) \leftarrow \mathsf{UserKG}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{pa(ID')}}, \mathsf{ID}', \mathsf{st}_{\mathsf{pa(ID')}})$
  (2.2) $(\mathsf{uk}_{\mathsf{ID}',\mathsf{t}}, \mathsf{st}'_{\mathsf{ID}'}) \leftarrow \mathsf{UpdKG}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}'}, \mathsf{t}, \mathsf{RL}_{\mathsf{ID}',\mathsf{t}}, \mathsf{uk}_{\mathsf{pa(ID')},\mathsf{t}}, \mathsf{st}_{\mathsf{ID}'})$[7].
(3) $\mathsf{tk}_{\mathsf{ID},\mathsf{t}} \leftarrow \mathsf{TranKG}(\mathsf{mpk}, \mathsf{pk}_{\mathsf{ID}}, \mathsf{uk}_{\mathsf{pa(ID)}})$
(4) $\mathsf{ct}_{\mathsf{ID},\mathsf{t}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}, \mathsf{t}, M)$.
(5) $\mathsf{ct}'_{\mathsf{ID},\mathsf{t}} \leftarrow \mathsf{Transform}(\mathsf{mpk}, \mathsf{ct}_{\mathsf{ID},\mathsf{t}}, \mathsf{tk}_{\mathsf{ID},\mathsf{t}})$.
(6) $\mathsf{dk}_{\mathsf{ID},\mathsf{t}} \leftarrow \mathsf{DecKG}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{t})$.
(7) $M' \leftarrow \mathsf{Dec}(\mathsf{mpk}, \mathsf{ct}'_{\mathsf{ID},\mathsf{t}}, \mathsf{dk}_{\mathsf{ID},\mathsf{t}})$.

Security definition. Note that the above modification of the SR-HIBE definition is just a change in form and will not affect the security definition since we just change the time that the secret key generated and has no restrict on when the secret key reveal query happens. We give a rigorous security definition for SR-HIBE by modifying the stateful version of the security definition for RHIBE which described in Section 2.2. The following game is played between an adversary $\mathcal{A}$ and the challenger $\mathcal{C}$.

Initial. At the beginning, $\mathcal{A}$ announces the challenge identity/time period pair $(\mathsf{ID}^*, \mathsf{t}^*) \in \mathcal{ID}_h \times \mathcal{T}$ and sends them to $\mathcal{C}$.

Setup. $\mathcal{C}$ runs $(\mathsf{mpk}, \mathsf{msk}_{\mathsf{kgc}}, \mathsf{RL}_{\mathsf{kgc}}, \mathsf{st}_{\mathsf{kgc}}) \leftarrow \mathsf{Setup}(1^\lambda, L)$, and prepares a table T to store the generated identity/public key/secret key tuple $(\mathsf{ID}, \mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}})$ during the game. The table T is initialized to contains the tuple $(\mathsf{kgc}, \perp, \mathsf{msk}_{\mathsf{kgc}})$. Then $\mathcal{C}$ executes $(\mathsf{uk}_{\mathsf{kgc},1}, \mathsf{st}_{\mathsf{kgc}}) \leftarrow$

[7] If $|\mathsf{ID}'| = L$, then this step is skipped.

$\mathsf{UpdKG}(\mathsf{mpk}, \mathsf{msk}_{\mathsf{kgc}}, \mathsf{t}_{\mathsf{cu}} = 1, \mathsf{RL}_{\mathsf{kgc},1} = \emptyset, \perp, \mathsf{st}_{\mathsf{kgc}})$. After that, $\mathcal{C}$ sends the public parameter mpk and the update key $\mathsf{uk}_{\mathsf{kgc},1}$ to $\mathcal{A}$.

$\mathcal{A}$ may adaptively issue any of the following queries to $\mathcal{C}$:

Public key reveal query: When $\mathcal{A}$ issues a query $\mathsf{ID} \in \mathcal{ID}_h$, $\mathcal{C}$ first checks if $(\mathsf{ID}, *, *) \notin \mathrm{T}$ and $(\mathsf{pa}(\mathsf{ID}), \mathsf{pk}_{\mathsf{pa}(\mathsf{ID})}, \mathsf{sk}_{\mathsf{pa}(\mathsf{ID})}) \in \mathrm{T}$ for some $\mathsf{pa}(\mathsf{ID})$. If not, return $\perp$. Otherwise, $\mathcal{C}$ runs $(\mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{RL}_{\mathsf{ID}}, \mathsf{st}_{\mathsf{ID}}, \mathsf{st}'_{\mathsf{pa}(\mathsf{ID})}) \leftarrow \mathsf{GenSK}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{pa}(\mathsf{ID})}, \mathsf{ID}, \mathsf{st}_{\mathsf{pa}(\mathsf{ID})})$, and stores $(\mathsf{ID}, \mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}})$ into the table T. If $\mathsf{ID} \in (\mathcal{ID})^{\leq L-1}$, $\mathcal{C}$ also need to execute $(\mathsf{uk}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}}, \mathsf{st}'_{\mathsf{ID}}) \leftarrow \mathsf{UpdKG}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{t}_{\mathsf{cu}}, \mathsf{RL}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}} = \emptyset, \mathsf{uk}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}_{\mathsf{cu}}}, \mathsf{st}_{\mathsf{ID}})$. Then, $\mathcal{C}$ returns the public key $\mathsf{pk}_{\mathsf{ID}}$ and the update key $\mathsf{uk}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}}$ to $\mathcal{A}$ when $\mathsf{ID} \in (\mathcal{ID})^{\leq L-1}$, or returns the public key $\mathsf{pk}_{\mathsf{ID}}$ to $\mathcal{A}$ if $\mathsf{ID} \in (\mathcal{ID})^{L}$.

In the following, we require all identities ID appearing in the following queries must have been queried in this query, namely, $(\mathsf{ID}, \mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}}) \in \mathrm{T}$.

Secret key reveal query: When $\mathcal{A}$ queries on $\mathsf{ID} \in \mathcal{ID}_h$, $\mathcal{C}$ first checks if the following condition is satisfied:

- If $\mathsf{t}_{\mathsf{cu}} \geq \mathsf{t}^*$ and $\mathsf{ID} \in \mathsf{predix}(\mathsf{ID}^*)$, then $\mathsf{ID} \in \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}^*}$.

If not, return $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ obtains the entry $(\mathsf{ID}, \mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}})$ and returns $\mathsf{sk}_{\mathsf{ID}}$ to $\mathcal{A}$.

Revoke & update key query: When $\mathcal{A}$ queries on $\mathsf{RL} \subseteq \mathcal{ID}_h$ (which denotes the set of identities that will be revoked), $\mathcal{C}$ first checks if the following condition holds:

- $\mathsf{RL}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}} \subseteq \mathsf{RL}$ for all $\mathsf{ID} \in (\mathcal{ID})^{\leq L-1} \cup \{\mathsf{kgc}\}$ that appear in table T.
- For all identities ID such that $(\mathsf{ID}, *, *) \in \mathrm{T}$ and $\mathsf{ID}' \in \mathsf{prefix}(\mathsf{ID})$, if $\mathsf{ID}' \in \mathsf{RL}$, then $\mathsf{ID} \in \mathsf{RL}$.
- If $\mathsf{t}_{\mathsf{cu}} = \mathsf{t}^* - 1$ and $\mathcal{A}$ has already issued the secret key reveal queries on some $\mathsf{ID}' \in \mathsf{prefix}(\mathsf{ID}^*)$, then $\mathsf{ID}' \in \mathsf{RL}$.

If not, return $\perp$. Otherwise, $\mathcal{C}$ increments $\mathsf{t}_{\mathsf{cu}} \leftarrow \mathsf{t}_{\mathsf{cu}} + 1$. Then, $\mathcal{C}$ executes the following two steps for all identities that have been issued a secret key generation queries and not been revoked, i.e., $\mathsf{ID} \in (\mathcal{ID})^{\leq L-1} \cup \{\mathsf{kgc}\}$, $(\mathsf{ID}, *, *) \in \mathrm{T}$ and $\mathsf{ID} \notin \mathsf{RL}$, in the identity hierarchy order:

1. Set $\mathsf{RL}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}} \leftarrow \mathsf{RL} \cap (\mathsf{ID} \| \mathcal{ID})$, where $\mathsf{kgc} \| \mathcal{ID} := \mathcal{ID}$.
2. Run $(\mathsf{uk}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}}, \mathsf{st}'_{\mathsf{ID}}) \leftarrow \mathsf{UpdKG}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{t}_{\mathsf{cu}}, \mathsf{RL}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}}, \mathsf{uk}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}_{\mathsf{cu}}}, \mathsf{st}_{\mathsf{ID}})$. When $\mathsf{ID} = \mathsf{kgc}$, $\mathsf{uk}_{\mathsf{pa}(\mathsf{kgc}),\mathsf{t}_{\mathsf{cu}}} := \perp$.

Finally, $\mathcal{C}$ sends all the generated update key information $\{\mathsf{uk}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}}\}_{(\mathsf{ID},*,*)\in \mathrm{T}}$ to $\mathcal{A}$.

Decryption key reveal query: When $\mathcal{A}$ issues a query $(\mathsf{ID}, \mathsf{t}) \in \mathcal{ID}_h \times \mathcal{T}$, $\mathcal{C}$ first checks the following conditions:

- $\mathsf{t} \leq \mathsf{t}_{\mathsf{cu}}$, $\mathsf{ID} \notin \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}$, $(\mathsf{ID}, \mathsf{t}) \neq (\mathsf{ID}^*, \mathsf{t}^*)$.

If those are not hold, return $\perp$. Otherwise, $\mathcal{C}$ obtains the tuple $(\mathsf{ID}, \mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}})$ from the table T, runs $\mathsf{dk}_{\mathsf{ID},\mathsf{t}} \rightarrow \mathsf{DecKG}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{t})$ and returns $\mathsf{dk}_{\mathsf{ID},\mathsf{t}}$ to $\mathcal{A}$.

Challenge phase: Once the adversary $\mathcal{A}$ submits two messages M_0, M_1 with equal length, $\mathcal{C}$ chooses a random bit $b \in \{0,1\}$, and sends the challenge ciphertext $\mathsf{ct}_b^* \leftarrow \mathsf{Enc}(\mathsf{ID}^*, \mathsf{t}^*, M_b)$ to $\mathcal{A}$.

Guess. $\mathcal{A}$ make a guess b' for b and wins the game if $b' = b$.

Definition 6 (Selective-identity security). *A SR-HIBE scheme $\mathcal{SR}$-$\mathcal{HIBE}$ is selective-identity secure if for any PPT adversary $\mathcal{A}$, its advantage denoted as* $\mathbf{Adv}^{\mathsf{SR-HIBE-sel}}_{\mathcal{SR}-\mathcal{HIBE},\mathcal{A}}(1^\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$ *is negligible in λ.*

Adaptive-identity security. A SR-HIBE scheme $\mathcal{SR}$-$\mathcal{HIBE}$ is called adaptive-identity secure if $\mathcal{A}$ chooses the challenge identity/time period pair $(\mathsf{ID}^*, \mathsf{t}^*)$ and two equal length messages M_0, M_1 at the **Challenge phase** with the restrictions that

- If $\mathsf{t}^* \leq \mathsf{t}_{\mathsf{cu}}$, then $\mathcal{A}$ has not submitted a decryption key reveal query on $(\mathsf{ID}^*, \mathsf{t}^*)$,
- If $\mathcal{A}$ has been issued a secret key reveal query for any $\mathsf{ID} \in \mathsf{prefix}(\mathsf{ID}^*)$, then $\mathsf{ID} \in \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}^*}$ and thus, $\mathsf{ID}^* \in \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}^*}$.

Note that those restrictions related to ID^*, t^* are only work in the challenge phase, while it is not considered before the challenge query.

3 Generic Construction of Server-Aided Revocable HIBE with DKER

In this section, we show how to construct a SR-HIBE scheme by combing a RHIBE with L levels and a $(L+1)$-level HIBE scheme. Let $\mathsf{r}.\Pi = (\mathsf{r.Setup}, \mathsf{r.Delegate}, \mathsf{r.GenSK}, \mathsf{r.KeyUp}, \mathsf{r.GenDK}, \mathsf{r.Enc}, \mathsf{r.Dec}, \mathsf{r.Rev})$ be an RHIBE scheme with identity space $\mathsf{r}.\mathcal{ID}_h$, plaintext space $\mathsf{r}.\mathcal{M}$, ciphertext space $\mathsf{r}.\mathcal{CT}$ and time period space $\mathsf{r}.\mathcal{T}$. Let $\mathsf{h}.\Pi = (\mathsf{h.Setup}, \mathsf{h.Delegate}, \mathsf{h.Enc}, \mathsf{r.Dec})$ be a $(L+1)$-level HIBE scheme with identity space $\mathsf{h}.\mathcal{ID}_h$, plaintext space $\mathsf{h}.\mathcal{M}$ and ciphertext space $\mathsf{h}.\mathcal{CT}$. We assume $\mathsf{r}.\mathcal{ID} = \mathsf{h}.\mathcal{ID}$, $\mathsf{r}.\mathcal{T} \subseteq \mathsf{h}.\mathcal{ID}$ and $\mathsf{h}.\mathcal{CT} \subseteq \mathsf{r}.\mathcal{M}$, where $\mathsf{h}.\mathcal{ID}$ and $\mathsf{r}.\mathcal{ID}$ are the element identity space.

The SR-HIBE scheme $\Pi = (\mathsf{Setup}, \mathsf{UserKG}, \mathsf{UpdKG}, \mathsf{TranKG}, \mathsf{DecKG}, \mathsf{Enc}, \mathsf{Transform}, \mathsf{Dec}, \mathsf{Rev})$ with identity space $\mathcal{ID}_h = \mathsf{r}.\mathcal{ID}_h \subseteq \mathsf{h}.\mathcal{ID}_h$, where the element identity space $\mathcal{ID} = \mathsf{r}.\mathcal{ID} = \mathsf{h}.\mathcal{ID}$, the plaintext space $\mathcal{M} = \mathsf{h}.\mathcal{M}$, time period space $\mathcal{T} = \mathsf{r}.\mathcal{T} \subseteq \mathsf{h}.\mathcal{ID}$ and the ciphertext space $\mathcal{CT} = \mathsf{r}.\mathcal{CT}$. In order to satisfy the security guarantee, without lose of generality, we assume that if $\mathcal{ID} = \{0,1\}^n$, then for the identity $\mathsf{ID} = (\mathsf{id}_1, \cdots, \mathsf{id}_i) \in \mathcal{ID}_h$, $i \leq [L]$, $\mathsf{id}_i \in \mathcal{ID}' = \{0\} \times \{0,1\}^{n-1}$, and the time period space $\mathcal{T} = \{1\} \times \{0,1\}^{n-1}$. Namely, it should keep that $\mathcal{ID}' \cap \mathcal{T} = \emptyset$. The generic construction of SR-HIBE are as follows.

- $\mathsf{Setup}(1^\lambda, L) \to (\mathsf{mpk}, \mathsf{msk}_{\mathsf{kgc}}, \mathsf{RL}_{\mathsf{kgc}}, \mathsf{st}_{\mathsf{kgc}})$. It takes the security parameter 1^λ and the maximum depth of the hierarchy $L \in \mathbb{N}$ as inputs, the algorithm does as follows:
  1. Run $(\mathsf{r.mpk}, \mathsf{r.msk}_{\mathsf{kgc}}, \mathsf{r.RL}_{\mathsf{kgc}}, \mathsf{r.st}_{\mathsf{kgc}}) \leftarrow \mathsf{r.Setup}(1^\lambda, L)$,
  2. Run $(\mathsf{h.mpk}, \mathsf{h.msk}_{\mathsf{kgc}}) \leftarrow \mathsf{h.Setup}(1^\lambda)$.

  Then, output the public parameter $\mathsf{mpk} := (\mathsf{r.mpk}, \mathsf{h.mpk})$, the KGC's secret key $\mathsf{msk}_{\mathsf{kgc}} := (\mathsf{r.msk}_{\mathsf{kgc}}, \mathsf{h.msk}_{\mathsf{kgc}})$, the revocation list $\mathsf{RL}_{\mathsf{kgc}} := \mathsf{r.RL}_{\mathsf{kgc}}$ and the initial state $\mathsf{st}_{\mathsf{kgc}} := \mathsf{r.st}_{\mathsf{kgc}}$.
- $\mathsf{UserKG}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{pa(ID)}}, \mathsf{ID}, \mathsf{st}_{\mathsf{pa(ID)}}) \to (\mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{st}'_{\mathsf{pa(ID)}}, \mathsf{RL}_{\mathsf{ID}}, \mathsf{st}_{\mathsf{ID}})$. On input the public parameter $\mathsf{mpk} = (\mathsf{r.mpk}, \mathsf{h.mpk})$, the parent's secret key $\mathsf{sk}_{\mathsf{pa(ID)}} = (\mathsf{r.msk}_{\mathsf{pa(ID)}}, \mathsf{h.sk}_{\mathsf{pa(ID)}})$ and identity $\mathsf{ID} \in \mathcal{ID}_h$, the parent $\mathsf{pa(ID)}$ does as follows:
  1. $(\mathsf{r.sk}_{\mathsf{ID}}, \mathsf{r.msk}_{\mathsf{ID}}, \mathsf{r.RL}_{\mathsf{ID}}, \mathsf{r.st}_{\mathsf{ID}}, \mathsf{r.st}'_{\mathsf{pa(ID)}}) \leftarrow \mathsf{r.GenSK}(\mathsf{r.mpk}, \mathsf{r.msk}_{\mathsf{pa(ID)}}, \mathsf{ID}, \mathsf{r.st}_{\mathsf{pa(ID)}})$,
  2. $\mathsf{h.sk}_{\mathsf{ID}} \leftarrow \mathsf{h.Delegate}(\mathsf{h.mpk}, \mathsf{h.sk}_{\mathsf{pa(ID)}}, \mathsf{ID})$.[8]

  Then, output the public key $\mathsf{pk}_{\mathsf{ID}} := \mathsf{r.sk}_{\mathsf{ID}}$, the secret key $\mathsf{sk}_{\mathsf{ID}} := (\mathsf{r.msk}_{\mathsf{ID}}, \mathsf{h.sk}_{\mathsf{ID}})$, the parent's updated state $\mathsf{st}_{\mathsf{pa(ID)}} := \mathsf{r.st}'_{\mathsf{pa(ID)}}$, the revocation list $\mathsf{RL}_{\mathsf{ID}} := \mathsf{r.RL}_{\mathsf{ID}}$ and the initial state $\mathsf{st}_{\mathsf{ID}} := \mathsf{r.st}_{\mathsf{ID}}$.
- $\mathsf{UpdKG}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{t}, \mathsf{RL}_{\mathsf{ID,t}}, \mathsf{uk}_{\mathsf{pa(ID),t}}, \mathsf{st}_{\mathsf{ID}}) \to (\mathsf{uk}_{\mathsf{ID,t}}, \mathsf{st}'_{\mathsf{ID}})$. On input the public parameter $\mathsf{mpk} = (\mathsf{r.mpk}, \mathsf{h.mpk})$, the secret key $\mathsf{sk}_{\mathsf{ID}} = (\mathsf{r.msk}_{\mathsf{ID}}, \mathsf{h.sk}_{\mathsf{ID}})$, the time period $\mathsf{t} \in \mathcal{T}$ and a revocation list $\mathsf{RL}_{\mathsf{ID,t}} = \mathsf{r.RL}_{\mathsf{ID,t}}$, the parent's update key $\mathsf{uk}_{\mathsf{pa(ID),t}} = \mathsf{r.ku}_{\mathsf{pa(ID),t}}$ and the state $\mathsf{st}_{\mathsf{ID}} = \mathsf{r.st}_{\mathsf{ID}}$, run

$$(\mathsf{r.ku}_{\mathsf{ID,t}}, \mathsf{r.st}'_{\mathsf{ID}}) \leftarrow \mathsf{r.KeyUp}(\mathsf{r.mpk}, \mathsf{r.msk}_{\mathsf{ID}}, \mathsf{t}, \mathsf{r.RL}_{\mathsf{ID,t}}, \mathsf{r.ku}_{\mathsf{pa(ID),t}}, \mathsf{r.st}_{\mathsf{ID}}).$$

  Then, output an update key $\mathsf{uk}_{\mathsf{ID,t}} := \mathsf{r.ku}_{\mathsf{ID,t}}$ and the updated state $\mathsf{st}'_{\mathsf{ID}} := \mathsf{r.st}'_{\mathsf{ID}}$.
- $\mathsf{TranKG}(\mathsf{mpk}, \mathsf{pk}_{\mathsf{ID}}, \mathsf{uk}_{\mathsf{pa(ID),t}}) \to \mathsf{tk}_{\mathsf{ID,t}}$. On input the public parameter $\mathsf{mpk} = (\mathsf{r.mpk}, \mathsf{h.mpk})$, a public key $\mathsf{pk}_{\mathsf{ID}} = \mathsf{r.sk}_{\mathsf{ID}}$ and the update key $\mathsf{uk}_{\mathsf{pa(ID),t}} = \mathsf{r.ku}_{\mathsf{pa(ID),t}}$, the server runs

$$\mathsf{r.dk}_{\mathsf{ID,t}} \leftarrow \mathsf{r.GenDK}(\mathsf{r.mpk}, \mathsf{r.sk}_{\mathsf{ID}}, \mathsf{r.ku}_{\mathsf{pa(ID),t}}),$$

  and outputs a transform key $\mathsf{tk}_{\mathsf{ID,t}} := \mathsf{r.dk}_{\mathsf{ID,t}}$ for identity ID in time period t.
- $\mathsf{DecKG}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{t}) \to \mathsf{dk}_{\mathsf{ID,t}}$. On input the public parameter $\mathsf{mpk} = (\mathsf{r.mpk}, \mathsf{h.mpk})$, the secret key $\mathsf{sk}_{\mathsf{ID}} = (\mathsf{r.msk}_{\mathsf{ID}}, \mathsf{h.sk}_{\mathsf{ID}})$ and the time period t, the user $\mathsf{ID} \in \mathcal{ID}_h$ runs

$$\mathsf{h.sk}_{\mathsf{ID,t}} \leftarrow \mathsf{h.Delegate}(\mathsf{h.mpk}, \mathsf{h.sk}_{\mathsf{ID}}, \mathsf{t})$$

  and outputs a decryption key $\mathsf{dk}_{\mathsf{ID,t}} := \mathsf{h.sk}_{\mathsf{ID,t}}$.
- $\mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}, \mathsf{t}, M) \to \mathsf{ct}_{\mathsf{ID,t}}$. On input the public parameter $\mathsf{mpk} = (\mathsf{r.mpk}, \mathsf{h.mpk})$, an identity $\mathsf{ID} \in \mathcal{ID}_h$, a time period $\mathsf{t} \in \mathcal{T}$ and a message $M \in \mathcal{M}$, the algorithm does as follows:

[8] When ID is level-1 user, $\mathsf{h.sk}_{\mathsf{pa(ID)}} = \mathsf{h.msk}_{\mathsf{kgc}}$, and $\mathsf{h.sk}_{\mathsf{ID}} \leftarrow \mathsf{h.GenSK}(\mathsf{h.mpk}, \mathsf{h.msk}_{\mathsf{kgc}}, \mathsf{ID})$

1. $\mathsf{h.ct}_{\mathsf{ID},\mathsf{t}} \leftarrow \mathsf{h.Enc}(\mathsf{h.mpk}, (\mathsf{ID}, \mathsf{t}), M)$,
2. $\mathsf{r.ct}_{\mathsf{ID},\mathsf{t}} \leftarrow \mathsf{r.Enc}(\mathsf{r.mpk}, \mathsf{ID}, \mathsf{t}, \mathsf{h.ct}_{\mathsf{ID},\mathsf{t}})$.

Finally, output the ciphertext $\mathsf{ct}_{\mathsf{ID},\mathsf{t}} := \mathsf{r.ct}_{\mathsf{ID},\mathsf{t}}$.

- $\mathsf{Transform}(\mathsf{mpk}, \mathsf{ct}_{\mathsf{ID},\mathsf{t}}, \mathsf{tk}_{\mathsf{ID},\mathsf{t}}) \rightarrow \mathsf{ct}'_{\mathsf{ID},\mathsf{t}}$. On input the public parameter $\mathsf{mpk} = (\mathsf{r.mpk}, \mathsf{h.mpk})$, a ciphertext $\mathsf{ct}_{\mathsf{ID},\mathsf{t}} = \mathsf{r.ct}_{\mathsf{ID},\mathsf{t}}$ and a transform key $\mathsf{tk}_{\mathsf{ID},\mathsf{t}} := \mathsf{r.dk}_{\mathsf{ID},\mathsf{t}}$ for identity ID in time period t, run

$$\mathsf{h.ct}'_{\mathsf{ID},\mathsf{t}} \leftarrow \mathsf{r.Dec}(\mathsf{r.mpk}, \mathsf{r.dk}_{\mathsf{ID},\mathsf{t}}, \mathsf{r.ct}_{\mathsf{ID},\mathsf{t}}).$$

  If $\mathsf{h.ct}'_{\mathsf{ID},\mathsf{t}} = \perp$, return $\perp$. Otherwise, output a partial decrypted ciphertext $\mathsf{ct}'_{\mathsf{ID},\mathsf{t}} := \mathsf{h.ct}'_{\mathsf{ID},\mathsf{t}}$.
- $\mathsf{Dec}(\mathsf{PP}, \mathsf{ct}'_{\mathsf{ID},\mathsf{t}}, \mathsf{dk}_{\mathsf{ID},\mathsf{t}}) \rightarrow M$. On input the public parameter $\mathsf{mpk} = (\mathsf{r.mpk}, \mathsf{h.mpk})$, a partial decrypted ciphertext $\mathsf{ct}'_{\mathsf{ID},\mathsf{t}} = \mathsf{h.ct}'_{\mathsf{ID},\mathsf{t}}$ and a decryption key $\mathsf{dk}_{\mathsf{ID},\mathsf{t}} = \mathsf{h.sk}_{\mathsf{ID},\mathsf{t}}$ for identity ID in time period t, run

$$\mathsf{h}.M' \leftarrow \mathsf{h.Dec}(\mathsf{h.mpk}, \mathsf{h.sk}_{\mathsf{ID},\mathsf{t}}, \mathsf{h.ct}'_{\mathsf{ID},\mathsf{t}}).$$

  If $\mathsf{h}.M' = \perp$, return $\perp$. Otherwise, output a message $M := \mathsf{h}.M'$.
- $\mathsf{Rev}((\mathsf{ID}, \mathsf{t}), \mathsf{RL}_{\mathsf{pa(ID)},\mathsf{t}}, \mathsf{st}_{\mathsf{pa(ID)}}) \rightarrow \mathsf{RL}'_{\mathsf{pa(ID)},\mathsf{t}}$. On input $(\mathsf{ID}, \mathsf{t}) \in \mathcal{ID}_h \times \mathcal{T}$, the revocation list $\mathsf{RL}_{\mathsf{pa(ID)},\mathsf{t}} = \mathsf{r.RL}_{\mathsf{pa(ID)},\mathsf{t}}$ and the state $\mathsf{st}_{\mathsf{pa(ID)}} = \mathsf{r.st}_{\mathsf{pa(ID)}}$, run

$$\mathsf{r.RL}'_{\mathsf{pa(ID)},\mathsf{t}} \leftarrow \mathsf{r.Rev}((\mathsf{ID}, \mathsf{t}), \mathsf{r.RL}_{\mathsf{pa(ID)},\mathsf{t}}, \mathsf{r.st}_{\mathsf{pa(ID)}})$$

  and output the updated revocation list $\mathsf{RL}'_{\mathsf{pa(ID)},\mathsf{t}} := \mathsf{r.RL}'_{\mathsf{pa(ID)},\mathsf{t}}$.

Correctness. It is immediate to see that the correctness of the constructed SR-HIBE scheme Π follows from that of the underlying RHIBE scheme $\mathsf{r}.\Pi$ and the HIBE scheme $\mathsf{h}.\Pi$.

Theorem 1. *If the underlying RHIBE scheme* $\mathsf{r}.\Pi$ *satisfies weak selective-identity security (resp. weak adaptive-identity security) and the underlying (L+1)-level HIBE scheme* $\mathsf{h}.\Pi$ *satisfies selective-identity security (resp. adaptive-identity security), then the constructed constructed SR-HIBE scheme* Π *satisfies selective-identity security (resp. adaptive-identity security).*

Proof (of Theorem 1). Here we only give the proof for the selective-identity security since it is essentially the same as that for the adaptive one. Let $(\mathsf{ID}^*, \mathsf{t}^*)$ be the challenge identity/time period pair. We call a query made by an adversary is valid, if the challenger's answer is not "$\perp$". The strategies that an adversary used to against the SR-HIBE scheme Π can be divided into the following cases:

– Type-I: The adversary issues valid secret key reveal queries on at least one $\mathsf{ID} \in \mathsf{prefix}(\mathsf{ID}^*)$.

- Type-I-i^*: The adversary issues a valid secret key reveal query on $\mathsf{ID}^*_{[i^*]}$ but not on any $\mathsf{ID} \in \mathsf{prefix}(\mathsf{ID}^*_{[i^*-1]})$.
- Type-II: The adversary does not issue valid secret key reveal queries on any $\mathsf{ID} \in \mathsf{prefix}(\mathsf{ID}^*)$.

By the strategy-dividing lemma in [15], in order to prove the theorem, it is sufficient to show that for each type of adversary, its advantage is negligible.

Lemma 1. *If there exists a PPT Type-I-i^* adversary $\mathcal{A}_{1-i^*}$ breaking the selective-identity security of SR-HIBE scheme Π with advantage ϵ, then there exists a PPT adversary $\mathcal{B}$ against the weak selective-identity security of the underlying RHIBE scheme $\mathsf{r}.\Pi$ with the same advantage.*

Proof (of Lemma 1). Recall that the condition of the secret key reveal query is that if none of $\mathsf{ID}' \in \mathsf{prefix}(\mathsf{ID}^*)$ has been revoked before t^*, i.e., $\mathsf{ID}' \notin \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}'),\mathsf{t}^*}$, then $\mathsf{sk}_{\mathsf{ID}'}$ will not be revealed after t^*. In revoke & update key query, the condition means that if $\mathcal{A}$ has issued a valid secret key reveal query on $\mathsf{ID}^*_{[i^*]}$ before t^* and $\mathsf{t}_{\mathsf{cu}} \geq \mathsf{t}^*$, then $\mathsf{ID}^*_{[i^*]} \in \mathsf{RL}_{\mathsf{ID}^*_{[i^*-1]},\mathsf{t}^*}$ and thus $\mathsf{ID}' \in \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}'),\mathsf{t}^*}$ for all $\mathsf{ID}' \in \mathsf{prefix}(\mathsf{ID}^*) \setminus \mathsf{prefix}(\mathsf{ID}^*_{[i^*-1]})$. In particular, if $\mathsf{t}_{\mathsf{cu}} \geq \mathsf{t}^*$, it guarantees that $\mathsf{ID}^* \in \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}^*),\mathsf{t}^*}$. Let $\mathcal{A}_{1-i^*}$ be a PPT Type-I-i^* adversary that breaking the selective-identity secure of SR-HIBE scheme Π. We can construct a PPT adversary $\mathcal{B}$ that can break the weak selective-identity secure of the RHIBE scheme $\mathsf{r}.\Pi$ by using the ability of $\mathcal{A}_{1-i^*}$ as following.

Initial. $\mathcal{A}_{1-i^*}$ announces to $\mathcal{B}$ the challenge identity/time period pair $(\mathsf{ID}^*, \mathsf{t}^*) \in \mathcal{ID}_h \times \mathcal{T}$, and $\mathcal{B}$ forwards them to the RHIBE challenger $\mathsf{r}.\mathcal{C}$ as its own challenge identity/time period pair.

Setup. After $\mathsf{r}.\mathcal{C}$ receives $(\mathsf{ID}^*, \mathsf{t}^*)$, it generates and sends $\mathsf{r.mpk}$ and $\mathsf{r.ku}_{\mathsf{kgc},1}$ to $\mathcal{B}$. Then $\mathcal{B}$ runs $\mathsf{h.Setup}(1^\lambda) \rightarrow (\mathsf{h.mpk}, \mathsf{h.msk}_{\mathsf{kgc}})$ and sends the public parameter $\mathsf{mpk} = (\mathsf{r.mpk}, \mathsf{h.mpk})$ and $\mathsf{uk}_{\mathsf{kgc},1} := \mathsf{r.ku}_{\mathsf{kgc},1}$ to $\mathcal{A}_{1-i^*}$. As in the real selective-identity security game, $\mathcal{B}$ also prepares a table T which initially contains $(\mathsf{kgc}, \perp, \mathsf{msk}_{\mathsf{kgc}} := (\perp, \mathsf{h.msk}_{\mathsf{kgc}}))$, and initializes a counter $\mathsf{t}_{\mathsf{cu}} := 1$ which always is synchronized by the one maintained by $\mathsf{r}.\mathcal{C}$.

Public key reveal query. When $\mathcal{A}_{1-i^*}$ issues a public key reveal query $\mathsf{ID} \in \mathcal{ID}_h$, $\mathcal{B}$ forwards it to the RHIBE challenger $\mathsf{r}.\mathcal{C}$ as a secret key generation query. Then, $\mathsf{r}.\mathcal{C}$ first makes the checks as in the selective-identity secure game of RHIBE. If it does not hold, return $\perp$ to $\mathcal{B}$ and $\mathcal{B}$ returns it to $\mathcal{A}_{1-i^*}$. Otherwise, run $(\mathsf{r.sk}_{\mathsf{ID}}, \mathsf{r.msk}_{\mathsf{ID}}, \mathsf{r.st}') \leftarrow \mathsf{r.GenSK}(\mathsf{r.mpk}, \mathsf{r.msk}_{\mathsf{pa}(\mathsf{ID})}, \mathsf{ID}, \mathsf{r.st})$. If $\mathsf{ID} \in (\mathcal{ID})^{\leq L-1}$, then $\mathsf{r}.\mathcal{C}$ further executes $(\mathsf{r}, \mathsf{ku}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}}, \mathsf{r.st}'_{\mathsf{ID}}) \leftarrow \mathsf{r.KeyUp}(\mathsf{r.mpk}, \mathsf{r.msk}_{\mathsf{ID}}, \mathsf{t}_{\mathsf{cu}}, \mathsf{r.RL}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}} = \emptyset, \mathsf{r.ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}_{\mathsf{cu}}}, \mathsf{r.st}_{\mathsf{ID}})$. Finally, $\mathsf{r}.\mathcal{C}$ sends the update key $\mathsf{r.ku}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}}$ to $\mathcal{B}$ when $\mathsf{ID} \in (\mathcal{ID})^{\leq L-1}$, or returns nothing to $\mathcal{B}$ if $\mathsf{ID} \in (\mathcal{ID})^L$. After that, $\mathcal{B}$ issues a secret key reveal query on ID to $\mathsf{r}.\mathcal{C}$ and receives $(\mathsf{r.sk}_{\mathsf{ID}}, \mathsf{r.msk}_{\mathsf{ID}})$. Then, $\mathcal{B}$

runs $\mathsf{h.Delegate}(\mathsf{h.mpk}, \mathsf{h.sk}_{\mathsf{pa(ID)}}, \mathsf{ID}) \rightarrow \mathsf{h.sk}_{\mathsf{ID}}$, stores the entry $(\mathsf{ID}, \mathsf{pk}_{\mathsf{ID}} := \mathsf{r.sk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}} := (\mathsf{r.msk}_{\mathsf{ID}}, \mathsf{h.sk}_{\mathsf{ID}}))$ in table T, and returns $\mathsf{pk}_{\mathsf{ID}}$ to $\mathcal{A}_{1-i^*}$.

Secret key reveal query. When $\mathcal{A}_{1-i^*}$ makes a secret key reveal query $\mathsf{ID} \in \mathcal{ID}_h$, $\mathcal{B}$ first checks if the condition that if $\mathsf{t_{cu}} \geq \mathsf{t}^*$ and $\mathsf{ID}' \in \mathsf{RL}_{\mathsf{pa(ID')},\mathsf{t}^*}$ for all $\mathsf{ID}' \in \mathsf{prefix}(\mathsf{ID}^*)$, then $\mathsf{ID} \notin \mathsf{prefix}(\mathsf{ID}^*) \setminus \mathsf{prefix}(\mathsf{ID}^*_{[i^*-1]})$ holds. Since $\mathcal{A}_{1-i^*}$ uses Type-I-i^* strategy, it does not issue a valid secret key reveal query on any $\mathsf{ID} \in \mathsf{prefix}(\mathsf{ID}^*_{[i^*-1]})$. If it does not hold, return $\perp$ to $\mathcal{A}_{1-i^*}$. Otherwise, it guarantees that there exists an entry indexed by ID in table T, then $\mathcal{B}$ obtains the entry $(\mathsf{ID}, \mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}} = (\mathsf{r.msk}_{\mathsf{ID}}, \mathsf{h.sk}_{\mathsf{ID}}))$ and returns $\mathsf{sk}_{\mathsf{ID}}$ to $\mathcal{A}_{1-i^*}$.

Revoke & update key query. When $\mathcal{A}_{1-i^*}$ issues a revoke & update key query $\mathsf{RL} \in \mathcal{ID}_h$, $\mathcal{B}$ forwards it to the RIBE challenger $\mathsf{r}.\mathcal{C}$. Then $\mathsf{r}.\mathcal{C}$ does the same check as in the selective-identity security game. If those are not hold, return $\perp$ to $\mathcal{B}$, and $\mathcal{B}$ it to $\mathcal{A}$. Otherwise, $\mathsf{r}.\mathcal{C}$ sets $\mathsf{t_{cu}} := \mathsf{t_{cu}} + 1$ and so does $\mathcal{B}$. Then $\mathsf{r}.\mathcal{C}$ does the same computations as in the selective-identity security game and returns the key updates $\mathsf{r.ku}_{\mathsf{ID},\mathsf{t_{cu}}}$ to $\mathcal{B}$. And $\mathcal{B}$ returns the update keys $\mathsf{uk}_{\mathsf{ID},\mathsf{t_{cu}}} := \mathsf{r.ku}_{\mathsf{ID},\mathsf{t_{cu}}}$ to $\mathcal{A}_{1-i^*}$. Here, as mentioned at the beginning, if $\mathsf{t_{cu}} \geq \mathsf{t}^*$, then it holds that $\mathsf{ID}^*_{[i^*]} \in \mathsf{RL}_{\mathsf{ID}^*_{[i^*]-1},\mathsf{t}^*}$, thus $\mathsf{ID}^* \in \mathsf{RL}_{\mathsf{pa(ID}^*),\mathsf{t}^*}$.

Decryption key reveal query. When $\mathcal{A}_{1-i^*}$ makes a decryption key reveal query $(\mathsf{ID}, \mathsf{t}) \in \mathcal{ID}_h \times \mathcal{T}$, $\mathcal{B}$ first checks whether $\mathsf{t} \leq \mathsf{t_{cu}}$, $\mathsf{ID} \notin \mathsf{RL}_{\mathsf{pa(ID)},\mathsf{t}}$ and $(\mathsf{ID}, \mathsf{t}) \neq (\mathsf{ID}^*, \mathsf{t}^*)$ hold simultaneously. If this is not the case, return $\perp$ to $\mathcal{A}_{1-i^*}$. Otherwise, $\mathcal{B}$ can answer the query since it possesses $\mathsf{h.sk}_{\mathsf{ID}}$ (As we assumed before that all the identities appeared in the queries has been queried a public key reveal query, thus there exists an entry $(\mathsf{ID}, \mathsf{pk}_{\mathsf{ID}} = \mathsf{r.sk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}} = (\mathsf{r.msk}_{\mathsf{ID}}, \mathsf{h.sk}_{\mathsf{ID}}))$ in table T). Thus, $\mathcal{B}$ runs $\mathsf{h.sk}_{\mathsf{ID},\mathsf{t}} \leftarrow \mathsf{h.Delegate}(\mathsf{h.mpk}, \mathsf{h.sk}_{\mathsf{ID}}, \mathsf{t})$ and returns $\mathsf{dk}_{\mathsf{ID},\mathsf{t}} := \mathsf{h.sk}_{\mathsf{ID},\mathsf{t}}$ to $\mathcal{A}_{1-i^*}$.

Challenge phase. Once the adversary $\mathcal{A}_{1-i^*}$ submits two messages M_0, M_1 with equal length, $\mathcal{B}$ generates the challenge ciphertext as follows:

1. Set $M'_0 = \mathsf{h.Enc}(\mathsf{h.mpk}, (\mathsf{ID}^*, \mathsf{t}^*), M_0)$, $M'_1 = \mathsf{h.Enc}(\mathsf{h.mpk}, (\mathsf{ID}^*, \mathsf{t}^*), M_1)$.
2. Choose a random bit $b \xleftarrow{\$} \{0,1\}$, set $M''_1 = M'_b$, $M''_0 = M'_{1\oplus b}$, where $\oplus$ denotes the addition modulo 2. Then send M''_0 and M''_1 to the RHIBE challenger $\mathsf{r}.\mathcal{C}$ as the challenge messages that he chooses.
3. $\mathsf{r}.\mathcal{C}$ chooses a random bit $\hat{b} \xleftarrow{\$} \{0,1\}$, generates the RHIBE challenge ciphertext $\mathsf{r.ct}^{(\hat{b})}_{\mathsf{ID}^*,\mathsf{t}^*} \leftarrow \mathsf{r.Enc}(\mathsf{r.mpk}, \mathsf{ID}^*, \mathsf{t}^*, M''_{\hat{b}})$ and sends $\mathsf{r.ct}^{(\hat{b})}_{\mathsf{ID}^*,\mathsf{t}^*}$ to $\mathcal{B}$.
4. $\mathcal{B}$ sets the challenge ciphertext $\mathsf{ct}^*_{\mathsf{ID}^*,\mathsf{t}^*} = \mathsf{r.ct}^{(\hat{b})}_{\mathsf{ID}^*,\mathsf{t}^*}$ and sends it to $\mathcal{A}_{1-i^*}$. Note that the challenge ciphertext $\mathsf{ct}^*_{\mathsf{ID}^*,\mathsf{t}^*}$ is an SR-HIBE encryption on $M_{b\oplus\hat{b}}$ under $(\mathsf{ID}^*, \mathsf{t}^*)$ since $M''_{\hat{b}} = M_{b\oplus\hat{b}}$, and the bit $b \oplus \hat{b}$ is uniformly random in $\{0,1\}$.

Guess. At some point, $\mathcal{A}_{1-i^*}$ make a guess $b' \xleftarrow{\$} \{0,1\}$ that $\mathsf{ct}^*_{\mathsf{ID}^*,\mathsf{t}^*}$ is an encryption of $M_{b'}$. Then $\mathcal{B}$ computes and sends $\hat{b}' = b \oplus b'$ to the RHIBE challenger $\mathsf{r}.\mathcal{C}$ as its guess for the bit $\hat{b}$.

By the assumption,

$$\mathbf{Adv}^{\mathsf{SR-HIBE-sel}}_{\mathcal{SR}-\mathcal{HIBE},\mathcal{A}_{1-i^*}}(1^\lambda) = \left|\Pr[b' = b \oplus \hat{b}] - \frac{1}{2}\right| = \epsilon.$$

While by the construction, it holds that $b' = b \oplus \hat{b} \Longleftrightarrow b' \oplus b = \hat{b} \Longleftrightarrow \hat{b} = \hat{b}'$. Thus,

$$\mathbf{Adv}^{\mathsf{RHIBE-sel-weak}}_{\mathcal{RHIBE},\mathcal{B}}(1^\lambda) = \left|\Pr[\hat{b}' = \hat{b}] - \frac{1}{2}\right| = \epsilon.$$

Thus, it holds that

$$\mathbf{Adv}^{\mathsf{RHIBE-sel-weak}}_{\mathcal{RHIBE},\mathcal{B}}(1^\lambda) = \mathbf{Adv}^{\mathsf{SR-HIBE-sel}}_{\mathcal{SR}-\mathcal{HIBE},\mathcal{A}_{1-i^*}}(1^\lambda),$$

as desired. This complete the proof of Lemma 1. □

Lemma 2. *If there exists a PPT Type-II adversary $\mathcal{A}_2$ breaking the selective-identity security of SR-HIBE scheme Π with advantage ϵ, then there exists a PPT adversary $\mathcal{B}$ against the selective-identity security of the underlying $(L+1)$-level HIBE scheme $\mathsf{h}.\Pi$ with the same advantage.*

Proof (of Lemma 2). Let $\mathcal{A}_2$ be the PPT Type-II adversary, then we can construct a PPT adversary $\mathcal{B}$ that can break the selective-identity security of the underlying HIBE scheme $\mathsf{h}.\Pi$ by using the ability of $\mathcal{A}_2$. Note that $\mathcal{A}_2$ uses Type-II strategy, it has never issues valid secret key reveal query on any $\mathsf{ID} \in \mathsf{prefix}(\mathsf{ID}^*)$, so $\mathcal{A}_2$ cannot issues decryption key reveal query on $(\mathsf{ID}^*, \mathsf{t}^*)$. The description of $\mathcal{B}$ is as follows:

Initial. $\mathcal{A}_2$ announces to $\mathcal{B}$ the challenge identity/time period pair $(\mathsf{ID}^*, \mathsf{t}^*) \in \mathcal{ID}_h \times \mathcal{T}$, and $\mathcal{B}$ forwards them to the HIBE challenger $\mathsf{h}.\mathcal{C}$ as the challenge on its own.

Setup. After $\mathsf{h}.\mathcal{C}$ receives $(\mathsf{ID}^*, \mathsf{t}^*)$, it runs $(\mathsf{h.mpk}, \mathsf{h.msk_{kgc}}) \leftarrow \mathsf{h.Setup}(1^\lambda)$ and sends $\mathsf{h.mpk}$ to $\mathcal{B}$. Then $\mathcal{B}$ initializes the counter $\mathsf{t_{cu}} = 1$, executes $(\mathsf{r.mpk}, \mathsf{r.msk_{kgc}}, \mathsf{r.RL_{kgc,1}}, \mathsf{r.st_{kgc}}) \leftarrow \mathsf{r.Setup}(1^\lambda, L)$ and $(\mathsf{r.ku_{kgc,1}}, \mathsf{r.st'_{kgc}}) \leftarrow \mathsf{r.KeyUp}(\mathsf{r.mpk}, \mathsf{r.msk_{kgc}}, \mathsf{t_{cu}} = 1, \mathsf{r.RL_{kgc,1}}, \mathsf{r.st_{kgc}})$, sets $\mathsf{RL_{kgc,t_{cu}}} := \mathsf{r.RL_{kgc,t_{cu}}}$, $\mathsf{st_{kgc}} := \mathsf{r.st_{kgc}}$ and sends the public parameter $\mathsf{mpk} = (\mathsf{r.mpk}, \mathsf{h.mpk})$ to $\mathcal{A}_2$. As in the real selective-identity security game, $\mathcal{B}$ also prepares a table T which initially contains $(\mathsf{kgc}, \perp, \mathsf{msk_{kgc}} := (\mathsf{r.msk_{kgc}}, \perp))$.

Public key reveal query. When $\mathcal{A}_2$ makes a public key reveal query on level-i user $\mathsf{ID} \in \mathcal{ID}_h$, $i \in [L]$, $\mathcal{B}$ forwards it to the HIBE challenger $\mathsf{h}.\mathcal{C}$ as a level-i secret key generation query. $\mathsf{h}.\mathcal{C}$ first makes the same check as in selective-identity security game of HIBE. If not, return $\perp$. Otherwise, $\mathsf{h}.\mathcal{C}$ runs $\mathsf{h.sk_{ID}} \leftarrow \mathsf{h.Delegate}(\mathsf{h.mpk}, \mathsf{h.sk_{pa(ID)}}, \mathsf{ID})$ and returns nothing to $\mathcal{B}$. Then, $\mathcal{B}$ issues a level-i secret key reveal query on the same ID. If $\mathsf{h}.\mathcal{C}$ returns $\perp$ to $\mathcal{B}$, then $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ executes $(\mathsf{r.sk_{ID}}, \mathsf{r.msk_{ID}}, \mathsf{r.RL_{ID}}, \mathsf{r.st_{ID}}, \mathsf{r.st'_{pa(ID)}}) \leftarrow \mathsf{r.GenSK}(\mathsf{r.mpk}, \mathsf{r.msk_{pa(ID)}}, \mathsf{ID}, \mathsf{r.st_{pa(ID)}})$ and stores the entry $(\mathsf{ID}, \mathsf{pk_{ID}} := \mathsf{r.sk_{ID}}, \mathsf{sk_{ID}} :=$

$(\mathsf{r.msk_{ID}}, \mathsf{h.sk_{ID}})$) in table T. If $i = L$, $\mathcal{B}$ returns $\mathsf{pk_{ID}} := \mathsf{r.sk_{ID}}$ to $\mathcal{A}_2$. If $i \leq L-1$, $\mathcal{B}$ further executes $(\mathsf{r.ku_{ID,t_{cu}}}, \mathsf{r.st'_{ID}}) \leftarrow \mathsf{r.KeyUp}(\mathsf{r.mpk}, \mathsf{r.msk_{ID}}, \mathsf{t_{cu}}, \mathsf{r.RL_{ID,t_{cu}}}, \mathsf{r.ku_{pa(ID),t_{cu}}}, \mathsf{r.st_{ID}})$, and returns $\mathsf{pk_{ID}}$ and $\mathsf{r.ku_{ID,t_{cu}}}$ to $\mathcal{A}_2$.

Secret key reveal query. When $\mathcal{A}_2$ issues a secret key reveal query $\mathsf{ID} \in \mathcal{ID}_h$, $\mathcal{B}$ first checks the condition that if $\mathsf{t_{cu}} \geq \mathsf{t}^*$ and $\mathsf{ID}' \notin \mathsf{RL_{pa(ID'),t^*}}$ for all $\mathsf{ID}' \in \mathsf{prefix(ID^*)}$, then $\mathsf{ID} \notin \mathsf{prefix(ID^*)}$ holds. If not, return $\perp$ to $\mathcal{A}_2$. Otherwise, it guaranteed that there is an entry indexed by ID exists in table T, and $\mathsf{ID} \notin \mathsf{prefix(ID^*)}$ since $\mathcal{A}_2$ uses Type-II strategy. Then $\mathcal{B}$ obtains the entry $(\mathsf{ID}, \mathsf{pk_{ID}} = \mathsf{r.sk_{ID}}, \mathsf{sk_{ID}} = (\mathsf{r.msk_{ID}}, \mathsf{h.sk_{ID}}))$ from table T and returns $\mathsf{sk_{ID}} = (\mathsf{r.msk_{ID}}, \mathsf{h.sk_{ID}})$ to $\mathcal{A}_2$.

Revoke & update key query. When $\mathcal{A}_2$ makes a query $\mathsf{RL} \in \mathcal{ID}_h$, $\mathcal{B}$ does the same check as in the selective-identity security game. If not, return $\perp$ to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ increases $\mathsf{t_{cu}} := \mathsf{t_{cu}} + 1$ and does the same computations and can answer $\mathcal{A}_2$ as in the selective-identity security game.

Decryption key reveal query. When $\mathcal{A}_2$ makes a decryption key reveal query $(\mathsf{ID}, \mathsf{t}) \in \mathcal{ID}_h \times \mathcal{T}$, $\mathcal{B}$ first checks whether $\mathsf{t} \leq \mathsf{t_{cu}}$, $\mathsf{ID} \notin \mathsf{RL_{pa(ID),t}}$ and $(\mathsf{ID}, \mathsf{t}) \neq (\mathsf{ID}^*, \mathsf{t}^*)$ hold simultaneously. If this is not the case, return $\perp$ to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ sends $(\mathsf{ID}, \mathsf{t})$ to the HIBE challenger $\mathsf{h}.\mathcal{C}$ as a level-$(i, i+1)$ secret key reveal query. As we assumed before that all the identities appeared in the queries has been queried a public key reveal query, and the condition $\mathsf{ID} \notin \mathsf{RL_{pa(ID),t}}$ guarantees that $\mathsf{pa(ID)}$ is not revoked at time t^*. Thus, it guarantees that $\mathsf{h}.\mathcal{C}$ has already generated $\mathsf{h.sk_{ID}}$. Thus, $\mathsf{h}.\mathcal{C}$ runs $\mathsf{h.sk_{ID,t}} \leftarrow \mathsf{h.Delegate}(\mathsf{h.mpk}, \mathsf{h.sk_{ID}}, \mathsf{t})$ and sends $\mathsf{h.sk_{ID,t}}$ to $\mathcal{B}$. Then, $\mathcal{B}$ returns $\mathsf{dk_{ID,t}} := \mathsf{h.sk_{ID,t}}$ to $\mathcal{A}_2$.

Challenge phase. Once the adversary $\mathcal{A}_2$ submits two messages M_0, M_1 with equal length, $\mathcal{B}$ generates the challenge ciphertext as follows:

1. Choose a random bit $b \xleftarrow{\$} \{0,1\}$, set and send $M'_0 = M_b$ and $M'_1 = M_{1 \oplus b}$ to $\mathsf{h}.\mathcal{C}$ as the challenge messages.
2. $\mathsf{h}.\mathcal{C}$ chooses a random bit $\hat{b} \xleftarrow{\$} \{0,1\}$, generates the HIBE challenge ciphertext $\mathsf{h.ct}^{(\hat{b})}_{\mathsf{ID}^*,\mathsf{t}^*} \leftarrow \mathsf{h.Enc}(\mathsf{h.mpk}, (\mathsf{ID}^*, \mathsf{t}^*), M'_{\hat{b}})$ and sends $\mathsf{h.ct}^{(\hat{b})}_{\mathsf{ID}^*,\mathsf{t}^*}$ to $\mathcal{B}$.
3. $\mathcal{B}$ runs $\mathsf{r.ct}^*_{\mathsf{ID}^*,\mathsf{t}^*} \leftarrow \mathsf{r.Enc}(\mathsf{r.mpk}, \mathsf{ID}^*, \mathsf{t}^*, \mathsf{h.ct}^{(\hat{b})}_{\mathsf{ID}^*,\mathsf{t}^*})$, set the SR-HIBE challenge ciphertext $\mathsf{ct}^*_{\mathsf{ID}^*,\mathsf{t}^*} := \mathsf{r.ct}^*_{\mathsf{ID}^*,\mathsf{t}^*}$ and sends it to $\mathcal{A}_2$. Note that $\mathsf{ct}^*_{\mathsf{ID}^*,\mathsf{t}^*}$ is an SR-HIBE encryption on $M_{\hat{b} \oplus b}$ under $(\mathsf{ID}^*, \mathsf{t}^*)$ since $M'_{\hat{b}} = M_{\hat{b} \oplus b}$ and the bit $\hat{b} \oplus b$ is uniformly random in $\{0,1\}$.

Guess. $\mathcal{A}_2$ outputs a guess $b' \xleftarrow{\$} \{0,1\}$ that $\mathsf{ct}^*_{\mathsf{ID}^*,\mathsf{t}^*}$ is an encryption of $M_{b'}$. Then $\mathcal{B}$ computes $\hat{b}' = b \oplus b'$ and returns $\hat{b}'$ to $\mathsf{h}.\mathcal{C}$ as the guess for the bit $\hat{b}$ chosen by $\mathsf{h}.\mathcal{C}$.

By the assumption,

$$\mathbf{Adv}^{\mathsf{SR-HIBE-sel}}_{\mathcal{SR}-\mathcal{HIBE},\mathcal{A}_2}(1^\lambda) = \left| \Pr[b' = b \oplus \hat{b}] - \frac{1}{2} \right| = \epsilon.$$

Besides, by the construction, $b' = b \oplus \hat{b} \Longleftrightarrow b' \oplus b = \hat{b} \Longleftrightarrow \hat{b} = \hat{b}'$. Thus,

$$\mathbf{Adv}^{\mathsf{HIBE-sel}}_{\mathcal{HIBE},\mathcal{B}}(1^\lambda) = \left|\Pr[\hat{b}' = \hat{b}] - \frac{1}{2}\right| = \epsilon.$$

Thus, it holds that

$$\mathbf{Adv}^{\mathsf{HIBE-sel}}_{\mathcal{HIBE},\mathcal{B}}(1^\lambda) = \mathbf{Adv}^{\mathsf{SR-HIBE-sel}}_{\mathcal{SR-HIBE},\mathcal{A}_2}(1^\lambda),$$

as desired. This complete the proof of Lemma 2. □

Combining Lemmas 1, 2 and the strategy-dividing lemma in [15], we can conclude that the constructed SR-HIBE scheme Π satisfies selective-identity security. This complete the proof of Theorem 1. □

3.1 Generic Construction of SR-IBE with DKER from any IBE and two-level HIBE

When the maximal hierarchical depth $L = 1$ in our generic construction, i.e., for the SR-IBE case, we obtain a generic construction of SR-IBE scheme with DKER from RIBE without DKER and two-level HIBE. While Ma and Lin [23] gave a generic construction of RIBE without DKER from any IBE scheme. Combining their work with our construction, we can get a generic construction of SR-IBE scheme with DKER from any IBE scheme and two-level HIBE scheme.

Observe that Ma and Lin [23] said their generic construction of RIBE with DKER from any IBE scheme and HIBE scheme was natural to server-aided. In their work, the sender chooses a random message M_1 and set the message $M_2 = M \oplus M_1$, and the server can get one of M_1 and M_2 while the recipient can recover the other one. Since the server may be untrusted, if he sends something that is different from what he obtained, then the recipient cannot gain the real message M. And the ciphertext contains a HIBE ciphertext and ℓ IBE ciphertexts in their construction, where ℓ is the length of identity. While in our construction, the server only can get a HIBE ciphertext on M, if he has dishonest behavior, then the recipient cannot decrypt to recover the message M and thus can detect this case. Thus, our construction can guarantee both the integrity and privacy of messages, and has shorter ciphertext size which contains ℓ IBE ciphertexts in our construction.

4 Conclusion

In this paper, we propose a generic construction of server-aided revocable hierarchical identity-based encryption. Specifically, we give a definition of SR-HIBE scheme by extending the SR-IBE scheme to the HIBE setting and propose a generic construction

of SR-HIBE scheme with DKER from any (weak) L-level RHIBE without DKER and $(L+1)$-level HIBE by modifying the definition of RHIBE scheme. In order to realize the server-aided revocable functionality, we use the "double encryption" technique. Besides, we obtain a generic construction of SR-IBE scheme with DKER from any IBE scheme and two-level HIBE scheme. Our construction has shorter ciphertext size and can guarantee both the integrity and privacy of messages.

References

1. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *Advances in Cryptology - EUROCRYPT 2010*, pages 553–572. Springer, 2010.
2. Alexandra Boldyreva, Vipul Goyal, and Virendra Kumar. Identity-based encryption with efficient revocation. In *Proceedings of the 2008 ACM Conference on Computer and Communications Security - CCS 2008*, pages 417–426, 2008.
3. Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT 2004*, pages 223–238. Springer, 2004.
4. Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *Advances in Cryptology - CRYPTO 2004*, pages 443–459. Springer, 2004.
5. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology - CRYPTO 2001*, pages 213–229. Springer, 2001.
6. Xavier Boyen and Qinyi Li. Towards tightly secure lattice short signature and id-based encryption. In *Advances in Cryptology - ASIACRYPT 2016*, pages 404–434. Springer, 2016.
7. Jie Chen, Hoon Wei Lim, San Ling, Huaxiong Wang, and Khoa Nguyen. Revocable identity-based encryption from lattices. In *Information Security and Privacy - ACISP 2012*, pages 390–403. Springer, 2012.
8. Hui Cui, Robert H. Deng, Yingjiu Li, and Baodong Qin. Server-aided revocable attribute-based encryption. In *Computer Security - ESORICS 2016*, pages 570–587. Springer, 2016.
9. Nico Döttling and Sanjam Garg. Identity-based encryption from the diffie-hellman assumption. In *Advances in Cryptology - CRYPTO 2017*, pages 537–569. Springer, 2017.
10. Keita Emura, Jae Hong Seo, and Taek-Young Youn. Semi-generic transformation of revocable hierarchical identity-based encryption and its DBDH instantiation. *IEICE Transactions*, 99-A(1):83–91, 2016.
11. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing - STOC 2008*, pages 197–206, 2008.
12. Ziyuan Hu, Shengli Liu, Kefei Chen, and Joseph K. Liu. Revocable identity-based encryption and server-aided revocable IBE from the computational diffie-hellman assumption. *Cryptography*, 2(4):33, 2018.
13. Ying-Hao Hung, Yuh-Min Tseng, and Sen-Shan Huang. Revocable id-based signature with short size over lattices. *Security and Communication Networks*, 2017:7571201:1–7571201:9, 2017.
14. Yuu Ishida, Junji Shikata, and Yohei Watanabe. Cca-secure revocable identity-based encryption schemes with decryption key exposure resistance. *IJACT*, 3(3):288–311, 2017.
15. Shuichi Katsumata, Takahiro Matsuda, and Atsushi Takayasu. Lattice-based revocable (hierarchical) IBE with decryption key exposure resistance. In *Public-Key Cryptography - PKC 2019*, pages 441–471, 2019.

16. Kwangsu Lee. Revocable hierarchical identity-based encryption with adaptive security. *IACR Cryptology ePrint Archive*, 2016:749, 2016.
17. Kwangsu Lee, Intae Kim, and Seong Oun Hwang. Privacy preserving revocable predicate encryption revisited. *Security and Communication Networks*, 8(3):471–485, 2015.
18. Kwangsu Lee, Dong Hoon Lee, and Jong Hwan Park. Efficient revocable identity-based encryption via subset difference methods. *Des. Codes Cryptogr.*, 85(1):39–76, 2017.
19. Kwangsu Lee and Seunghwan Park. Revocable hierarchical identity-based encryption with shorter private keys and update keys. *Des. Codes Cryptogr.*, 86(10):2407–2440, 2018.
20. Huang Lin, Zhenfu Cao, Yuguang Fang, Muxin Zhou, and Haojin Zhu. How to design space efficient revocable IBE from non-monotonic ABE. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2011*, pages 381–385, 2011.
21. San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Lattice-based group signatures: Achieving full dynamicity (and deniability) with ease. *CoRR*, abs/1801.08737, 2018.
22. Zhenhua Liu, Xiangsong Zhang, Yupu Hu, and Tsuyoshi Takagi. Revocable and strongly unforgeable identity-based signature scheme in the standard model. *Security and Communication Networks*, 9(14):2422–2433, 2016.
23. Xuecheng Ma and Dongdai Lin. A generic construction of revocable identity-based encryption. *IACR Cryptology ePrint Archive*, 2019:299, 2019.
24. Xianping Mao, Junzuo Lai, Kefei Chen, Jian Weng, and Qixiang Mei. Efficient revocable identity-based encryption from multilinear maps. *Security and Communication Networks*, 8(18):3511–3522, 2015.
25. Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology - CRYPTO 2001*, pages 41–62. Springer, 2001.
26. Khoa Nguyen, Huaxiong Wang, and Juanyang Zhang. Server-aided revocable identity-based encryption from lattices. In *Cryptology and Network Security- CANS 2016*, pages 107–123, 2016.
27. Seunghwan Park, Kwangsu Lee, and Dong Hoon Lee. New constructions of revocable identity-based encryption from multilinear maps. *IEEE Trans. Information Forensics and Security*, 10(8):1564–1577, 2015.
28. Baodong Qin, Robert H. Deng, Yingjiu Li, and Shengli Liu. Server-aided revocable identity-based encryption. In *Computer Security - ESORICS 2015*, pages 286–304. Springer, 2015.
29. Baodong Qin, Qinglan Zhao, Dong Zheng, and Hui Cui. Server-aided revocable attribute-based encryption resilient to decryption key exposure. In *Cryptology and Network Security - CANS 2017*, pages 504–514. Springer, 2017.
30. Geumsook Ryu, Kwangsu Lee, Seunghwan Park, and Dong Hoon Lee. Unbounded hierarchical identity-based encryption with efficient revocation. In *Information Security Applications - WISA2015*, pages 122–133. Springer, 2015.
31. Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology - EUROCRYPT 2005*, pages 457–473. Springer, 2005.
32. Jae Hong Seo and Keita Emura. Efficient delegation of key generation and revocation functionalities in identity-based encryption. In *Topics in Cryptology - CT-RSA 2013*, pages 343–358, 2013.
33. Jae Hong Seo and Keita Emura. Revocable identity-based encryption revisited: Security model and construction. In *Public-Key Cryptography - PKC 2013*, pages 216–234. Springer, 2013.
34. Jae Hong Seo and Keita Emura. Revocable hierarchical identity-based encryption. *Theor. Comput. Sci.*, 542:44–62, 2014.

35. Jae Hong Seo and Keita Emura. Revocable hierarchical identity-based encryption: History-free update, security against insiders, and short ciphertexts. In *Topics in Cryptology - CT-RSA 2015*, pages 106–123, 2015.
36. Jae Hong Seo and Keita Emura. Revocable hierarchical identity-based encryption via history-free approach. *Theor. Comput. Sci.*, 615:45–60, 2016.
37. Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology - CRYPTO 1984*, pages 47–53. Springer, 1984.
38. Atsushi Takayasu and Yohei Watanabe. Lattice-based revocable identity-based encryption with bounded decryption key exposure resistance. In *Information Security and Privacy - ACISP 2017*, pages 184–204. Springer, 2017.
39. Tung-Tso Tsai, Yuh-Min Tseng, and Sen-Shan Huang. Efficient revocable certificateless public key encryption with a delegated revocation authority. *Security and Communication Networks*, 8(18):3713–3725, 2015.
40. Shengbao Wang, Zhenfu Cao, Qi Xie, and Wenhao Liu. Practical identity-based encryption in multiple private key generator (PKG) environments. *Security and Communication Networks*, 8(1):43–50, 2015.
41. Yohei Watanabe, Keita Emura, and Jae Hong Seo. New revocable IBE in prime-order groups: Adaptively secure, decryption key exposure resistant, and with short public parameters. In *Topics in Cryptology - CT-RSA 2017*, pages 432–449. Springer, 2017.
42. Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT 2005*, pages 114–127. Springer, 2005.
43. Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *Advances in Cryptology - CRYPTO 2009*, pages 619–636. Springer, 2009.
44. Shota Yamada. Asymptotically compact adaptively secure lattice ibes and verifiable random functions via generalized partitioning techniques. In *Advances in Cryptology - CRYPTO 2017*, pages 161–193. Springer, 2017.