cryptography

MDPI

Article

Revocable Identity-Based Encryption and Server-Aided Revocable IBE from the Computational Diffie-Hellman Assumption †

Ziyuan Hu [1], Shengli Liu [1,3,*], Kefei Chen [2,3] and Joseph K. Liu [4]

1 Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China; huziyuan1989@sjtu.edu.cn
2 Department of Mathematics, Hangzhou Normal University, Hangzhou 310036, China; kfchen@hznu.edu.cn
3 Westone Cryptologic Research Center, Beijing 100070, China
4 Faculty of Information Technology, Monash University, Clayton VIC 3800, Australia; joseph.liu@monash.edu
* Correspondence: slliu@sjtu.edu.cn
† Part of this work was published in ACISP 2018. This is the full version.

Received: 30 August 2018; Accepted: 18 October 2018; Published: 23 October 2018

Abstract: An Identity-based encryption (IBE) simplifies key management by taking users' identities as public keys. However, how to dynamically revoke users in an IBE scheme is not a trivial problem. To solve this problem, IBE scheme with revocation (namely revocable IBE scheme) has been proposed. Apart from those lattice-based IBE, most of the existing schemes are based on decisional assumptions over pairing-groups. In this paper, we propose a revocable IBE scheme based on a weaker assumption, namely Computational Diffie-Hellman (CDH) assumption over non-pairing groups. Our revocable IBE scheme is inspired by the IBE scheme proposed by Döttling and Garg in Crypto2017. Like Döttling and Garg's IBE scheme, the key authority maintains a complete binary tree where every user is assigned to a leaf node. To adapt such an IBE scheme to a revocable IBE, we update the nodes along the paths of the revoked users in each time slot. Upon this updating, all revoked users are forced to be equipped with new encryption keys but without decryption keys, thus they are unable to perform decryption any more. We prove that our revocable IBE is adaptive IND-ID-CPA secure in the standard model. Our scheme serves as the first revocable IBE scheme from the CDH assumption. Moreover, we extend our scheme to support Decryption Key Exposure Resistance (DKER) and also propose a server-aided revocable IBE to decrease the decryption workload of the receiver. In our schemes, the size of updating key in each time slot is only related to the number of newly revoked users in the past time slot.

Keywords: revocable identity-based encryption; server-aided revocable identity-based encryption; CDH assumption

1. Introduction

The concept of Identity-Based Encryption (IBE) was proposed by Shamir [1] in 1984. In an IBE scheme, the public key of a user can simply be the identity id of the user, like name, email address, etc. An IBE scheme considers three parties: key authority, sender and receiver. The key authority is in charge of generating secret key $\mathsf{sk}_{\mathsf{id}}$ for user id. A sender simply encrypts plaintexts under the receiver's identity id and the receiver uses his own secret key $\mathsf{sk}_{\mathsf{id}}$ for decryption. With IBE, there is no need for senders to ask for authenticated public keys from Public-Key Infrastructures, hence key management is greatly simplified.

Over the years, there have been many IBE schemes proposed from various assumptions in the standard model. Most of the assumptions are decisional ones, like the bilinear Diffie-Hellman

assumption [2–4] over pairing-groups, or the decisional learning-with-errors (LWE) assumption from lattices [5–7]. Most recently, a breakthrough work was done by Döttling and Garg [8], who proposed the first IBE scheme based solely on the Computational Diffie-Hellman (CDH) assumption over groups free of pairings.

Though IBE enjoys the advantage of easy key management, how to revoke users in an IBE system is a non-trivial problem. It was Boneh and Franklin [9] who first proposed revocable IBE (RIBE) to solve the problem. Later, Boldyreva et al. [10] formalized the definition of selective-ID security and constructed a more efficient RIBE scheme based on a fuzzy IBE scheme [11]. Then Libert and Vergnaud proposed the first adaptive-ID secure revocable IBE scheme [12]. In [13], Seo and Emura strengthened the security model by introducing an additional important security notion, called Decryption Key Exposure Resistance (DKER). They also constructed a revocable IBE scheme in the strengthened model, and the security of this scheme is from the Decisional Bilinear Diffie-Hellman (DBDH) assumption. Since then, most of the revocable IBE schemes constructed from pairing groups achieved DKER. For example, in the strengthened security model, Lee et al. [14] constructed a revocable IBE scheme via subset difference methods to reduce the size of key updating based on the DBDH assumption, and Watanabe et al. [15] introduced a new revocable IBE with short public parameters based on both the Decisional Diffie-Hellman (DDH) assumption and the Augmented Decisional Diffie-Hellman (ADDH) assumption over pairing-friendly group. Furthermore, Park et al. [16] constructed a revocable IBE whose key update cost is only $O(1)$, but the scheme relied on multilinear maps. Without pairing, it seems difficult to achieve DKER. In [17], Chen et al. proposed the first selective-ID secure revocable IBE scheme from the LWE assumption over lattices in the traditional security model (without DKER). Later, Takayasu and Watanabe [18] designed a lattice-based revocable IBE with bounded DKER. Meanwhile, to improve the decryption efficiency for the receiver, Qin et al. [19] provided a new model named Server-Aided Revocable Identity-Based Encryption (SR-IBE) which used a server as an intermediary to help the receiver to decrypt part of the ciphertext. In fact, the revocable property is so important that it is studied not only in IBE but also in Identity-Based Proxy Re-encryption [20], Fine-Grained Encryption of Cloud Data [21,22] and Attribute-Based Encryption [23]. It should be noted that bilinear pairings are essential techniques in these schemes [20–23].

Please Please note that all the existing RIBE schemes are based on assumptions over pairing-friendly groups or the LWE assumption over lattices. On the other hand, Döttling and Garg's IBE scheme [8] is based on the CDH assumption over non-pairing group, but it does not consider user revocation. In this paper, we aim to fill the gap by designing RIBE from the CDH assumption without use of pairings.

1.1. Our Contributions

In this paper, we propose the first revocable IBE (RIBE) schemes and server-aided revocable IBE (SR-IBE) based on the Computational Diffie-Hellman (CDH) assumption over groups free of pairings. The corner stone of this scheme is the IBE scheme proposed by Döttling and Garg [8]. Our RIBE schemes enjoy the following features.

1. **Weaker security assumption.** The securities of our RIBE and SR-IBE schemes can be reduced to the CDH assumption. Hence our schemes serve as the first RIBE/SR-IBE schemes from the CDH assumption over non-pairing groups. More precisely, our first RIBE scheme can achieve adaptive-IND-ID-CPA security but without the property of decryption key exposure resistance(DKER). Our second RIBE scheme obtains decryption key exposure resistance but with selective-IND-ID-CPA security. Our SR-IBE scheme is selective-SR-ID-CPA secure. The securities of the three schemes can be reduced to the CDH assumption.
2. **Smaller size of key updating.** When a time slot begins, the key updating algorithm of our RIBE/SR-IBE will issue updating keys whose size is only linear to the number of newly revoked users in the past time slot. In comparison, most of the existing RIBE/SR-IBE schemes have to

update keys whose number is related to the number of all revoked users across all the previous time slots.

In Table 1, we compare our RIBE scheme with some existing RIBE schemes.

Table 1. Comparison with RIBE schemes (in the standard model). Here N is the total number of users, r is the number of all revoked users and Δr is the number of newly revoked users the past time slot. DKER means decryption key exposure resistance.

IBE	Security Assumption	Pairing Free	Security Model	Key Updating Size	DKER
[17]	LWE	✓	Selective-IND-ID-CPA	$O(r \log(N/r))$	×
[18]	LWE	✓	Selective-IND-ID-CPA	$O(r \log(N/r))$	Bounded
[10]	DBDH	×	Selective-IND-ID-CPA	$O(r \log(N/r))$	×
[12]	DBDH	×	Adaptive-IND-ID-CPA	$O(r \log(N/r))$	×
[13]	DBDH	×	Adaptive-IND-ID-CPA	$O(r \log(N/r))$	✓
[14]	DBDH	×	Adaptive-IND-ID-CPA	$O(r)$	✓
[15]	DDH and ADDH	×	Adaptive-IND-ID-CPA	$O(r \log(N/r))$	✓
[16]	Multilinear	×	Selective-IND-ID-CPA	$O(1)$	✓
Our RIBE 1	CDH	✓	Adaptive-IND-ID-CPA	$O(\Delta r(\log N - \log(\Delta r)))$	×
Our RIBE 2	CDH	✓	Selective-IND-ID-CPA	$O(\Delta r(\log N - \log(\Delta r)))$	✓
Our SR-IBE	CDH	✓	Selective-IND-ID-CPA	$O(\Delta r(\log N - \log(\Delta r)))$	✓

Remark 1. *Döttling and Garg's IBE makes use of garbled circuits to implement the underlying cryptographic primitives. Hence it is prohibitive in terms of efficiency. Our RIBE inherits their idea, hence the efficiency of our RIBE scheme is also incomparable to the RIBE schemes from bilinear maps. However, since no RIBE scheme is available from the CDH assumption over non-pairing groups, our scheme serves as a theoretical exploration in the field of RIBE.*

1.2. Paper Organization

In Section 2, we collect notations and some basic definitions used in the paper and present the framework. We illustrate our idea of RIBE in Section 3. In Section 4, we construct a revocable IBE scheme (without DKER) based on the CDH assumption and present the correctness and security analysis of the scheme. Then we show how to make our RIBE to obtain DKER in Section 5. In Section 6, we provide a SR-IBE scheme from the CDH assumption. In Section 7, we illustrate the key updating complexity analysis of our scheme.

2. Preliminaries

2.1. Notations

The security parameter is denoted by λ. "probabilistic polynomial-time" is abbreviated by "PPT". Let n, a and b be integers. Denote by $[n]$ the set $\{1, \cdots, n\}$, by $[a, b]$ the set $\{a, a+1, \cdots, b\}$, by $\{0,1\}^*$ the set of bit-strings of arbitrary length, and by $\{0,1\}^{\leq \ell}$ the set of bit-strings of length at most ℓ. Let ε be an empty string. and $|v|$ be the bit-length of string v. Obviously, $|\varepsilon| = 0$. Denote by $x||y$ the concatenation of two bit-strings x and y, by x_i the i-th bit of x, by $x \xleftarrow{\$} \mathcal{S}$ the process of sampling the element x from the set $\mathcal{S}$ uniformly at random, and by $a \leftarrow \mathcal{X}$ the process of sampling the element a over the distribution $\mathcal{X}$. By $a \leftarrow f(\cdot)$ we mean that a is the output of a function f. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ is negligible if for any polynomial $p(\lambda)$ it holds that $\mathsf{negl}(\lambda) < 1/p(\lambda)$ for all sufficiently large $\lambda \in \mathbb{N}$.

2.2. Pseudorandom Functions

Let PRF: $\mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be an efficiently computable function. For an adversary $\mathcal{A}$, define its advantage function as

$$\mathrm{Adv}^{\mathrm{PRF}}_{\mathcal{A}}(1^\lambda) := |\Pr[b = 1 \mid k \xleftarrow{\$} \mathcal{K}; b \leftarrow \mathcal{A}^{\mathsf{PRF}(k,\cdot)}] - \Pr[b = 1 \mid b \leftarrow \mathcal{A}^{\mathsf{RF}(\cdot)}]|,$$

where $\mathsf{RF} : \mathcal{X} \to \mathcal{Y}$ is a truly random function. PRF is a pseudorandom function (PRF) if the above advantage function $\mathrm{Adv}^{\mathrm{PRF}}_{\mathcal{A}}(1^\lambda)$ is negligible for any PPT $\mathcal{A}$.

2.3. Revocable Identity-Based Encryption

A revocable IBE (RIBE) consists of seven PPT algorithms $\mathsf{RIBE} = (\mathsf{RIBE.Setup}, \mathsf{RIBE.KG}, \mathsf{RIBE.KU}, \mathsf{RIBE.KU}, \mathsf{RIBE.Enc}, \mathsf{RIBE.Enc}, \mathsf{RIBE.R})$. Let $\mathcal{M}$ denote the message space, $\mathcal{ID}$ the identity space and $\mathcal{T}$ the space of time slots.

- **Setup**: The setup algorithm RIBE.Setup is run by the key authority. The input of the algorithm is a security parameter λ and n, where the maximal number of users is 2^n. The output of this algorithm consists of a pair of key $(\mathsf{mpk}, \mathsf{msk})$, an initial state st = (KL, PL, RL,KU), where KL is the key list, PL is the list of public information, RL is the list of revoked users and KU is the update key list. In formula, $(\mathsf{mpk}, \mathsf{msk}, \mathsf{st}) \leftarrow \mathsf{RIBE.Setup}(1^\lambda, 1^n)$.
- **Private Key Generation**: This algorithm RIBE.KG is run by the key authority which takes as input the key pair $(\mathsf{mpk}, \mathsf{msk})$, an identity id and the state st. The output of this algorithm is a private key $\mathsf{sk}_{\mathsf{id}}$ and an updated state st'. In formula, $(\mathsf{sk}_{\mathsf{id}}, \mathsf{st}') \leftarrow \mathsf{RIBE.KG}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}, \mathsf{st})$.
- **Key Update Generation**: This algorithm RIBE.KU is run by the authority. Given the key pair $(\mathsf{mpk}, \mathsf{msk})$, an update time t, and a state st, this algorithm updates the update key list KU and the the list of public information PL. In formula, $\mathsf{st}' \leftarrow \mathsf{RIBE.KU}(\mathsf{mpk}, \mathsf{msk}, \mathsf{t}, \mathsf{st})$.
- **Decryption key generation**: This algorithm RIBE.DK is run by the receiver. Given the master public key mpk, a private key $\mathsf{sk}_{\mathsf{id}}$, the update key list KU and the time slot t, this algorithm outputs a decryption key $\mathsf{sk}^{(\mathsf{t})}_{\mathsf{id}}$ for time slot t. In formula, $\mathsf{sk}^{(\mathsf{t})}_{\mathsf{id}} \leftarrow \mathsf{RIBE.DK}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{id}}, \mathsf{KU}, \mathsf{t})$.
- **Encryption**: This algorithm RIBE.Enc is run by the sender. Given the public key mpk, a public list PL, an identity id, a time slot t and a message m, this algorithm outputs a ciphertext ct. In formula, $\mathsf{ct} \leftarrow \mathsf{RIBE.Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{t}, m, \mathsf{PL})$.
- **Decryption**: This algorithm RIBE.Enc is run by the receiver. The algorithm takes as input the master public key mpk, the decryption key $\mathsf{sk}^{(\mathsf{t})}_{\mathsf{id}}$ and the ciphertext ct, and outputs a message m or a failure symbol $\perp$. In formula, $m/\perp \leftarrow \mathsf{RIBE.Dec}(\mathsf{mpk}, \mathsf{sk}^{(\mathsf{t})}_{\mathsf{id}}, \mathsf{ct})$.
- **Revocation**: This algorithm RIBE.R is run by the key authority. Given a revoked identity id and the time slot t during which id is revoked and a state $\mathsf{st} = (\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU})$, this algorithm updates the revocation list RL with $\mathsf{RL} \leftarrow \mathsf{RL} \cup \{(\mathsf{id}, \mathsf{t})\}$. It outputs a new state $\mathsf{st}' = (\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU})$.

Correctness. For all $(\mathsf{mpk}, \mathsf{msk}, \mathsf{st}) \leftarrow \mathsf{RIBE.Setup}(1^\lambda, N)$, all $m \in \mathcal{M}$, all identity $\mathsf{id} \in \mathcal{ID}$, all time slot $\mathsf{t} \in \mathcal{T}$, and revocation list RL, for all $(\mathsf{sk}_{\mathsf{id}}, \mathsf{st}') \leftarrow \mathsf{RIBE.KG}(\mathsf{msk}, \mathsf{id}, \mathsf{st})$, $\mathsf{st}'' \leftarrow \mathsf{RIBE.KU}(\mathsf{msk}, \mathsf{t}, \mathsf{st})$, and $\mathsf{sk}^{(\mathsf{t})}_{\mathsf{id}} \leftarrow \mathsf{RIBE.DK}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{id}}, \mathsf{KU}, \mathsf{t})$, we have $\mathsf{RIBE.Dec}(\mathsf{mpk}, \mathsf{sk}^{(\mathsf{t})}_{\mathsf{id}}, \mathsf{RIBE.Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{t}, m, \mathsf{PL})) = m$ if $(\mathsf{id}, \mathsf{t}) \notin \mathsf{RL}$(i.e., id is not revoked at time t) and $\mathsf{PL} \in \mathsf{st}''$.

Now we explain how a revocable IBE system works. To setup the system, the key authority invokes RIBE.Setup to generate master public key mpk, master secret key msk and the state st. Then it publishes the public key mpk. When a user registers in the system with identity id, the key authority invokes $\mathsf{RIBE.KG}(\mathsf{msk}, \mathsf{id}, \mathsf{st})$ to generate the private key $\mathsf{sk}_{\mathsf{id}}$ for user id. If a user id needs to be revoked during time slot t, the key authority invokes $\mathsf{RIBE.R}(\mathsf{id}, \mathsf{t}, \mathsf{st})$. Next it updates the state st. At the beginning of each time slot t, the key authority might invoke $\mathsf{RIBE.KU}(\mathsf{msk}, \mathsf{t}, \mathsf{st})$ to update keys by updating set KU. Then it publishes some information about the updated set KU. Meanwhile it may also publish some public information PL. During time slot t, when a user wants to send a message m to another user id, he/she invokes $\mathsf{RIBE.Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{t}, m, \mathsf{PL})$ to encrypt m to obtain the ciphertext ct, then sends $(\mathsf{t}, \mathsf{ct})$ to user id. To decrypt a ciphertext ct encrypted at time t, the receiver id first invokes $\mathsf{RIBE.DK}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{id}}, \mathsf{KU}, \mathsf{t})$ to generate its own decryption key $\mathsf{sk}^{(\mathsf{t})}_{\mathsf{id}}$ of time t. The receiver id invokes $\mathsf{RIBE.Dec}(\mathsf{mpk}, \mathsf{sk}^{(\mathsf{t})}_{\mathsf{id}}, \mathsf{ct})$ to decrypt the ciphertext and recover the plaintext.

Remark. In the definition of our RIBE, KL is the key list which stores the essential information used to generate the update key. PL is a public information list which is used in the encryption algorithm. In the traditional definition of RIBE in other works, no PL is defined. However, in our construction, PL will serve as an essential input to the encryption algorithm and that is the reason we define it. Nevertheless, our definition can be regarded as a general one, while the traditional definition of RIBE can be seen as a special case of $\mathsf{PL} = \emptyset$.

Security. Now we formalize the security of a revocable IBE. We first consider four oracles: private key generation oracle $\mathrm{KG}(\cdot)$, key update oracle KU, decryption key generation oracle $\mathrm{DK}(\cdot,\cdot)$ and revocation oracle $\mathrm{RVK}(\cdot,\cdot)$ which are shown in Table 2. The security of adaptive-IND-ID-CPA defines as follows.

Table 2. Three oracles that the adversary can query.

$\underline{\mathrm{KG}(\mathsf{id}):}$	$\underline{\mathrm{KU}:}$
$(\mathsf{sk}_{\mathsf{id}}, \mathsf{st}') \leftarrow \mathsf{RIBE.KG}(\mathsf{msk}, \mathsf{id}, \mathsf{st})$	$\mathsf{st}' \leftarrow \mathsf{RIBE.KU}(\mathsf{msk}, \mathsf{t}, \mathsf{st})$ $\mathsf{st} := \mathsf{st}'.$ Parse $\mathsf{st} = (\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU})$
Output $\mathsf{sk}_{\mathsf{id}}$.	Output $(\mathsf{KU}, \mathsf{PL})$.
$\underline{\mathrm{RVK}(\mathsf{id}, \mathsf{t}):}$	$\underline{\mathrm{DK}(\mathsf{id}, \mathsf{t}):}$
$\mathsf{st}' \leftarrow \mathsf{RIBE.R}(\mathsf{id}, \mathsf{t}, \mathsf{st})$	$(\mathsf{sk}_{\mathsf{id}}, \mathsf{st}') \leftarrow \mathsf{RIBE.KG}(\mathsf{msk}, \mathsf{id}, \mathsf{st})$
$\mathsf{st}' := (\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU})$	$\mathsf{sk}_{\mathsf{id}}^{(\mathsf{t})} \leftarrow \mathsf{RIBE.DK}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{id}}, \mathsf{KU}, \mathsf{t})$
Output RL.	Output $\mathsf{sk}_{\mathsf{id}}^{(\mathsf{t})}$.

Definition 1. *Let RIBE = (RIBE.Setup, RIBE.KG, RIBE.KU, RIBE.DK, RIBE.Enc, RIBE.Dec, RIBE.R) be a revocable IBE scheme. Below describes an experiment played between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$.*

$$
\begin{aligned}
&\boldsymbol{EXP}_{\mathcal{A}}^{\textit{adaptive-IND-ID-CPA}}(\lambda):\\
&\quad (mpk, msk, st) \leftarrow \mathit{RIBE.Setup}(1^{\lambda}, 1^{n});\\
&\quad \mathit{Parse}\ st = (KL, PL, RL, KU);\\
&\quad (M_0, M_1, id^*, t^*, \overline{st_{\mathcal{A}}}) \leftarrow \mathcal{A}^{\mathrm{KG}(\cdot),\mathrm{KU},\mathrm{RVK}(\cdot,\cdot)}(mpk);\\
&\quad \theta \xleftarrow{\$} \{0,1\};\\
&\quad ct^* \leftarrow \mathit{RIBE.Enc}(mpk, id^*, t^*, M_{\theta}, PL)\\
&\quad \theta' \leftarrow \mathcal{A}^{\mathrm{KG}(\cdot),\mathrm{KU},\mathrm{DK}(\cdot,\cdot),\mathrm{RVK}(\cdot,\cdot)}(ct^*, \overline{st_{\mathcal{A}}})\\
&\quad \mathit{If}\ \theta = \theta' \mathit{Return}\ 1; \mathit{If}\ \theta \neq \theta' \mathit{Return}\ 0.
\end{aligned}
$$

The experiment has the following requirements for $\mathcal{A}$.

- *The two plaintexts submitted by $\mathcal{A}$ have the same length, i.e., $|M_0| = |M_1|$.*
- *The time slot t submitted to* KU *and* $\mathrm{RVK}(\cdot,\cdot)$ *by $\mathcal{A}$ is in ascending order.*
- *If the challenger has published* KU *at time t, then it is not allowed to query oracle $\mathrm{RVK}(\cdot, \mathsf{t}')$ with $\mathsf{t}' < \mathsf{t}$.*
- *If $\mathcal{A}$ has queried id^* to oracle $\mathrm{KG}(\cdot)$, then there must be query (id^*, t) to oracle $\mathrm{RVK}(\cdot)$ satisfies $\mathsf{t} < t^*$, i.e., id^* must has been revoked before time t^*.*
- *If id^* is not revoked at time t^*, $\mathrm{DK}(\cdot,\cdot)$ cannot be queried on (id^*, t^*).*

A revocable IBE scheme is adaptive-IND-ID-CPA secure (with DKER) if for all PPT adversary $\mathcal{A}$, the following advantage is negligible in the security parameter λ, i.e.,

$$\mathbf{Adv}_{RIBE,\mathcal{A}}^{\textit{adaptive-IND-ID-CPA}}(\lambda) = |\Pr[\boldsymbol{EXP}_{\mathcal{A}}^{\textit{adaptive-IND-ID-CPA}}(\lambda) = 1] - 1/2| = \mathit{negl}(\lambda).$$

Remark 2. *The security definition without DKER is similarly defined with changing the experiment so that an adversary $\mathcal{A}$ is not allowed to make any decryption key reveal query, i.e., $\mathcal{A}$ cannot query for the oracle* $\mathrm{DK}(\cdot,\cdot)$.

Next we define selective-IND-ID-CPA security for RIBE, where the adversary has to determine the target identity id^*, target time slot t^* at the beginning of the experiment. Clearly, selective-IND-ID-CPA security is weaker than adaptive-IND-ID-CPA security.

Definition 2. *Let $RIBE = (RIBE.Setup, RIBE.KG, RIBE.KU, RIBE.DK, RIBE.Enc, RIBE.Dec, RIBE.R)$ be a revocable IBE scheme. Below describes an experiment played between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$.*

$\boldsymbol{EXP}_{\mathcal{A}}^{selective\text{-}IND\text{-}ID\text{-}CPA}(\lambda)$:
$(id^*, t^*) \leftarrow \mathcal{A}$
$(mpk, msk, st) \leftarrow RIBE.Setup(1^\lambda, 1^n)$;
$Parse\ st = (KL, PL, RL, KU)$;
$(M_0, M_1, \overline{st_{\mathcal{A}}}) \leftarrow \mathcal{A}^{\mathrm{KG}(\cdot),\mathrm{KU},\mathrm{RVK}(\cdot,\cdot)}(mpk)$;
$\theta \xleftarrow{\$} \{0,1\}$;
$ct^* \leftarrow RIBE.Enc(mpk, id^*, t^*, M_\theta, PL)$
$\theta' \leftarrow \mathcal{A}^{\mathrm{KG}(\cdot),\mathrm{KU},\mathrm{DK}(\cdot,\cdot),\mathrm{RVK}(\cdot,\cdot)}(ct^*, \overline{st_{\mathcal{A}}})$
$If\ \theta = \theta' Return\ 1; If\ \theta \neq \theta' Return\ 0.$

The requirements for $\mathcal{A}$ in this experiment are the same as the requirements in $\boldsymbol{EXP}_{\mathcal{A}}^{adaptive\text{-}IND\text{-}ID\text{-}CPA}(\lambda)$. A revocable IBE scheme is selective-IND-ID-CPA secure (with DKER) if for all PPT adversary $\mathcal{A}$, the following advantage is negligible in the security parameter λ, i.e.,

$$\mathbf{Adv}_{RIBE,\mathcal{A}}^{selective\text{-}IND\text{-}ID\text{-}CPA}(\lambda) = |\Pr[\boldsymbol{EXP}_{\mathcal{A}}^{selective\text{-}IND\text{-}ID\text{-}CPA}(\lambda) = 1] - 1/2| = negl(\lambda).$$

Selective-IND-ID-CPA security without DKER is defined can be similarly defined by changing the experiment so that an adversary $\mathcal{A}$ is not allowed to query for the oracle $\mathrm{DK}(\cdot,\cdot)$.

2.4. Server-Aided Revocable Identity-Based Encryption

In a server-aided revocable identity-based encryption scheme [19], there are four parities and they work as follows (as shown in Figure 1):

- **Key Authority** generates a public key and a secret key for every registered user and issues the secret key to the user and the public key to the server. In each time slot, the key authority delivers a update key list (to revoke users) to the server.
- **Sender** encrypts a message for an identity and a time slot and sends the ciphertext to the server.
- **Sever** combines the update key list and the stored users' public keys to generate the transformation keys in every time slot for all users. When receiving a ciphertext, the server transforms it to a partially decrypted ciphertext using the transformation key corresponding to the receiver's identity and the corresponding time slot. Then it sends the partially decrypted ciphertext to the receiver.
- **Receiver** recovers the sender's message from the partially decrypted ciphertext using a decryption key which can be generated by his/her own secret key and the corresponding time slot.

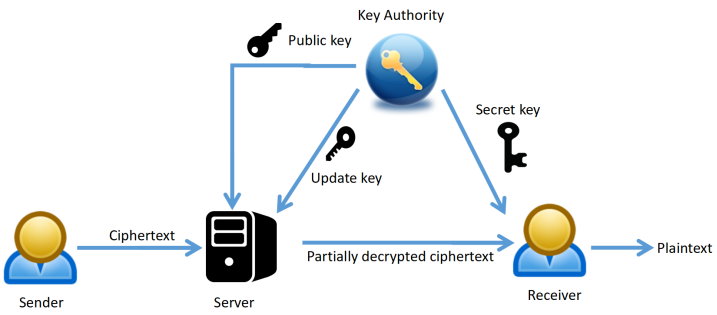

Figure 1. System model of a server-aided revocable IBE.

Now, we formally define Server-Aided Revocable Identity Based Encryption (SR-IBE) which was first proposed by Qin et al. [19]. A SR-IBE scheme consists of ten PPT algorithms $\Sigma = (\mathsf{Setup}, \mathsf{PubKG}, \mathsf{KU}, \mathsf{TranKG}, \mathsf{PrivKG}, \mathsf{DK}, \mathsf{Enc}, \mathsf{Transform}, \mathsf{Dec}, \mathsf{R})$. Let $\Sigma.\mathcal{M}$ denote the message space, $\Sigma.\mathcal{ID}$ the identity space and $\Sigma.\mathcal{T}$ the space of time slots.

- **Setup**: The setup algorithm Setup is run by the key authority. The input of the algorithm is a security parameter λ and a parameter n, which indicates that the maximal number of users is 2^n. The output of this algorithm consists of a pair of key $(\mathsf{msk}, \mathsf{mpk})$ and an initial state $\mathsf{st} = (\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU})$, where KL is the key list, PL is the list of public information, RL is the list of revoked users and KU is the update key list. In formula, $(\mathsf{msk}, \mathsf{mpk}, \mathsf{st}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$.
- **Public Key Generation**: The public key generation algorithm PubKG is run by the key authority. It takes as input a master secret key msk, an identity $\mathsf{id} \in \{0,1\}^n$ and a state st. The output of this algorithm is the public key $\mathsf{pk}_{\mathsf{id}}$ on identity id. In formula, $\mathsf{pk}_{\mathsf{id}} \leftarrow \mathsf{PubKG}(\mathsf{msk}, \mathsf{id}, \mathsf{st})$.
- **Key Update Generation**: The key update generation algorithm KU is run by the key authority. It takes as input a master secret key msk, an update time t and a state st. The output of this algorithm is an update key list $\mathsf{KU}^{(\mathsf{t})}$ and an updated state st′. In formula, $(\mathsf{KU}^{(\mathsf{t})}, \mathsf{st}') \leftarrow \mathsf{KU}(\mathsf{msk}, \mathsf{t}, \mathsf{st})$.
- **Transformation Key Generation**: The transformation key generation algorithm TranKG is run by the server. It takes as input the master public key mpk, the public key $\mathsf{pk}_{\mathsf{id}}$ and an update key list $\mathsf{KU}^{(\mathsf{t})}$. The output of this algorithm is the transformation key $\mathsf{tk}_{\mathsf{id}}^{(\mathsf{t})}$. In formula, $\mathsf{tk}_{\mathsf{id}}^{(\mathsf{t})} \leftarrow \mathsf{TranKG}(\mathsf{mpk}, \mathsf{pk}_{\mathsf{id}}, \mathsf{KU}^{(\mathsf{t})})$.
- **Private Key Generation**: The private key generation algorithm PrivKG is run by the key authority. It takes the master secret key msk and an identity $\mathsf{id} \in \{0,1\}^n$ as input. The output of this algorithm is the private key $\mathsf{sk}_{\mathsf{id}}$ on identity id. In formula, $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{PrivKG}(\mathsf{msk}, \mathsf{id})$.
- **Decryption Key Generation**: The decryption key generation algorithm DK is run by the receiver. It takes the secret key $\mathsf{sk}_{\mathsf{id}}$ and a slot t as input. The output of this algorithm is the decryption key $\mathsf{Dk}_{\mathsf{id}}^{(\mathsf{t})}$. In formula, $\mathsf{Dk}_{\mathsf{id}}^{(\mathsf{t})} \leftarrow \mathsf{DK}(\mathsf{sk}_{\mathsf{id}}, \mathsf{t})$.
- **Encryption**: The encryption algorithm Enc is run by the sender. It takes the master public key mpk, an identity id, a time plot t, a plaintext message m and a public list PL as the input. The output of this algorithm is the ciphertext ct. In formula, $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{t}, m, \mathsf{PL})$.
- **Transformation**: The transformation algorithm transform is run by the server. It takes the master public key mpk, the transformation key $\mathsf{tk}_{\mathsf{id}}^{(\mathsf{t})}$ and the ciphertext ct as the input. The output of this algorithm is the partially decrypted ciphertext ct′. In formula, $\mathsf{ct}' \leftarrow \mathsf{Transform}(\mathsf{mpk}, \mathsf{tk}_{\mathsf{id}}^{(\mathsf{t})}, \mathsf{ct})$.
- **Decryption**: The decryption algorithm Dec is run by the receiver. The input of this algorithm consists of the master public key mpk, the decryption key $\mathsf{Dk}_{\mathsf{id}}^{(\mathsf{t})}$ and the partially decrypted ciphertext ct′. The output of this algorithm is the plaintext m. In formula, $\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{mpk}, \mathsf{Dk}_{\mathsf{id}}^{(\mathsf{t})}, \mathsf{ct}')$.
- **Revocation**: The revocation algorithm R is run by the key authority. The input of this algorithm consists of an identity id, a time plot t and a state st. The output of this algorithm is the updated state st′. In formula, $\mathsf{st}' \leftarrow \mathsf{R}(\mathsf{id}, \mathsf{t}, \mathsf{st})$.

Correctness. The correctness requires that for all message m, if the receiver is not revoked at time period t and all parties follow the algorithms above, then we have $\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{mpk}, \mathsf{Dk}_{\mathsf{id}}^{(\mathsf{t})}, \mathsf{Transform}(\mathsf{mpk}, \mathsf{tk}_{\mathsf{id}}^{(\mathsf{t})}, \mathsf{ct}))$.

Security. Now we formalize the security of SR-IBE. We first consider five oracles: public key generation oracle $\textsc{PubKG}(\cdot)$, key update oracle KU, private key generation oracle $\textsc{PrivKG}(\cdot)$, decryption key generation oracle $\mathrm{DK}(\cdot,\cdot)$ and revocation oracle $\textsc{Rvk}(\cdot,\cdot)$ which are shown in Table 3. The selective-SR-ID-CPA security is defined as follows.

Table 3. Five oracles that the adversary of a SR-IBE scheme can query.

$\underline{\text{PUBKG}(\mathsf{id}):}$ $\mathsf{pk}_{\mathsf{id}} \leftarrow \mathsf{PubKG}(\mathsf{msk}, \mathsf{id}, \mathsf{st})$ Output $\mathsf{pk}_{\mathsf{id}}$.	$\underline{\text{KU}:}$ $\mathsf{st}' \leftarrow \mathsf{KU}(\mathsf{msk}, \mathsf{t}, \mathsf{st})$ $\mathsf{st} := \mathsf{st}'$. Parse $\mathsf{st} = (\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU})$ Output $(\mathsf{KU}, \mathsf{PL})^{\cdot}$
$\underline{\text{PRIVKG}(\mathsf{id}):}$ $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{PrivKG}(\mathsf{msk}, \mathsf{id})$ Output $\mathsf{sk}_{\mathsf{id}}$.	
$\underline{\text{RVK}(\mathsf{id}, \mathsf{t}):}$ $\mathsf{st}' \leftarrow \mathsf{R}(\mathsf{id}, \mathsf{t}, \mathsf{st})$ $\mathsf{st}' := (\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU})$ Output RL.	$\underline{\text{DK}(\mathsf{id}, \mathsf{t}):}$ $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{PrivKG}(\mathsf{msk}, \mathsf{id})$ $\mathsf{Dk}_{\mathsf{id}}^{(\mathsf{t})} \leftarrow \mathsf{DK}(\mathsf{sk}_{\mathsf{id}}, \mathsf{t})$ Output $\mathsf{Dk}_{\mathsf{id}}^{(\mathsf{t})}$.

Definition 3. *Let* $\Sigma = (\mathsf{Setup}, \mathsf{PubKG}, \mathsf{KU}, \mathsf{TranKG}, \mathsf{PrivKG}, \mathsf{DK}, \mathsf{Enc}, \mathsf{Transform}, \mathsf{Dec}, \mathsf{R})$ *be a server-aided revocable IBE scheme. Below describes an experiment played between a challenger* $\mathcal{C}$ *and a PPT adversary* $\mathcal{A}$*.*

$\boldsymbol{EXP}_{\mathcal{A}}^{\textit{selective-SR-ID-CPA}}(\lambda)$:
$(id^*, t^*) \leftarrow \mathcal{A}$;
$(mpk, msk, st) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$;
$\mathsf{Parse}\ \mathsf{st} = (KL, PL, RL, KU)$;
$(m_0, m_1, st_{\mathcal{A}}) \leftarrow \mathcal{A}^{\text{PUBKG}(\cdot),\text{KU},\text{PRIVKG}(\cdot),\text{DK}(\cdot,\cdot),\text{RVK}(\cdot,\cdot)}(mpk)$;
$\theta \xleftarrow{\$} \{0,1\}$;
$ct^* \leftarrow \mathsf{Enc}(mpk, id^*, t^*, m, PL))$;
$\theta' \leftarrow \mathcal{A}^{\text{PUBKG}(\cdot),\text{KU},\text{PRIVKG}(\cdot),\text{DK}(\cdot,\cdot),\text{RVK}(\cdot,\cdot)}(mpk, ct^*, st_{\mathcal{A}})$;
If $\theta = \theta'$ *Return* 1; *If* $\theta \neq \theta'$ *Return* 0.

The experiment has the following requirements for $\mathcal{A}$*.*

- *The two plaintexts submitted by* $\mathcal{A}$ *have the same length, i.e.,* $|m_0| = |m_1|$*.*
- *The time slot* t *submitted to* KU *and* $\text{RVK}(\cdot,\cdot)$ *by* $\mathcal{A}$ *is in ascending order.*
- *If the challenger has published* KU *at time* t*, then it is not allowed to query oracle* $\text{RVK}(\cdot, \mathsf{t}')$ *with* $\mathsf{t}' < \mathsf{t}$*.*
- *If* $\mathcal{A}$ *has queried* id^* *to oracle* $\text{PRIVKG}(\cdot)$*, then there must exist a query* (id^*, t) *to oracle* $\text{RVK}(\cdot)$ *satisfying* $\mathsf{t} < t^*$*, i.e.,* id^* *must has been revoked before time* t^**.*
- *If* id^* *is not revoked at time* t^**,* $\text{DK}(\cdot,\cdot)$ *cannot be queried on* (id^*, t^*)*.*

A server-aided revocable IBE scheme is selective-SR-ID-CPA secure (with DKER) if for all PPT adversary $\mathcal{A}$*, the following advantage is negligible in the security parameter* λ*, i.e.,*

$$\mathbf{Adv}_{\textit{SR-IBE},\mathcal{A}}^{\textit{selective-SR-ID-CPA}}(\lambda) = |\Pr[\mathbf{EXP}_{\mathcal{A}}^{\textit{selective-SR-ID-CPA}}(\lambda) = 1] - 1/2| = \textit{negl}(\lambda).$$

2.5. Garbled Circuits

A garbled circuits scheme consists of two PPT algorithms $(\mathsf{GCircuit}, \mathsf{Eval})$.

- $\mathsf{GCircuit}(\lambda, \mathsf{C}) \rightarrow (\tilde{\mathsf{C}}, \{\mathsf{lab}_{w,b}\}_{w \in \mathsf{inp}(\mathsf{C}), b \in \{0,1\}})$: The algorithm GCircuit takes a security parameter λ and a circuit C as input. This algorithm outputs a garbled circuit $\tilde{\mathsf{C}}$ and labels $\{\mathsf{lab}_{w,b}\}_{w \in \mathsf{inp}(\mathsf{C}), b \in \{0,1\}}$ where each $\mathsf{lab}_{w,b} \in \{0,1\}^\lambda$. Here $\mathsf{inp}(\mathsf{C})$ represents the set $[\ell]$ where ℓ is the bit-length of the input of the circuit C.
- $\mathsf{Eval}(\tilde{\mathsf{C}}, \{\mathsf{lab}_{w,x_w}\}_{w \in \mathsf{inp}(\mathsf{C})}) \rightarrow y$: The algorithm Eval takes as input a garbled circuit $\tilde{\mathsf{C}}$ and a set of label $\{\mathsf{lab}_{w,x_w}\}_{w \in \mathsf{inp}(\mathsf{C})}$, and it outputs y.

Correctness. In a garbled circuit scheme, for any circuit C and an input $x \in \{0,1\}^\ell$, it holds that

$$\Pr[\mathsf{C}(x) = \mathsf{Eval}(\tilde{\mathsf{C}}, \{\mathsf{lab}_{w,x_w}\}_{w \in \mathsf{inp}(\mathsf{C})})] = 1$$

where $(\tilde{\mathsf{C}}, \{\mathsf{lab}_{w,b}\}_{w\in\mathsf{inp}(\mathsf{C}),b\in\{0,1\}}) \leftarrow \mathsf{GCircuit}(1^\lambda, \mathsf{C})$.

Security. In a garbled circuit scheme, the security means that there is a PPT simulator Sim such that for any C, x and for any PPT adversary $\mathcal{A}$, the following advantage of $\mathcal{A}$ is negligible in the security parameter λ:

$$\mathbf{Adv}^{GC}_{\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}(\tilde{\mathsf{C}}, \{\mathsf{lab}_{w,x_w}\}_{w\in\mathsf{inp}(\mathsf{C})}) = 1] - \Pr[\mathcal{A}(\mathsf{Sim}(1^\lambda, \mathsf{C}(\mathsf{x}))) = 1]| = \mathsf{negl}(\lambda),$$

where $(\tilde{\mathsf{C}}, \{\mathsf{lab}_{w,b}\}_{w\in\mathsf{inp}(\mathsf{C}),b\in\{0,1\}}) \leftarrow \mathsf{GCircuit}(1^\lambda, \mathsf{C})$.

2.6. Computational Diffie-Hellman Assumption

Let $(\mathbb{G}, g, p) \leftarrow \mathsf{GGen}(1^\lambda)$ be a group generation algorithm which outputs a cyclic group $\mathbb{G}$ of order p and a generator of $\mathbb{G}$.

Definition 4. [CDH Assumption] *The computational Diffie-Hellman (CDH) assumption holds w.r.t. GGen, if for any PPT algorithm $\mathcal{A}$ its advantage ϵ in solving computational Diffie-Hellman (CDH) assumption in $\mathbb{G}$ is negligible. In formula, Pr* $\left[\mathcal{A}(g, g^a, g^b) = g^{ab} \mid (\mathbb{G}, g, p) \leftarrow \mathsf{GGen}(1^\lambda); a, b \leftarrow \mathbb{Z}_p\right] = \mathsf{negl}(\lambda)$.

2.7. Chameleon Encryption

A chameleon encryption scheme has five PPT algorithms $\mathsf{CE} = (\mathsf{HGen}, \mathsf{H}, \mathsf{H}^{-1}, \mathsf{HEnc}, \mathsf{HDec})$.

- HGen: The algorithm HGen takes the security parameter λ and a message-length n as input. This algorithm outputs a key k and a trapdoor t.
- H: The algorithm H takes the key k, a message $x \in \{0,1\}^n$ and a randomness r as input. This algorithm outputs a hash value h and the length of h is λ.
- H^{-1}: The algorithm H^{-1} takes a trapdoor t, a previously used message $x \in \{0,1\}^n$, random coins r and a message $x' \in \{0,1\}^n$ as input. It outputs r'.
- HEnc: The algorithm HEnc takes a key k, a hash value h, an index $i \in [n]$, a bit $b \in \{0,1\}$, and a message $m \in \{0,1\}^*$ as input. It outputs a ciphertext ct.
- HDec: The algorithm HDec takes a key k, a message $x \in \{0,1\}^n$, a randomness r and a ciphertext ct as input. It outputs a value m or $\perp$.

The chameleon encryption scheme enjoys the following properties:

- **Uniformity.** For all $x, x' \in \{0,1\}^n$,if both r and r' are chosen uniformly at random, the two distribution $\mathsf{H}(k, x; r)$ and $\mathsf{H}(k, x'; r')$ are statistically indistinguishable.
- **Trapdoor Collisions.** For any $x, x' \in \{0,1\}^n$ and r, if $(k, t) \leftarrow \mathsf{HGen}(1^\lambda, n)$ and $r' \leftarrow \mathsf{H}^{-1}(t, (x, r), x')$, then it holds that $\mathsf{H}(k, x; r) = \mathsf{H}(k, x'; r')$. Moreover, if r is chosen uniformly and randomly, r' is statistically close to uniform.
- **Correctness.** For all $x \in \{0,1\}^n$, randomness r, index $i \in [n]$ and message m, if $(k, t) \leftarrow \mathsf{HGen}(1^\lambda, n)$, $h \leftarrow \mathsf{H}(k, x; r)$ and $ct \leftarrow \mathsf{HEnc}(k, (h, i, x_i), m)$, then $\mathsf{HDec}(k, ct, (x, r)) = m$
- **Security.** For a PPT adversary $\mathcal{A}$ against a chameleon encryption, consider the following experiment:

$\mathbf{EXP}^{\text{IND-CE}}_{\mathcal{A}}(\lambda)$:
$(k, t) \leftarrow \mathsf{HGen}(1^\lambda, n)$.
$(x, r, i, m_0, m_1) \leftarrow \mathcal{A}(k)$.
$b \xleftarrow{\$} \{0,1\}$.
$ct \leftarrow \mathsf{HEnc}(k, (\mathsf{H}(k, x; r), i, 1 - x_i), m_b)$.
$b' \leftarrow \mathcal{A}(k, ct, (x, r))$.
Output 1 if $b = b'$ and 0 otherwise.

The security of a chameleon encryption defines as follows: For any PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in experiment $\mathbf{EXP}^{\text{IND-CE}}_{\mathcal{A}}(\lambda)$ satisfies $|\Pr[\mathbf{Adv}^{\text{IND-CE}}_{\mathcal{A}}(\lambda) = 1] - 1/2| = \mathsf{negl}$.

In [8], such a chameleon encryption was constructed from the CDH assumption.

3. Idea of Our Revocable IBE Scheme

3.1. Idea of the DG Scheme

In the IBE scheme [8] proposed by Döttling and Garg, say the DG scheme, each id is an n-bit binary string. In other words, each user can be regarded as a leaf of a complete binary tree of depth n, which is the length of a user's identity id. For each level $j \in [n]$ in the tree, the key authority generates a pair of chameleon encryption key and trapdoor (k_j, td_j). As shown in Figure 2, a leaf v is attached with a key pair $(\mathsf{ek}_v, \mathsf{dk}_v)$, which is the public/secret key of an IND-CPA secure public-key encryption scheme PKE=(G, E, D), i.e., $(\mathsf{ek}_v, \mathsf{dk}_v) \leftarrow \mathsf{G}(1^\lambda)$. In addition, a non-leaf node v in the tree is attached with four values: the hash value h_v of this node, the hash value $h_{v||0}$ of the left child node, the hash value $h_{v||1}$ of the right child node, a randomness r such that $h_v = \mathsf{H}(k_{|v|}, h_{v||0}||h_{v||1}; r_v)$. epecially for $|v| = n-1$, $(h_{v||0}, h_{v||1}) := (\mathsf{ek}_{v||0}, \mathsf{ek}_{v||1})$. The master public key of IBE is given by the hash keys $(k_0, \ldots, k_{n-1})$ and the hash value h_ε of the root. The master secret key is the seed of a pseudorandom function to generate r_v and the trapdoors of the chameleon encryption.

Key Generation. Each user is assigned to a leaf in the tree according to id. The secret key is just all the values attached to those nodes on the path from the root to the leaf. For example, in Figure 2, if $\mathsf{id} = 010$, then the secret key is $\mathsf{sk}_{010} = (\{h_\varepsilon, h_0, h_1, r_\varepsilon\}, \{h_0, h_{00}, h_{01}, r_0\}, \{h_{01}, \mathsf{ek}_{010}, \mathsf{ek}_{011}, r_{01}\}, \mathsf{dk}_{010})$.

Encryption. As for encryption, two kinds of circuits are defined.

(1) $Q[m](\mathsf{ek})$ is a circuit with m hardwired and its input is ek. It computes and outputs the PKE ciphertext of message m under the public-key ek.

(2) $\mathsf{P}[\beta \in \{0,1\}, k, \overline{\mathsf{lab}}](h)$ is a circuit which hardwires bit β, key k and a serial of labels $\overline{\mathsf{lab}}$. It computes and outputs $\{\mathsf{HEnc}(k, (h, j+\beta\cdot\lambda, b), \mathsf{lab}_{j,b})\}_{j\in[\lambda], b\in\{0,1\}}$, where $\overline{\mathsf{lab}}$ is the short for $\{\mathsf{lab}_{j,b}\}_{j\in[\lambda],b\in\{0,1\}}$.

To encrypt a message m under id, the sender generates a series of garbled circuits from the bottom to the top. Specifically, for level n, it generates $\tilde{Q}$, the garbled circuit of $Q[m]$, and the corresponding label $\overline{\mathsf{lab}}$, i.e., $(\tilde{Q}, \overline{\mathsf{lab}}) \leftarrow \mathsf{GCircuit}(1^\lambda, \mathsf{T}[m])$.

Then, id_n, k_{n-1} and $\overline{\mathsf{lab}}$ are hardwired into circuit $P^{n-1}[\mathsf{id}_n, k_{n-1}, \overline{\mathsf{lab}}]$. Next, invoke the garbled circuit $(\tilde{P}^{n-1}, \overline{\mathsf{lab}}') \leftarrow \mathsf{GCircuit}(1^\lambda, P^{n-1}[\mathsf{id}_n, k_{n-1}, \overline{\mathsf{lab}}])$.

Let $\overline{\mathsf{lab}} := \overline{\mathsf{lab}}'$. Invoke $(\tilde{P}^{n-2}, \overline{\mathsf{lab}}') \leftarrow \mathsf{GCircuit}(1^\lambda, P^{n-2}[\mathsf{id}_{n-1}, k_{n-2}, \overline{\mathsf{lab}}])$. Repeat this procedure and we have $(\tilde{P}^0, \overline{\mathsf{lab}}') \leftarrow \mathsf{GCircuit}(1^\lambda, P^0[\mathsf{id}_1, k_0, \overline{\mathsf{lab}}])$. Recall that $\overline{\mathsf{lab}}' = \{\mathsf{lab}_{j,b}\}_{j\in[\lambda],b\in\{0,1\}}$. Choose λ labels from $\overline{\mathsf{lab}}'$ according to the λ bits of h_ϵ.

The final ciphertext is $\mathsf{ct} = (\{\mathsf{lab}_{j,h_{\epsilon j}}\}_{j\in[\lambda]}, \tilde{P}^0, \ldots, \tilde{P}^{n-1}, \tilde{\mathsf{T}})$.

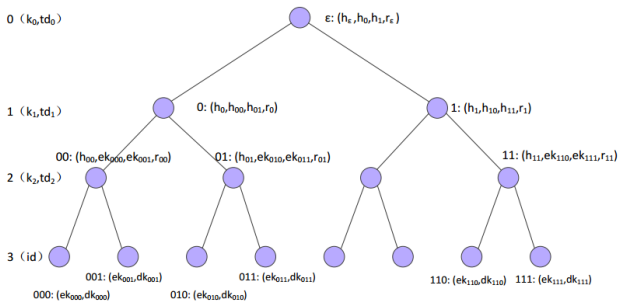

Figure 2. The IBE tree of depth $n = 3$.

Decryption. The decryption goes from the top to bottom. It will invoke the evaluation algorithm Eval of the garbled circuits to obtain chameleon encryption of labels, and uses the secret key of chameleon

encryption scheme to recover the corresponding label. For the leaf, it will use the decryption algorithm of PKE to recover the message m.

3.2. Idea of Our Revoked IBE Scheme

Our revocable IBE is based on the original DG scheme. An important observation of the DG scheme is that among all the elements in the secret key $\mathsf{sk_{id}} = (\{h_{\mathsf{v}}, h_{\mathsf{v}||0}, h_{\mathsf{v}||1}, r_{\mathsf{v}}\}_{v\in\mathcal{V}}, \mathsf{dk_{id}})$ of user id, $\mathsf{dk_{id}}$ is the most critical element. Recall that $\mathcal{V} = \{\varepsilon, \mathsf{id}[1], \mathsf{id}[12], \ldots, \mathsf{id}[12\ldots n-1]\}$ and $\mathsf{dk_{id}}$ is the decryption key of the underlying building block PKE. The sibling of leaf id knows everything about $\mathsf{sk_{id}}$ except $\mathsf{dk_{id}}$. This gives us a hint for revocation. To revoke user id, we can change the decryption key $\mathsf{dk_{id}}$ in $\mathsf{sk_{id}}$ into a new one $\mathsf{dk'_{id}}$ and this fresh decryption key will not issued to the revoked user id. As long as the essential element $\mathsf{dk'_{id}}$ is missing, user id will not be able to decrypt anything. Now we outline how the revocable IBE works.

The tree is updated according to the revoked users.

- If a leaf v_{id} is revoked during time period t, then a new public/secret key pair will generated with $(\mathsf{ek'_{id}}, \mathsf{dk'_{id}}) \leftarrow \mathsf{G}(1^\lambda)$ for this leaf. As a result, $h_{v_{\mathsf{id}}} = \mathsf{ek_{id}}$ is replaced with a fresh value $h^{(\mathsf{t})}_{v_{\mathsf{id}}} := \mathsf{ek'_{id}}$. This fresh value will not consistent to what the father node of v_{id} has. Therefore, we have to change the attachments of all nodes along the path from the revoked leaf v_{id} to root bottom upward.
- For i from $n-1$ down to 0

  Let $v := v_{\mathsf{id}[12\ldots i]}$. Choose random coins $r^{(\mathsf{t})}_v$; $h^{(\mathsf{t})}_v := \mathsf{H}(h^{(\mathsf{t})}_{v||0}, h^{(\mathsf{t})}_{v||1}, r^{(\mathsf{t})}_v)$;

  Here $h^{(\mathsf{t})}_{v||b} := h_{v||b}$ if $h^{(\mathsf{t})}_{v||b}$ is not defined, where $b \in \{0,1\}$.

In this way, a new tree is built with root attached with new value $(h^{(\mathsf{t})}_\varepsilon, h^{(\mathsf{t})}_0, h^{(\mathsf{t})}_1, r^{(\mathsf{t})}_\varepsilon)$. Please Please note that the hash keys $(k_0, \ldots, k_{n-1})$ remain unchanged.

When revocation happens, what a sender does is updating the new hash value $h^{(\mathsf{t})}_\varepsilon$, then invoking the encryption algorithm for encryption.

For decryption to go smoothly, the IBE system has to issue updating keys to users. The updating key includes all the information of the nodes on the paths from revoked leaves to the root, but the new $\mathsf{dk}^{(\mathsf{t})}_{\mathsf{id}}$ is not issued. In Figure 3, for example, two users, namely 000 and 010, are revoked and determine two paths. Then all the nodes along the two paths are marked with cross. All the nodes are updated with new attachments, but leaf 000 is only attached with a new $\mathsf{ek}^{(\mathsf{t})}_{000}$ (without $\mathsf{dk}^{(\mathsf{t})}_{000}$) and leaf 010 is only attached with a new $\mathsf{ek}^{(\mathsf{t})}_{010}$ (without $\mathsf{dk}^{(\mathsf{t})}_{010}$). The updating key are $\{\varepsilon, (h^{(\mathsf{t})}_\varepsilon, h^{(\mathsf{t})}_0, h^{(\mathsf{t})}_1, r^{(\mathsf{t})}_\varepsilon)$, $0, (h^{(\mathsf{t})}_0, h^{(\mathsf{t})}_{00}, h^{(\mathsf{t})}_{01}, r^{(\mathsf{t})}_0)$, $00, (h^{(\mathsf{t})}_{00}, h^{(\mathsf{t})}_{000}, h^{(\mathsf{t})}_{001}, r^{(\mathsf{t})}_{00})$, $01, (h^{(\mathsf{t})}_{01}, h^{(\mathsf{t})}_{010}, h^{(\mathsf{t})}_{011}, r^{(\mathsf{t})}_{01})$, $000, (h^{(\mathsf{t})}_{000} = \mathsf{ek}^{(\mathsf{t})}_{000}, \perp), 010, (h^{(\mathsf{t})}_{010} = \mathsf{ek}^{(\mathsf{t})}_{010}, \perp)\}$.

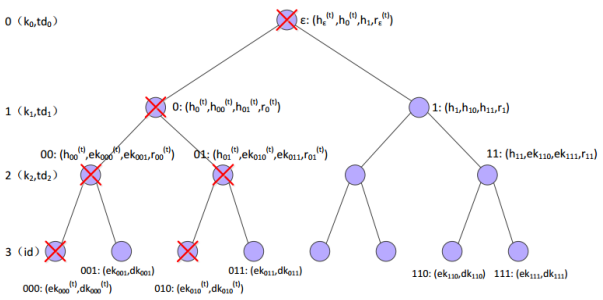

Figure 3. The IBE tree of depth $n = 3$ when users "000" and "010" have been revoked.

Any legal user is able to update his secret key $\mathsf{sk_{id}}$ with the new attachments of nodes along the path from his leaf to the root. For example, the updated secret key $\mathsf{sk}^{(\mathsf{t})}_{001}$ of user 001 is now $\{\varepsilon, (h^{(\mathsf{t})}_\varepsilon, h^{(\mathsf{t})}_0, h^{(\mathsf{t})}_1, r^{(\mathsf{t})}_\varepsilon)$, $0, (h^{(\mathsf{t})}_0, h^{(\mathsf{t})}_{00}, h^{(\mathsf{t})}_{01}, r^{(\mathsf{t})}_0)$, $00, (h^{(\mathsf{t})}_{00}, h^{(\mathsf{t})}_{000}, h^{(\mathsf{t})}_{001}, r^{(\mathsf{t})}_{00})$, $001, (h_{001} =$

$\mathsf{ek}_{001}, \mathsf{dk}_{001})\}$. The updated secret key $\mathsf{sk}_{111}^{(\mathsf{t})}$ of user 111 is now $\{\varepsilon, (h_\varepsilon^{(\mathsf{t})}, h_0^{(\mathsf{t})}, h_1^{(\mathsf{t})}, r_\varepsilon^{(\mathsf{t})}), 1, (h_1, h_{10}, h_{11}, r_1), 11, (h_{11}, h_{110}, h_{111}, r_{11}), 111, (h_{111} = \mathsf{ek}_{001}, \mathsf{dk}_{111})\}$.

In this way, any legal user is able to decrypt ciphertexts since he knows the secret key corresponding to the new tree. Any revoked user id is unable to implement decryption anymore, since the new $\mathsf{dk}_{\mathsf{id}}^{(\mathsf{t})}$ is missing.

4. Revocable IBE Scheme

In this section, we present our construction of revocable IBE scheme from chameleon encryption (without DKER). Let PRF: $\{0,1\}^\lambda \times \{0,1\}^{\leq \ell+n} \cup \{\varepsilon\} \rightarrow \{0,1\}^\lambda$ be a pseudorandom function. Let $\mathsf{CE} = (\mathsf{HGen}, \mathsf{H}, \mathsf{H}^{-1}, \mathsf{HEnc}, \mathsf{HDec})$ be a chameleon encryption scheme and $\mathsf{PKE} = (\mathsf{G}, \mathsf{E}, \mathsf{D})$ be an IND-CPA secure public-key encryption scheme. We denote by $\mathsf{id}[i]$ the i-th bit of id and by $\mathsf{id}[1 \cdots i]$ the first i bits of id. Define $\mathsf{id}[1 \cdots 0] := \varepsilon$. We first introduce five subroutines which will be used repeatedly in our scheme (as shown in Table 4). All of these five subroutines are run by the key authority. The subroutines NodeGen and LeafGen are invoked by the key authority in setup algorithm, where NodeGen is used to generate non-leaf nodes and LeafGen to generate leaves and their parents. Just like [8], given all chameleon keys, trapdoors, a randomness s, a node v and a length parameter ℓ, the NodeGen subroutine generates four values stored in node v: the hash value of the node h_v, the hash value of it left-child node $h_{v||0}$, the hash value of it right-child node $h_{v||1}$, and the randomness of this node r_v. Given all chameleon keys k_{n-1} and trapdoors td_{n-1} of the $n-1$-th level, a randomness s, a node v in the $n-1$-th level and a length parameter ℓ, the LeafGen subroutine generates two pairs of public/secret keys $(\mathsf{ek}_{v||0}, \mathsf{dk}_{v||0})$, $(\mathsf{ek}_{v||1}, \mathsf{dk}_{v||1})$ of the PKE scheme, and generates the hash value h_v and the randomness r_v of the node v. The children of v are two leaves associated by $\mathsf{ek}_{v||0}$ and $\mathsf{ek}_{v||1}$. Each user can be uniquely represented by a leaf node. The subroutine FindNodes, subroutine NodeChange and subroutine LeafChange are invoked by the key authority in key update algorithm. Given a revocation list RL, a time t and the global key list KL, subroutine $\mathsf{FindNodes}(\mathsf{RL}, \mathsf{t}, \mathsf{KL})$ outputs all leaves which are revoked at time t and all their ancestor nodes. Given a chameleon key, a chameleon trapdoor, a node v, two hash values $(h_{v||0}, h_{v||0})$ of the two children of node v and a randomness s, subroutine NodeChange outputs a new hash value and a new randomness for node v. Given a leaf node v, a time t, a randomness s, subroutine LeafChange outputs a fresh public key by invoking the key generation algorithm G of PKE.

Table 4. Five subroutines run by the key authority.

$\underline{\mathsf{NodeGen}}((k_0, \cdots, k_n), (td_0, \cdots, td_n, s), v \in \{0,1\}^{\leq n-1} \cup \{\varepsilon\}, \ell)$:	$\underline{\mathsf{FindNodes}}(\mathsf{RL}, \mathsf{t}, \mathsf{KL})$:
Let $i := \lvert v \rvert$	$\mathsf{Y} \leftarrow \emptyset$
$h_v \leftarrow \mathsf{H}(k_i, 0^{2\lambda}; \mathsf{PRF}(s, 0^\ell \lvert\rvert v))$,	$\forall(\mathsf{id}, \mathsf{t}_i) \in \mathsf{RL}$
$h_{v\lvert\rvert 0} \leftarrow \mathsf{H}(k_{i+1}, 0^{2\lambda}; \mathsf{PRF}(s, 0^\ell \lvert\rvert v \lvert\rvert 0))$,	If t_i=t, then add id to Y.
$h_{v\lvert\rvert 1} \leftarrow \mathsf{H}(k_{i+1}, 0^{2\lambda}; \mathsf{PRF}(s, 0^\ell \lvert\rvert v \lvert\rvert 1))$.	For $i = n-1$ to 0: $\quad \backslash\backslash$ find the ancestors of id $\in$ Y.
$r_v \leftarrow \mathsf{H}^{-1}(td_i, (0^{2\lambda}, \mathsf{PRF}(s, 0^\ell \lvert\rvert v)), h_{v\lvert\rvert 0} \lvert\rvert h_{v\lvert\rvert 1})$.	$\forall(v, \cdot, \cdot) \in \mathsf{KL}$ with $\lvert v \rvert = i$:
Output $(h_v, h_{v\lvert\rvert 0}, h_{v\lvert\rvert 1}, r_v)$.	If $(v\lvert\rvert 0 \in \mathsf{Y}) \vee (v\lvert\rvert 1 \in \mathsf{Y})$, add v to Y.
	Output Y.
$\underline{\mathsf{LeafGen}}(k_{n-1}, (td_{n-1}, s), v \in \{0,1\}^{n-1}, \ell)$:	$\underline{\mathsf{NodeChange}}(k, td, v \in \{0,1\}^{\leq n-1} \cup \{\varepsilon\}, h_{v\lvert\rvert 0}, h_{v\lvert\rvert 1}, \mathsf{t}, s)$:
	$h_v^{(\mathsf{t})} \leftarrow \mathsf{H}(k, 0^{2\lambda}; \mathsf{PRF}(s, \mathsf{t} \lvert\rvert v))$,
$h_v \leftarrow \mathsf{H}(k_n, 0^{2\lambda}; \mathsf{PRF}(s, 0^\ell \lvert\rvert v))$,	$r_v^{(\mathsf{t})} \leftarrow \mathsf{H}^{-1}(td, (0^{2\lambda}, \mathsf{PRF}(s, \mathsf{t} \lvert\rvert v)); h_{v\lvert\rvert 0} \lvert\rvert h_{v\lvert\rvert 1})$.
$(\mathsf{ek}_{v\lvert\rvert 0}, \mathsf{dk}_{v\lvert\rvert 0}) \leftarrow \mathsf{G}(1^\lambda, \mathsf{PRF}(s, 0^\ell \lvert\rvert v \lvert\rvert 0))$,	Output $(h_v^{(\mathsf{t})}, h_{v\lvert\rvert 0}, h_{v\lvert\rvert 1}, r_v^{(\mathsf{t})})$.
$(\mathsf{ek}_{v\lvert\rvert 1}, \mathsf{dk}_{v\lvert\rvert 1}) \leftarrow \mathsf{G}(1^\lambda, \mathsf{PRF}(s, 0^\ell \lvert\rvert v \lvert\rvert 1))$,	$\underline{\mathsf{LeafChange}}(v \in \{0,1\}^n, \mathsf{t}, s)$:
$r_v \leftarrow \mathsf{H}^{-1}(td_{n-1}, (0^{2\lambda}, \mathsf{PRF}(s, 0^\ell \lvert\rvert v)), \mathsf{ek}_{v\lvert\rvert 0} \lvert\rvert \mathsf{ek}_{v\lvert\rvert 1})$.	$(\mathsf{ek}_v^{(\mathsf{t})}, \mathsf{dk}_v^{(\mathsf{t})}) \leftarrow \mathsf{G}(1^\lambda, \mathsf{PRF}(s, \mathsf{t} \lvert\rvert v))$.
Output $((h_v, \mathsf{ek}_{v\lvert\rvert 0}, \mathsf{ek}_{v\lvert\rvert 1}, r_v), \mathsf{dk}_{v\lvert\rvert 0}, \mathsf{dk}_{v\lvert\rvert 1})$.	Output $(\mathsf{ek}_v^{(\mathsf{t})}, \perp)$.

Construction of RIBE. Now we describe our revocable IBE scheme (RIBE.Setup, RIBE.KG, RIBE.KU, RIBE.DK, RIBE.Enc, RIBE.Dec, RIBE.R).

- **Setup** $\mathsf{RIBE.Setup}(1^\lambda, 1^n)$: given a security parameter λ, an integer n where 2^n is the maximal number of users that the scheme supports. Define identity space as $\mathcal{ID} = \{0,1\}^n$ and time space as $\mathcal{T} = \{0,1\}^\ell$, and do the following.

  1. Sample $s \xleftarrow{\$} \{0,1\}^\lambda$.
  2. For each $i \in [n]$, invoke $(k_i, td_i) \xleftarrow{\$} \mathsf{HGen}(1^\lambda, 2\lambda)$.
  3. Initialize key list $\mathsf{KL} := \emptyset$, public list $\mathsf{PL} = \emptyset$, key update list $\mathsf{KU} = \emptyset$ and revocation list $\mathsf{RL} := \emptyset$.
  4. $\mathsf{mpk} := (k_0, \cdots, k_{n-1}, \ell)$; $\mathsf{st} := \{\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU}\}$; $\mathsf{msk} := (\mathsf{mpk}, td_0, \cdots, td_{n-1}, s)$.
  5. Output $(\mathsf{mpk}, \mathsf{msk}, \mathsf{st})$.

- **Private Key Generation** $\mathsf{RIBE.KG}(\mathsf{msk}, \mathsf{id} \in \{0,1\}^n, \mathsf{st})$: See Figure 4 for illustrations.

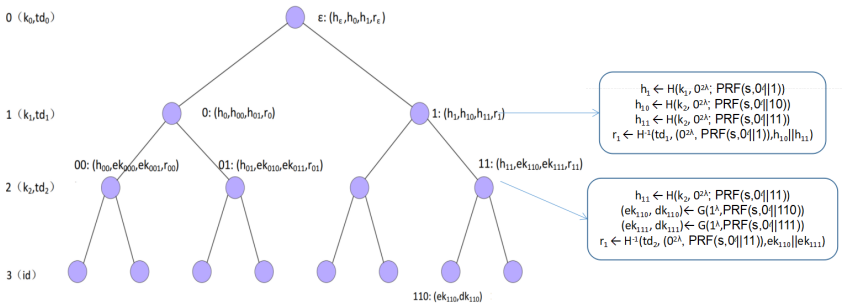

Figure 4. The illustration of Private Key Generation for user "110". The private key sk_{110} of user "110" collects all the node values along the path from "110" to "ε" in the tree.

  1. Parse $\mathsf{msk} = (\mathsf{mpk}, td_0, \cdots, td_{n-1}, s)$ and $\mathsf{mpk} = (k_0, \cdots, k_{n-1}, \ell)$.
  2. $\mathsf{W} := \{\varepsilon, \mathsf{id}[1], \cdots, \mathsf{id}[1 \cdots n-1]\}$, where ε is the empty string.
  3. For all $v \in \mathsf{W} \setminus \{\mathsf{id}[1 \cdots n-1]\}$:
     $(h_v, h_{v||0}, h_{v||1}, r_v) \leftarrow \mathsf{NodeGen}((k_0, \cdots, k_{n-1}), (td_0, \cdots, td_{n-1}, s), v, \ell)$,
     $\mathsf{KL} := \mathsf{KL} \cup \{(v, h_v, h_{v||0}, h_{v||1}, r_v)\}$,
     $\mathsf{lk}_v := (h_v, h_{v||0}, h_{v||1}, r_v)$.
  4. For $v = \mathsf{id}[1 \cdots n-1]$:
     $(h_v, h_{v||0} = \mathsf{ek}_{v||0}, h_{v||1} = \mathsf{ek}_{v||1}, r_v, \mathsf{dk}_{v||0}, \mathsf{dk}_{v||1}) \leftarrow \mathsf{LeafGen}(k_{n-1}, (td_{n-1}, s), v, \ell)$,
     $\mathsf{KL} := \mathsf{KL} \cup \{(v, h_v, \mathsf{ek}_{v||0}, \mathsf{ek}_{v||1}, r_v), (v||0, \mathsf{ek}_{v||0}, \perp), (v||1, \mathsf{ek}_{v||1}, \perp)\}$,
     $\mathsf{lk}_v := (h_v, \mathsf{ek}_{v||0}, \mathsf{ek}_{v||1}, r_v)$.
  5. $\mathsf{st} = \{\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU}\}$ and $\mathsf{sk}_{\mathsf{id}} := (\mathsf{t} = 0, \mathsf{id}, \{\mathsf{lk}_v\}_{v \in \mathsf{W}}, \mathsf{dk}_{\mathsf{id}})$.
  6. Output $(\mathsf{sk}_{\mathsf{id}}, \mathsf{st})$.

- **Key Update Generation** $\mathsf{RIBE.KU}(\mathsf{msk}, \mathsf{t}, \mathsf{st})$: See Figure 5 for illustrations.

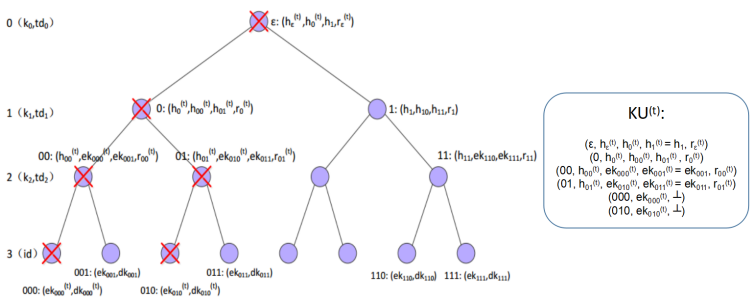

Figure 5. The illustration of $\mathsf{KU}^{(\mathsf{t})}$ when users "000" and "010" have been revoked at time slot t.

1. Parse $\mathsf{msk} = (\mathsf{mpk}, td_0, \cdots, td_{n-1}, s)$, $\mathsf{st} = \{\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU}\}$ and $\mathsf{mpk} = (k_0, \cdots, k_{n-1}, \ell)$.
2. $\mathsf{Y} \leftarrow \mathsf{FindNodes}(\mathsf{RL}, \mathsf{t}, \mathsf{KL})$. // Y stores all revoked leaves and their ancestors
3. If $\mathsf{Y} = \emptyset$, Output(KU, PL) //stay unchanged.
4. Set key update list $\mathsf{KU}^{(\mathsf{t})} := \emptyset$.
5. For all node $v \in \mathsf{Y}$ such that $|v| = n$: // deal with all leaves in Y
   $(\mathsf{ek}_v^{(\mathsf{t})}, \perp) \leftarrow \mathsf{LeafChange}(v, \mathsf{t}, s)$,
   $\mathsf{KU}^{(\mathsf{t})} := \mathsf{KU}^{(\mathsf{t})} \cup \{(v, \mathsf{ek}_v^{(\mathsf{t})}, \perp)\}$. // new attachments for all leaves in Y
   $h_v^{(\mathsf{t})} := \mathsf{ek}_v^{(\mathsf{t})}$.
6. For $i = n-1$ to 0: // generate new attachments for all non-leaf nodes in Y
   For all node $v \in \mathsf{Y}$ and $|v| = i$:
   Set $j := \mathsf{t}$, $\mathsf{KU}^{(0)} := \mathsf{KL}$.
   While$(j \geq 0)$
   If $\exists v||b$ s.t. $(v||b, h_{v||b}, \cdot, \cdot, \cdot) \in \mathsf{KU}^{(j)}$,
   $h_{v||b}^{(\mathsf{t})} := h_{v||b}$,
   Break;
   $j := j - 1$.
   $(h_v^{(\mathsf{t})}, h_{v||0}^{(\mathsf{t})}, h_{v||1}^{(\mathsf{t})}, r_v^{(\mathsf{t})}) \leftarrow \mathsf{NodeChange}(k_i, td_i, v, h_{v||0}^{(\mathsf{t})}, h_{v||1}^{(\mathsf{t})}, \mathsf{t}, s)$.
   $\mathsf{KU}^{(\mathsf{t})} := \mathsf{KU}^{(\mathsf{t})} \cup \{(v, h_v^{(\mathsf{t})}, h_{v||0}^{(\mathsf{t})}, h_{v||1}^{(\mathsf{t})}, r_v^{(\mathsf{t})})\}$.
7. $\mathsf{KU} := \mathsf{KU} \cup \{(\mathsf{t}, \mathsf{KU}^{(\mathsf{t})})\}$ and $\mathsf{PL} := \mathsf{PL} \cup \{(\mathsf{t}, h_\varepsilon^{(\mathsf{t})})\}$.
8. $\mathsf{st} := \{\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU}\}$
9. Output st.

- **Decryption Key Generation** $\mathsf{RIBE.DK}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{id}}, \mathsf{KU}, \mathsf{t})$: See Figure 6 for illustrations.

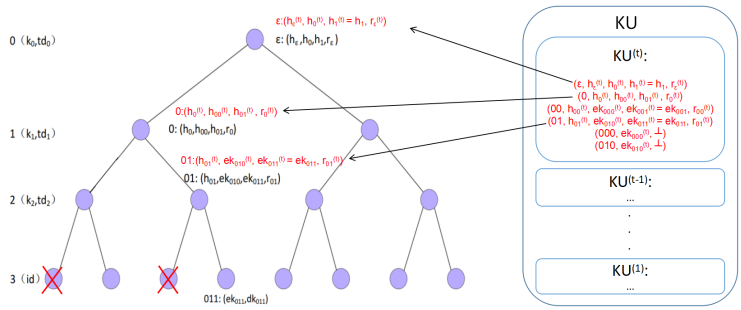

Figure 6. The illustration of the Decryption Key Generation for user "011" when users "000" and "010" have been revoked at time slot t. For i from 1 to t, the node values along the path from "011" to "ε" in the tree will be replaced by the corresponding node values in $\mathsf{KU}^{(i)}$. The decryption key $\mathsf{sk}_{011}^{(\mathsf{t})}$ of user "011" collects all the updated node values along the path from "011" to "ε" in the tree.

1. $\mathsf{W} := \{\varepsilon, \mathsf{id}[1], \cdots, \mathsf{id}[1 \cdots n-1]\}$, where ε is the empty string.
2. Parse $\mathsf{mpk} = (k_0, \cdots, k_{n-1}, \ell)$ and $\mathsf{sk}_{\mathsf{id}} = (0, \mathsf{id}, \{h_v, h_{v||0}, h_{v||1}, r_v\}_{v \in \mathsf{W}}, \mathsf{dk}_{\mathsf{id}})$.
3. From KU retrieve a set $\Omega := \{(\tilde{\mathsf{t}}, \mathsf{KU}^{(\tilde{\mathsf{t}})}) \mid (\tilde{\mathsf{t}}, \mathsf{KU}^{(\tilde{\mathsf{t}})}) \in \mathsf{KU}, 0 \leq \tilde{\mathsf{t}} < \mathsf{t}\}$.
4. For each $(\tilde{\mathsf{t}}, \mathsf{KU}^{(\tilde{\mathsf{t}})}) \in \Omega$ with $\tilde{\mathsf{t}}$ in ascending order, does the following:
   For $i = 0$ to $n-1$:
   $v := \mathsf{id}[1 \cdots i]$ (Recall $\mathsf{id}[1 \cdots 0] = \varepsilon$).
   If $\exists (v, h_v^{(\tilde{\mathsf{t}})}, h_{v||0}^{(\tilde{\mathsf{t}})}, h_{v||1}^{(\tilde{\mathsf{t}})}, r_v^{(\tilde{\mathsf{t}})}) \in \mathsf{KU}^{(\tilde{\mathsf{t}})}$:
   $\mathsf{lk}_v^{(\mathsf{t})} := (h_v^{(\tilde{\mathsf{t}})}, h_{v||0}^{(\tilde{\mathsf{t}})}, h_{v||1}^{(\tilde{\mathsf{t}})}, r_v^{(\tilde{\mathsf{t}})})$.

5. If $\exists(\tilde{t}, \mathsf{KU}^{(\tilde{t})}) \in \mathsf{KU}$ s.t. $(\mathsf{id}, \mathsf{ek}_v^{(\tilde{t})}, \perp) \in \mathsf{KU}^{(\tilde{t})}$: \\id is revoked at $\tilde{t}$
   Output $\mathsf{sk}_{\mathsf{id}}^{(t)} := (t, \mathsf{id}, \{\mathsf{lk}_v^{(t)}\}_{v \in \mathsf{W}}, \perp)$.
6. Output $\mathsf{sk}_{\mathsf{id}}^{(t)} := (t, \mathsf{id}, \{\mathsf{lk}_v^{(t)}\}_{v \in \mathsf{W}}, \mathsf{dk}_{\mathsf{id}})$

- **Encryption** $\mathsf{RIBE.Enc}(\mathsf{mpk}, \mathsf{id}, t, m, \mathsf{PL})$):
  We describe two circuits that will be garbled during the encryption procedure.
  - $\mathsf{Q}[m](ek)$: Compute and output $\mathsf{E}(ek, m)$.
  - $\mathsf{P}[\beta \in \{0,1\}, k, \overline{\mathsf{lab}}](h)$: Compute and output $\{\mathsf{HEnc}(k, (h, j + \beta \cdot \lambda, b), \mathsf{lab}_{j,b})\}_{j \in [\lambda], b \in \{0,1\}}$, where $\overline{\mathsf{lab}}$ is the short for $\{\mathsf{lab}_{j,b}\}_{j \in [\lambda], b \in \{0,1\}}$.

  Encryption proceeds as follows:

  1. Retrieve the last item $(\bar{t}, h_\epsilon^{(\bar{t})})$ from PL. If $t < \bar{t}$, output $\perp$; otherwise $h_\epsilon^{(t)} := h_\epsilon^{(\bar{t})}$.
  2. Parse $\mathsf{mpk} = (k_0, \cdots, k_{n-1}, \ell)$.
  3. $(\tilde{Q}, \overline{\mathsf{lab}}) \xleftarrow{\$} \mathsf{GCircuit}(1^\lambda, \mathsf{Q}[m])$.
  4. For $i = n-1$ to 0,
     $(\tilde{P}^i, \overline{\mathsf{lab}}') \xleftarrow{\$} \mathsf{GCircuit}(1^\lambda, \mathsf{P}[\mathsf{id}[i+1], k_i, \overline{\mathsf{lab}}])$ and set $\overline{\mathsf{lab}} := \overline{\mathsf{lab}}'$.
  5. Output $\mathsf{ct} := \left(\left\{ \mathsf{lab}_{j, h_{\varepsilon,j}^{(t)}} \right\}_{j \in [\lambda]}, \{\tilde{P}^0, \cdots, \tilde{P}^{n-1}, \tilde{Q}\} \right)$, where $h_{\varepsilon,j}^{(t)}$ is the j^{th} bit of $h_\varepsilon^{(t)}$.

- **Decryption** $\mathsf{RIBE.Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{id}}^{(t)}, \mathsf{ct})$

  1. $\mathsf{W} := \{\varepsilon, \mathsf{id}[1], \cdots, \mathsf{id}[1 \cdots n-1]\}$, where ε is the empty string.
  2. Parse $\mathsf{mpk} = (k_0, \cdots, k_{n-1}, \ell)$ and $\mathsf{sk}_{\mathsf{id}}^{(t)} = (\mathsf{id}, \{\mathsf{lk}_v^{(t)}\}_{v \in \mathsf{W}}, \mathsf{dk}_{\mathsf{id}})$, where $\mathsf{lk}_v^{(t)} = (h_v^{(t)}, h_{v||0}^{(t)}, h_{v||1}^{(t)}, r_v^{(t)})$.
  3. Parse $\mathsf{ct} := \left(\left\{ \mathsf{lab}_{j, h_{\varepsilon,j}^{(t)}} \right\}_{j \in [\lambda]}, \{\tilde{P}^0, \cdots, \tilde{P}^{n-1}, \tilde{Q}\} \right)$
  4. Set $y := h_\varepsilon^{(t)}$.
  5. For $i = 0$ to $n-1$:
     Set $v := \mathsf{id}[1 \cdots i]$ (Recall $\mathsf{id}[1 \cdots 0] = \varepsilon$);
     $\{c_{j,b}\}_{j \in [\lambda], b \in \{0,1\}} \leftarrow \mathsf{Eval}(\tilde{P}^i, \{\mathsf{lab}_{j,y_j}\}_{j \in [\lambda]})$;
     If $i \neq n-1$, set $v' := \mathsf{id}[1 \cdots i+1]$ and $y := h_{v'}^{(t)}$, and for each $j \in [\lambda]$,

     $$\{\mathsf{lab}_{j,y_j}\}_{j \in [\lambda]} \leftarrow \mathsf{HDec}(k_i, c_{j,y_j}, (h_{v||0}^{(t)} || h_{v||1}^{(t)}), r_v^{(t)}).$$

     If $i = n-1$, set $y := \mathsf{ek}_{\mathsf{id}}$ and for each $j \in [\lambda]$, compute

     $$\{\mathsf{lab}_{j,y_j}\}_{j \in [\lambda]} \leftarrow \mathsf{HDec}(k_i, c_{j,y_j}, (\mathsf{ek}_{v||0} || \mathsf{ek}_{v||1}) = (h_{v||0}^{(t)} || h_{v||1}^{(t)}), r_v^{(t)}).$$

  6. Compute $f \leftarrow \mathsf{Eval}(\tilde{Q}, \{\mathsf{lab}_{j,y_j}\}_{j \in [\lambda]})$.
  7. Output $m \leftarrow \mathsf{D}(\mathsf{dk}_{\mathsf{id}}, f)$.

- **Revocation** $\mathsf{RIBE.R}(\mathsf{id}, t, \mathsf{st})$:

  1. Parse $\mathsf{st} := \{\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU}\}$.
  2. Update the revocation list by $\mathsf{RL} := \mathsf{RL} \cup \{(\mathsf{id}, t)\}$.
  3. $\mathsf{st} := \{\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU}\}$.
  4. Output st.

Remark. It is possible for us to reduce the cost of users' key updating in our construction. Now we provide a more efficient variant of decryption key generation algorithm RIBE.DK'. With this variant algorithm, if a user has already generated a key $\mathsf{sk}_{\mathsf{id}}^{(\mathsf{t}')}$ at time period t' where $\mathsf{t}' \leq \mathsf{t}$, he or she can use $\mathsf{sk}_{\mathsf{id}}^{(\mathsf{t}')}$ as the input instead of $\mathsf{sk}_{\mathsf{id}}$ and generates the decryption key with lower computational cost. The algorithm proceeds as follows:

Decryption Key Generation $\mathsf{RIBE.DK'}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{id}}^{(\mathsf{t}')}, \mathsf{KU}, \mathsf{t})$:

1. $\mathsf{W} := \{\varepsilon, \mathsf{id}[1], \cdots, \mathsf{id}[1 \cdots n-1]\}$, where ε is the empty string.
2. Parse $\mathsf{mpk} = (k_0, \cdots, k_{n-1}, \ell)$ and $\mathsf{sk}_{\mathsf{id}}^{(\mathsf{t}')} = (\mathsf{t}', \mathsf{id}, \{h_v, h_{v||0}, h_{v||1}, r_v\}_{v \in \mathsf{W}}, \mathsf{dk}_{\mathsf{id}})$.
3. If $\mathsf{t}' > \mathsf{t}$, Output $\perp$.
4. If $\mathsf{t}' = \mathsf{t}$, Output $\mathsf{sk}_{\mathsf{id}}^{(\mathsf{t}')}$.
5. From KU retrieve a set $\Omega := \{(\tilde{\mathsf{t}}, \mathsf{KU}^{(\tilde{\mathsf{t}})}) \mid (\tilde{\mathsf{t}}, \mathsf{KU}^{(\tilde{\mathsf{t}})}) \in \mathsf{KU}, \mathsf{t}' \leq \tilde{\mathsf{t}} < \mathsf{t}\}$.
6. For each $(\tilde{\mathsf{t}}, \mathsf{KU}^{(\tilde{\mathsf{t}})}) \in \Omega$ with $\tilde{\mathsf{t}}$ in ascending order, does the following:
   For $i = 0$ to $n-1$:
   $v := \mathsf{id}[1 \cdots i]$ (Recall $\mathsf{id}[1 \cdots 0] = \varepsilon$).
   If $\exists (v, h_v^{(\tilde{\mathsf{t}})}, h_{v||0}^{(\tilde{\mathsf{t}})}, h_{v||1}^{(\tilde{\mathsf{t}})}, r_v^{(\tilde{\mathsf{t}})}) \in \mathsf{KU}^{(\tilde{\mathsf{t}})}$:
   $\mathsf{lk}_v^{(\mathsf{t})} := (h_v^{(\tilde{\mathsf{t}})}, h_{v||0}^{(\tilde{\mathsf{t}})}, h_{v||1}^{(\tilde{\mathsf{t}})}, r_v^{(\tilde{\mathsf{t}})})$.
7. If $\exists (\tilde{\mathsf{t}}, \mathsf{KU}^{(\tilde{\mathsf{t}})}) \in \mathsf{KU}$ s.t. $(\mathsf{id}, \mathsf{ek}_v^{(\tilde{\mathsf{t}})}, \perp) \in \mathsf{KU}^{(\tilde{\mathsf{t}})}$: \\id is revoked at $\tilde{\mathsf{t}}$
   Output $\mathsf{sk}_{\mathsf{id}}^{(\mathsf{t})} := (\mathsf{t}, \mathsf{id}, \{\mathsf{lk}_v^{(\mathsf{t})}\}_{v \in \mathsf{W}}, \perp)$.
8. Output $\mathsf{sk}_{\mathsf{id}}^{(\mathsf{t})} := (\mathsf{t}, \mathsf{id}, \{\mathsf{lk}_v^{(\mathsf{t})}\}_{v \in \mathsf{W}}, \mathsf{dk}_{\mathsf{id}})$.

4.1. Correctness

We first show that our revocable IBE is correct. During the time slot t, the key updating algorithm RIBE.KU (together with the key generation algorithm RIBE.KG) uniquely determines a fresh tree of time t. The root of the fresh tree has attachment $(h_\varepsilon^{(\mathsf{t})}, h_0^{(\mathsf{t})}, h_1^{(\mathsf{t})}, r_\varepsilon^{(\mathsf{t})})$. Set $\mathsf{W} := \{\varepsilon, \mathsf{id}[1], \cdots, \mathsf{id}[1 \cdots n-1]\}$, where ε is the empty string. Please Please note that each id uniquely determines a path (from the root of the tree to the leaf of id). W records all non-leaf nodes on the path. For all nodes $v \in \mathsf{W}$, we have $\mathsf{H}(k_{|v|}, h_{v||0}^{(\mathsf{t})} || h_{v||1}^{(\mathsf{t})}; r_v^{(\mathsf{t})}) = h_v^{(\mathsf{t})}$, and $(h_{v||0}^{(\mathsf{t})}, h_{v||1}^{(\mathsf{t})}) := (\mathsf{ek}_{v||0}, \mathsf{ek}_{v||1})$ if $|v| = n-1$.

Consider the ciphertext $\mathsf{ct} = \left(\left\{ \mathsf{lab}_{\ell, h_{\varepsilon,\ell}^{(\mathsf{t})}} \right\}_{\ell \in [\lambda]}, \{\tilde{P}^0, \cdots, \tilde{P}^n, \tilde{Q}\} \right)$, which is the output of $\mathsf{RIBE.Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{t}, m, \mathsf{PL})$. Consider the secret key $\mathsf{sk}_{\mathsf{id}}^{(\mathsf{t})} := (\mathsf{id}, \{\mathsf{lk}_v^{(\mathsf{t})}\}_{v \in \mathsf{W}}, \mathsf{dk}_{\mathsf{id}})$, which is the output RIBE.DK. Obviously, $\mathsf{sk}_{\mathsf{id}}^{(\mathsf{t})}$ is exactly the the secret key of id in the tree (of time t). As long as the $h_\varepsilon^{(\mathsf{t})}$ used in RIBE.Enc to generate ct is identical to the $h_\varepsilon^{(\mathsf{t})}$ in $\mathsf{lk}_\varepsilon^{(\mathsf{t})} = (h_\varepsilon^{(\mathsf{t})}, h_0^{(\mathsf{t})}, h_1^{(\mathsf{t})}, r_\varepsilon^{(\mathsf{t})})$, the decryption RIBE.Dec can always recover the plaintext due to the correctness of the DG scheme.

Below we show the details of the correctness (this analysis is similar to that in [8]). For all nodes $v \in \mathsf{W}$, we have the following facts.

1. $\{c_{j,b}\}_{j \in [\lambda], b \in \{0,1\}} := \mathsf{Eval}\left(\tilde{P}^{|v|}, \left\{ \mathsf{lab}_{j, h_{v,j}^{(\mathsf{t})}} \right\}_{j \in [\lambda]} \right) = P[\mathsf{id}[|v|+1], k_{|v|},$
   $\{\mathsf{lab}'_{j,b}\}_{j \in [\lambda], b}](h_v^{(\mathsf{t})}) = \{\mathsf{HEnc}(k_{|v|}, (h_v^{(\mathsf{t})}, j + \mathsf{id}[|v|+1] \cdot \lambda, b), \mathsf{lab}'_{j,b})\}_{j \in [\lambda], b \in \{0,1\}}$. Recall that $\overline{\mathsf{lab}'} := \{\mathsf{lab}'_{j,b}\}_{j \in [\lambda], b \in \{0,1\}}$ and $(\overline{\mathsf{lab}'}, \tilde{P}^{(|v|+1)})$ are the output of $\mathsf{GCircuit}(1^\lambda, \mathsf{P}[\mathsf{id}[|v|+2], k_{|v|+1}, \overline{\mathsf{lab}''}])$.

2. Due to the correctness of the chameleon encryption, we know that given $(h^{(\mathbf{t})}_{v||0}, h^{(\mathbf{t})}_{v||1}, r^{(\mathbf{t})}_{v})$ one can recover $\left\{\mathsf{lab}'_{\ell,h^{(\mathbf{t})}_{v||\mathbf{id}[|v|+1],\ell}}\right\}_{\ell\in[\lambda]}$ by decrypting $\{c_{j,h^{(\mathbf{t})}_{v||\mathbf{id}[|v|+1],j}}\}_{j\in[\lambda]}$. And $\left\{\mathsf{lab}'_{\ell,h^{(\mathbf{t})}_{v||\mathbf{id}[|v|+1],\ell}}\right\}_{\ell\in[\lambda]}$ is the label for the next garbled circuit $\tilde{P}^{(|v|+1)}$.
3. When $|v| = n-1$, we obtain the set of labels $\left\{\mathsf{lab}_{j,\mathsf{ek}_{\mathsf{id}},j}\right\}_{j\in[\lambda]}$. Recall that $\{\mathsf{lab}_{j,b}\}_{j\in[\lambda],b\in\{0,1\}}$ and $\tilde{Q}$ are the output of $\mathsf{GCircuit}(1^\lambda, Q[m])$. And $\left\{\mathsf{lab}_{j,\mathsf{ek}_{\mathsf{id}},j}\right\}_{j\in[\lambda]}$ is the result of $\{\mathsf{lab}_{j,b}\}_{j\in[\lambda],b\in\{0,1\}}$ selected by $\mathsf{ek}_{\mathsf{id}}$. Thus,

$$f := \mathsf{Eval}\left(\tilde{Q}, \left\{\mathsf{lab}_{j,\mathsf{ek}_{\mathsf{id}},j}\right\}_{j\in[\lambda]}\right) = Q[m](\mathsf{ek}_{\mathsf{id}}) = \mathsf{E}(\mathsf{ek}_{\mathsf{id}}, m).$$

Due to the correctness of $\mathsf{PKE} = (\mathsf{G}, \mathsf{E}, \mathsf{D})$, given decryption key $\mathsf{dk}_{\mathsf{id}}$, one can always recover the original message m correctly with $m \leftarrow \mathsf{D}(\mathsf{dk}_{\mathsf{id}}, f)$.

4.2. Security

In this subsection, we prove that our revocable IBE scheme is IND-ID-CPA secure. Assume q is a polynomial upper bound for the running-time of an adversary $\mathcal{A}$, and it is also an upper bound for the number of $\mathcal{A}$'s queries (which contains private key queries, key update queries, and revocation queries).

Theorem 1. *Assume that t_{max} is the size of the time space and 2^n be the maximal number of users. If PRF is a pseudorandomn function, the garbled circuit scheme is secure, the chameleon encryption scheme CE is secure and $PKE = (G, E, D)$ is IND-CPA secure, the above proposed revocable IBE scheme is adaptive-IND-ID-CPA secure (without decryption key exposure resistance) More specificly, for any PPT adversary $\mathcal{A}$ issuing at most q queries, there exist PPT adversaries $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$ and $\mathcal{B}_4$ such that*

$$\begin{aligned}\boldsymbol{Adv}^{IND\text{-}ID\text{-}CPA}_{\mathcal{A}}(\lambda) \leq\ & \boldsymbol{Adv}^{PRF}_{\mathcal{B}_1}(\lambda) + (n+1)\cdot\boldsymbol{Adv}^{GC}_{\mathcal{B}_2}(\lambda) + n\cdot\lambda\cdot\boldsymbol{Adv}^{CE}_{\mathcal{B}_3}(\lambda) \\ & + (2q+1)\cdot\boldsymbol{Adv}^{PKE}_{\mathcal{B}_4}(\lambda). \end{aligned} \quad (1)$$

Proof. The full proof of Theorem 1 is in Appendix B.1. □

5. Revocable IBE Scheme with DKER

In this section, we present the construction of revocable IBE scheme with decryption key exposure resistance from the CDH assumption. In [24], Katsumata et al. provided a generic construction of RIBE scheme with DKER from a hierarchal IBE (HIBE) scheme (the formal definition of HIBE is provided in Appendix A) and a RIBE scheme without DKER. Following this idea, based on the previous RIBE scheme $\mathsf{RIBE} = (\mathsf{RIBE.Setup}, \mathsf{RIBE.KG}, \mathsf{RIBE.KU}, \mathsf{RIBE.DK}, \mathsf{RIBE.Enc}, \mathsf{RIBE.Dec}, \mathsf{RIBE.R})$ in Section 4 and a HIBE scheme $\mathsf{HIBE} = (\mathsf{HIBE.Setup}, \mathsf{RIBE.KG}, \mathsf{HIBE.Enc}, \mathsf{HIBE.Dec})$ in [8], both of which are based on the CDH assumption, we can construct a revocable IBE scheme Π with DKER from the CDH assumption.

Let $\mathcal{ID}$, $\mathcal{T}$ and $\mathcal{M}$ denote identity space, time period space and plaintext space respectively. We assume $\Pi.\mathcal{ID} = \mathsf{RIBE}.\mathcal{ID}$ and $\Pi.\mathcal{T} = \mathsf{RIBE}.\mathcal{T}$. $\forall\ \mathsf{id} \in \mathsf{RIBE}.\mathcal{ID}$ and $\forall\ \mathsf{t} \in \mathsf{RIBE}.\mathcal{T}$, $(\mathsf{id}||\mathsf{t}) \in \mathsf{HIBE}.\mathcal{ID}$. In addition, we assume $\Pi.\mathcal{M} = \mathsf{RIBE}.\mathcal{M} = \mathsf{HIBE}.\mathcal{M}$.

Construction of RIBE with DKER. Now we describe our revocable IBE scheme $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{KU}, \mathsf{DK}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{R})$ with DKER following [24].

- $\mathsf{Setup}(1^\lambda, 1^n)$: given a security parameter λ, an integer n where 2^n is the maximal number of users that the scheme supports, i.e., $\Pi.\mathcal{ID} = \{0,1\}^n$. Define the time space as $\Pi.\mathcal{T} = \{0,1\}^\ell$.

  1. Run $(\mathsf{RIBE.mpk}, \mathsf{RIBE.msk}, \mathsf{RIBE.st}) \leftarrow \mathsf{RIBE.Setup}(1^\lambda, 1^n)$.
  2. Parse $\mathsf{RIBE.st} := \{\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU}\}$.
  3. Run $(\mathsf{HIBE.mpk}, \mathsf{HIBE.msk}) \leftarrow \mathsf{HIBE.Setup}(1^\lambda)$.
  4. Output $(\mathsf{mpk} := (\mathsf{RIBE.mpk}, \mathsf{HIBE.mpk}), \mathsf{msk} := (\mathsf{RIBE.msk}, \mathsf{HIBE.msk}), \mathsf{st} := \mathsf{RIBE.st})$.

- $\mathsf{KG}(\mathsf{msk}, \mathsf{id} \in \{0,1\}^n, \mathsf{st})$:

  1. Parse $\mathsf{msk} := (\mathsf{RIBE.msk}, \mathsf{HIBE.msk})$ and $\mathsf{st} := \mathsf{RIBE.st}$.
  2. Run $(\mathsf{RIBE.sk_{id}}, \mathsf{RIBE.st}) \leftarrow \mathsf{RIBE.KG}(\mathsf{RIBE.msk}, \mathsf{id} \in \{0,1\}^n, \mathsf{RIBE.st})$.
  3. Run $\mathsf{HIBE.sk_{id}} \leftarrow \mathsf{HIBE.KG}(\mathsf{HIBE.msk}, \mathsf{id} \in \{0,1\}^n)$.
  4. Output $(\mathsf{sk_{id}} := (\mathsf{RIBE.sk_{id}}, \mathsf{HIBE.sk_{id}}), \mathsf{st} := \mathsf{RIBE.st})$.

- $\mathsf{KU}(\mathsf{msk}, \mathsf{t}, \mathsf{st})$:

  1. Parse $\mathsf{msk} := (\mathsf{RIBE.msk}, \mathsf{HIBE.msk})$ and $\mathsf{st} := \mathsf{RIBE.st}$.
  2. Run $\mathsf{RIBE.st} \leftarrow \mathsf{RIBE.KU}(\mathsf{RIBE.msk}, \mathsf{t}, \mathsf{RIBE.st})$.
  3. Output $(\mathsf{st} := \mathsf{RIBE.st})$.

- $\mathsf{DK}(\mathsf{mpk}, \mathsf{sk_{id}}, \mathsf{KU}, \mathsf{t})$:

  1. Parse $\mathsf{mpk} := (\mathsf{RIBE.mpk}, \mathsf{HIBE.mpk})$ and $\mathsf{sk_{id}} := (\mathsf{RIBE.sk_{id}}, \mathsf{HIBE.sk_{id}})$.
  2. Run $\mathsf{RIBE.sk_{id}^{(t)}} \leftarrow \mathsf{RIBE.DK}(\mathsf{RIBE.mpk}, \mathsf{RIBE.sk_{id}}, \mathsf{KU}, \mathsf{t})$.
  3. Run $\mathsf{HIBE.sk_{id||t}} \leftarrow \mathsf{HIBE.KG}(\mathsf{HIBE.sk_{id}}, \mathsf{t} \in \{0,1\}^\ell)$.
  4. Output $\mathsf{sk_{id}^{(t)}} := (\mathsf{RIBE.sk_{id}^{(t)}}, \mathsf{HIBE.sk_{id||t}})$.

- $\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{t}, m, \mathsf{PL}))$:

  1. Parse $\mathsf{mpk} := (\mathsf{RIBE.mpk}, \mathsf{HIBE.mpk})$.
  2. Sample a pair $(\mathsf{RIBE}.m, \mathsf{HIBE}.m) \in \mathcal{M}^2$ uniformly at random, subject to $\mathsf{RIBE}.m + \mathsf{HIBE}.m = m$.
  3. Run $\mathsf{RIBE.ct} \leftarrow \mathsf{RIBE.Enc}(\mathsf{RIBE.mpk}, \mathsf{id}, \mathsf{t}, \mathsf{RIBE}.m, \mathsf{PL})$.
  4. Run $\mathsf{HIBE.ct} \leftarrow \mathsf{HIBE.Enc}(\mathsf{HIBE.mpk}, (\mathsf{id}||\mathsf{t}), \mathsf{HIBE}.m)$.
  5. Output $\mathsf{ct} := (\mathsf{RIBE.ct}, \mathsf{HIBE.ct})$.

- $\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk_{id}^{(t)}}, \mathsf{ct})$:

  1. Parse $\mathsf{mpk} := (\mathsf{RIBE.mpk}, \mathsf{HIBE.mpk})$ and $\mathsf{sk_{id}^{(t)}} := (\mathsf{RIBE.sk_{id}^{(t)}}, \mathsf{HIBE.sk_{id||t}})$.
  2. Run $\mathsf{RIBE.m} \leftarrow \mathsf{RIBE.Dec}(\mathsf{RIBE.mpk}, \mathsf{RIBE.sk_{id}^{(t)}}, \mathsf{RIBE.ct})$.
  3. Run $\mathsf{HIBE.m} \leftarrow \mathsf{HIBE.Dec}(\mathsf{HIBE.mpk}, \mathsf{HIBE.sk_{id||t}}, \mathsf{HIBE.ct},)$.
  4. Output $m := \mathsf{RIBE.m} + \mathsf{HIBE.m}$.

- $\mathsf{R}(\mathsf{id}, \mathsf{t}, \mathsf{st})$:

  1. Run $\mathsf{RIBE.st} \leftarrow \mathsf{RIBE.R}(\mathsf{id}, \mathsf{t}, \mathsf{st})$.
  2. Output $\mathsf{st} := \mathsf{RIBE.st}$.

Obviously, the correctness of scheme Π follows from the correctness of the underlying RIBE scheme and HIBE scheme. The security of scheme Π is guaranteed by the following theorem.

Theorem 2. *(Theorem 1 in [24]) If the underlying RIBE scheme in the above RIBE scheme* Π *is selective-IND-ID-CPA secure but without decryption key exposure resistance (DKER), and the underlying*

HIBE scheme in Π *is selective-IND-ID-CPA secure, then the resulting RIBE scheme* Π *is selective-IND-ID-CPA secure with DKER.*

Please note that our RIBE scheme RIBE in Section 4 is adaptive-IND-ID-CPA secure without DKER and the hierarchal IBE HIBE constructed in [8] is selective-IND-ID-CPA secure. Both of RIBE and HIBE are based on the CDH assumption. Following Theorem 2, the constructed RIBE scheme Π will be selective-IND-ID-CPA secure with DKER based on the CDH assumption.

Corollary 1. *When instantiating the building blocks with our RIBE scheme RIBE in Section 4 and the hierarchal IBE HIBE in [8], the RIBE scheme* Π *is selective-IND-ID-CPA secure with DKER based on the CDH assumption.*

6. Server-Aided Revocable IBE Scheme

In this section, we present a server-aided version of our revocable IBE scheme. Following the ideas in Sections 4 and 5, we use a standard HIBE scheme $\mathsf{HIBE} = (\mathsf{HIBE.Setup}, \mathsf{HIBE.KG}, \mathsf{HIBE.Enc}, \mathsf{HIBE.Dec})$ in [8] as a building block to construct such a SR-IBE Σ, so that Σ can obtain DKER. To describe our server-aided revocable IBE scheme Σ, we make use of these five subroutines (NodeGen, LeafGen, FindNodes, NodeChange, LeafChange) as defined in Section 4.

Let $\Sigma.\mathcal{ID}$, $\Sigma.\mathcal{T}$ and $\Sigma.\mathcal{M}$ denote the identity space, the time period space and the plaintext space of scheme Σ respectively. Let $\mathsf{HIBE}.\mathcal{ID}$ and $\mathsf{HIBE}.\mathcal{M}$ denote the identity space and the plaintext space of scheme HIBE respectively. For all $\mathsf{id} \in \Sigma.\mathcal{ID}$ and all $\mathsf{t} \in \Sigma.\mathcal{T}$, we assume $(\mathsf{id}||\mathsf{t}) \in \mathsf{HIBE}.\mathcal{ID}$. In addition, we assume $\Sigma.\mathcal{M} = \mathsf{HIBE}.\mathcal{M}$.

Idea. To convert the RIBE scheme Π with DKER in Section 5 to the SR-IBE scheme Σ, the problem is how to divide the decryption ability between the server and the users.

- **Key Generation:** Recall that in RIBE Π the secret key of a user is $(\Pi.\mathsf{sk}_{\mathsf{id}} := (\mathsf{RIBE.sk}_{\mathsf{id}}, \mathsf{HIBE.sk}_{\mathsf{id}})$. Moreover, as shown in Figure 2, the RIBE private key $\mathsf{RIBE.sk}_{\mathsf{id}}$ can be treated as a path from the root to the leaf corresponding to id in a tree. Now for SR-IBE Σ, we divide the RIBE private key $\mathsf{RIBE.sk}_{\mathsf{id}}$ into two parts, the non-leaf part and the leaf part. The non-leaf part (we name it $\mathsf{pk}_{\mathsf{id}}$) is assigned to the server and the leaf part $(\mathsf{ek}_{\mathsf{id}}, \mathsf{dk}_{\mathsf{id}})$ (in fact $\mathsf{dk}_{\mathsf{id}}$ is enough) to user id. Besides, user id is also assigned with the HIBE private key $\mathsf{HIBE.sk}_{\mathsf{id}}$. This is shown in Figure 7.

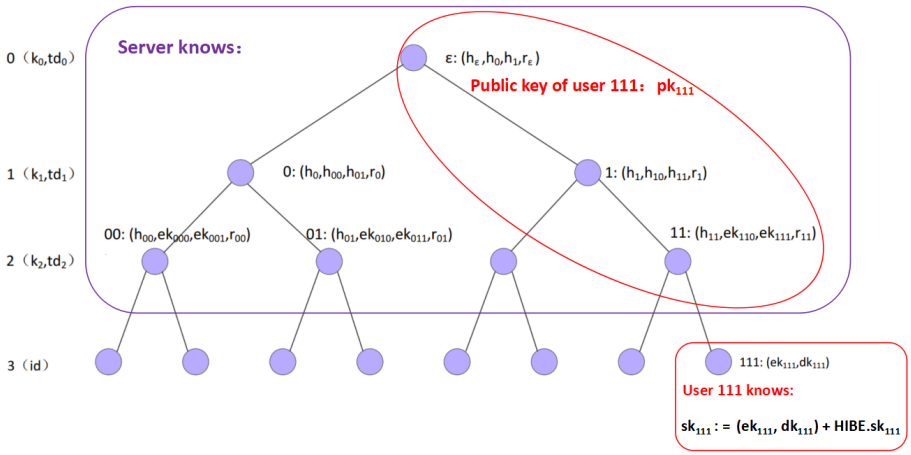

Figure 7. Separation of $\Pi.\mathsf{sk}_{111}$ to the server and the user "111" in SR-IBE Σ, where pk_{111} is the public key and sk_{111} is the private key of user "111".

- **Key Update:** If a user has been revoked in RIBE Π, the updating information in the leaf node corresponding to the user will not be issued. In other words, all the key updating information only occurs in the upper part of the tree excluding the leaves. Therefore, in SR-IBE Σ the key

authority can issue the key updating list to the server and the server is in charge of updating keys for users.

- **Decryption:** Recall that in the RIBE scheme Π with DKER in Section 5, the ciphertext consists of two parts: the ciphertext of the RIBE scheme RIBE.ct and the ciphertext of the HIBE scheme HIBE.ct. To decrypt RIBE.ct in SR-IBE Σ, the decryption is implemented from the top to the bottom along the path in the tree. The server will decrypt the upper non-leaf part while the user will decrypt the leaf part. Meanwhile, the user is alway able to use $\mathsf{HIBE.sk_{id}}$ and time slot t to compute $\mathsf{HIBE.sk_{id||t}}$ and decrypt HIBE.ct with it. The process is shown in Figure 8.

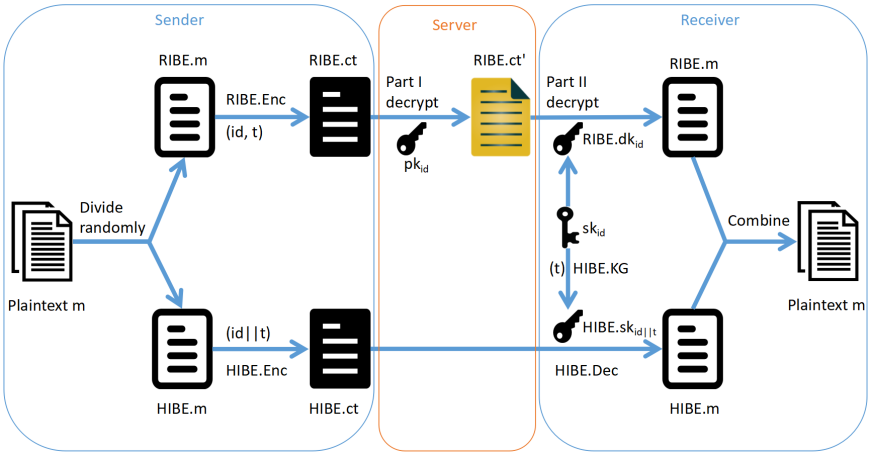

Figure 8. The process of the SR-IBE scheme.

Construction of SR-IBE. Now we describe our server-aided revocable IBE scheme $\Sigma = (\mathsf{Setup}, \mathsf{PubKG}, \mathsf{KU}, \mathsf{TranKG}, \mathsf{PrivKG}, \mathsf{DK}, \mathsf{Enc}, \mathsf{Transform}, \mathsf{Dec}, \mathsf{R})$.

- $\mathsf{Setup}(1^\lambda, 1^n)$: given a security parameter λ, an integer n where 2^n is the maximal number of users that the scheme supports. Define identity space as $\Sigma.\mathcal{ID} = \{0,1\}^n$ and time space as $\Sigma.\mathcal{T} = \{0,1\}^\ell$, and do the following.

  1. Sample $s \xleftarrow{\$} \{0,1\}^\lambda$.
  2. For each $i \in [n]$, invoke $(k_i, td_i) \xleftarrow{\$} \mathsf{HGen}(1^\lambda, 2\lambda)$.
  3. Initialize key list $\mathsf{KL} := \emptyset$, public list $\mathsf{PL} = \emptyset$, key update list $\mathsf{KU} = \emptyset$ and revocation list $\mathsf{RL} := \emptyset$.
  4. Run $(\mathsf{HIBE.mpk}, \mathsf{HIBE.msk}) \leftarrow \mathsf{HIBE.Setup}(1^\lambda)$.
  5. $\mathsf{mpk} := (k_0, \cdots, k_{n-1}, \ell, \mathsf{HIBE.mpk})$; $\mathsf{st} := \{\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU}\}$; $\mathsf{msk} := (\mathsf{mpk}, td_0, \cdots, td_{n-1}, s, \mathsf{HIBE.msk})$.
  6. Output $(\mathsf{mpk}, \mathsf{msk}, \mathsf{st})$.

- $\mathsf{PubKG}(\mathsf{msk}, \mathsf{id} \in \{0,1\}^n, \mathsf{st})$

  1. Parse $\mathsf{msk} = (\mathsf{mpk}, td_0, \cdots, td_{n-1}, s, \mathsf{HIBE.msk})$, $\mathsf{st} = \{\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU}\}$ and $\mathsf{mpk} = (k_0, \cdots, k_{n-1}, \ell, \mathsf{HIBE.mpk})$.
  2. $\mathsf{W} := \{\varepsilon, \mathsf{id}[1], \cdots, \mathsf{id}[1 \cdots n-1]\}$, where ε is the empty string.
  3. For all $v \in \mathsf{W} \setminus \{\mathsf{id}[1 \cdots n-1]\}$:
     $(h_v, h_{v||0}, h_{v||1}, r_v) \leftarrow \mathsf{NodeGen}((k_0, \cdots, k_{n-1}), (td_0, \cdots, td_{n-1}, s), v, \ell)$,
     $\mathsf{KL} := \mathsf{KL} \cup \{(v, h_v, h_{v||0}, h_{v||1}, r_v)\}$,
     $\mathsf{lk}_v := (h_v, h_{v||0}, h_{v||1}, r_v)$.
  4. For $v = \mathsf{id}[1 \cdots n-1]$:
     $(h_v, h_{v||0} = \mathsf{ek}_{v||0}, h_{v||1} = \mathsf{ek}_{v||1}, r_v, \mathsf{dk}_{v||0}, \mathsf{dk}_{v||1}) \leftarrow \mathsf{LeafGen}(k_{n-1}, (td_{n-1}, s), v, \ell)$,

$\mathsf{KL} := \mathsf{KL} \cup \{(v, h_v, \mathsf{ek}_{v||0}, \mathsf{ek}_{v||1}, r_v), (v||0, \mathsf{ek}_{v||0}, \perp), (v||1, \mathsf{ek}_{v||1}, \perp)\}$,
$\mathsf{lk}_v := (h_v, \mathsf{ek}_{v||0}, \mathsf{ek}_{v||1}, r_v)$.

5. $\mathsf{st} = \{\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU}\}$ and $\mathsf{pk}_{\mathsf{id}} := (\mathsf{t} = 0, \mathsf{id}, \{\mathsf{lk}_v\}_{v \in \mathsf{W}})$.
6. Output $(\mathsf{pk}_{\mathsf{id}}, \mathsf{st})$.
   // This algorithm is almost the same as the Private Key Generation algorithm in Section 4 except that there is no $\mathsf{dk}_{\mathsf{id}}$ in $\mathsf{pk}_{\mathsf{id}}$.

- $\mathsf{KU}(\mathsf{msk}, \mathsf{t}, \mathsf{st})$:

1. Parse $\mathsf{msk} = (\mathsf{mpk}, td_0, \cdots, td_{n-1}, s, \mathsf{HIBE.msk})$, $\mathsf{st} = \{\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU}\}$ and $\mathsf{mpk} = (k_0, \cdots, k_{n-1}, \ell, \mathsf{HIBE.mpk})$.
2. $\mathsf{Y} \leftarrow \mathsf{FindNodes}(\mathsf{RL}, \mathsf{t}, \mathsf{KL})$. // Y stores all revoked leaves and their ancestors
3. If $\mathsf{Y} = \emptyset$, Output(KU, PL)
4. Set key update list $\mathsf{KU}^{(\mathsf{t})} := \emptyset$.
5. For all node $v \in \mathsf{Y}$ such that $|v| = n$: $(\mathsf{ek}_v^{(\mathsf{t})}, \perp) \leftarrow \mathsf{LeafChange}(v, \mathsf{t}, s)$,
   $\mathsf{KU}^{(\mathsf{t})} := \mathsf{KU}^{(\mathsf{t})} \cup \{(v, \mathsf{ek}_v^{(\mathsf{t})}, \perp)\}$. $h_v^{(\mathsf{t})} := \mathsf{ek}_v^{(\mathsf{t})}$.
6. For $i = n-1$ to 0: For all node $v \in \mathsf{Y}$ and $|v| = i$:
   Set $j := \mathsf{t}$, $\mathsf{KU}^{(0)} := \mathsf{KL}$.
   While$(j \geq 0)$
   If $\exists v||b$ s.t. $(v||b, h_{v||b}, \cdot, \cdot, \cdot) \in \mathsf{KU}^{(j)}$,
   $h_{v||b}^{(\mathsf{t})} := h_{v||b}$,
   Break;
   $j := j - 1$.
   $(h_v^{(\mathsf{t})}, h_{v||0}^{(\mathsf{t})}, h_{v||1}^{(\mathsf{t})}, r_v^{(\mathsf{t})}) \leftarrow \mathsf{NodeChange}(k_i, td_i, v, h_{v||0}^{(\mathsf{t})}, h_{v||1}^{(\mathsf{t})}, \mathsf{t}, s)$.
   $\mathsf{KU}^{(\mathsf{t})} := \mathsf{KU}^{(\mathsf{t})} \cup \{(v, h_v^{(\mathsf{t})}, h_{v||0}^{(\mathsf{t})}, h_{v||1}^{(\mathsf{t})}, r_v^{(\mathsf{t})})\}$.
7. $\mathsf{KU} := \mathsf{KU} \cup \{(\mathsf{t}, \mathsf{KU}^{(\mathsf{t})})\}$ and $\mathsf{PL} := \mathsf{PL} \cup \{(\mathsf{t}, h_\varepsilon^{(\mathsf{t})})\}$.
8. $\mathsf{st} := \{\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU}\}$
9. Output (st).
   // This algorithm is identical to the Key Update Generation algorithm in Section 4.

- $\mathsf{TranKG}(\mathsf{mpk}, \mathsf{pk}_{\mathsf{id}}, \mathsf{KU})$:

1. $\mathsf{W} := \{\varepsilon, \mathsf{id}[1], \cdots, \mathsf{id}[1 \cdots n-1]\}$, where ε is the empty string.
2. Parse $\mathsf{mpk} = (k_0, \cdots, k_{n-1}, \ell, \mathsf{HIBE.mpk})$ and $\mathsf{pk}_{\mathsf{id}} = (0, \mathsf{id}, \{h_v, h_{v||0}, h_{v||1}, r_v\}_{v \in \mathsf{W}})$.
3. From KU retrieve a set $\Omega := \{(\tilde{\mathsf{t}}, \mathsf{KU}^{(\tilde{\mathsf{t}})}) \mid (\tilde{\mathsf{t}}, \mathsf{KU}^{(\tilde{\mathsf{t}})}) \in \mathsf{KU}, 0 \leq \tilde{\mathsf{t}} < \mathsf{t}\}$.
4. For each $(\tilde{\mathsf{t}}, \mathsf{KU}^{(\tilde{\mathsf{t}})}) \in \Omega$ with $\tilde{\mathsf{t}}$ in ascending order, does the following:
   For $i = 0$ to $n-1$:
   $v := \mathsf{id}[1 \cdots i]$ (Recall $\mathsf{id}[1 \cdots 0] = \varepsilon$).
   If $\exists (v, h_v^{(\tilde{\mathsf{t}})}, h_{v||0}^{(\tilde{\mathsf{t}})}, h_{v||1}^{(\tilde{\mathsf{t}})}, r_v^{(\tilde{\mathsf{t}})}) \in \mathsf{KU}^{(\tilde{\mathsf{t}})}$:
   $\mathsf{lk}_v^{(\mathsf{t})} := (h_v^{(\tilde{\mathsf{t}})}, h_{v||0}^{(\tilde{\mathsf{t}})}, h_{v||1}^{(\tilde{\mathsf{t}})}, r_v^{(\tilde{\mathsf{t}})})$.
5. Output $\mathsf{tk}_{\mathsf{id}}^{(\mathsf{t})} := (\mathsf{t}, \mathsf{id}, \{\mathsf{lk}_v^{(\mathsf{t})}\}_{v \in \mathsf{W}})$
   // This algorithm is almost the same as the Decryption Key Generation algorithm in Section 4 except that all update operations do not involve leaf nodes, i.e., $\mathsf{dk}_{\mathsf{id}}$.

- $\mathsf{PrivKG}(\mathsf{msk}, \mathsf{id} \in \{0,1\}^n)$

1. Parse $\mathsf{msk} = (\mathsf{mpk}, td_0, \cdots, td_{n-1}, s, \mathsf{HIBE.msk})$ and $\mathsf{mpk} = (k_0, \cdots, k_{n-1}, \ell, \mathsf{HIBE.mpk})$.
2. $(\mathsf{ek}_{\mathsf{id}}, \mathsf{dk}_{\mathsf{id}}) \leftarrow \mathsf{G}(1^\lambda, \mathsf{PRF}(s, 0^\ell||\mathsf{id}))$
3. Run $\mathsf{HIBE.sk}_{\mathsf{id}} \leftarrow \mathsf{HIBE.KG}(\mathsf{HIBE.msk}, \mathsf{id})$.

4. $\mathsf{sk}_{\mathsf{id}} := (\mathsf{dk}_{\mathsf{id}}, \mathsf{HIBE.sk}_{\mathsf{id}})$.
5. Output $\mathsf{sk}_{\mathsf{id}}$.

- $\mathsf{DK}(\mathsf{sk}_{\mathsf{id}}, \mathsf{t})$

1. Parse $\mathsf{sk}_{\mathsf{id}} = (\mathsf{dk}_{\mathsf{id}}, \mathsf{HIBE.sk}_{\mathsf{id}})$.
2. Run $\mathsf{HIBE.sk}_{\mathsf{id}||\mathsf{t}} \leftarrow \mathsf{HIBE.KG}(\mathsf{HIBE.sk}_{\mathsf{id}}, \mathsf{t})$.
3. $\mathsf{dk}_{\mathsf{id}}^{(\mathsf{t})} := (\mathsf{dk}_{\mathsf{id}}, \mathsf{HIBE.sk}_{\mathsf{id}||\mathsf{t}})$.
4. Output $\mathsf{dk}_{\mathsf{id}}^{(\mathsf{t})}$.

- $\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{t}, m, \mathsf{PL}))$:
Same to the encryption algorithm in Section 4, we use these two circuits that will be garbled during the encryption procedure.

– $\mathsf{Q}[m](ek)$: Compute and output $\mathsf{E}(ek, m)$.
– $\mathsf{P}[\beta \in \{0,1\}, k, \overline{\mathsf{lab}}](h)$: Compute and output $\{\mathsf{HEnc}(k, (h, j+\beta\cdot\lambda, b), \mathsf{lab}_{j,b})\}_{j\in[\lambda], b\in\{0,1\}}$, where $\overline{\mathsf{lab}}$ is the short for $\{\mathsf{lab}_{j,b}\}_{j\in[\lambda],b\in\{0,1\}}$.

Encryption proceeds as follows:

1. Retrieve the last item $(\bar{\mathsf{t}}, h_{\varepsilon}^{(\bar{\mathsf{t}})})$ from PL. If $\mathsf{t} < \bar{\mathsf{t}}$, output $\perp$; otherwise $h_{\varepsilon}^{(\mathsf{t})} := h_{\varepsilon}^{(\bar{\mathsf{t}})}$.
2. Parse $\mathsf{mpk} = (k_0, \cdots, k_{n-1}, \ell, \mathsf{HIBE.mpk})$.
3. Sample a pair $(\mathsf{RIBE}.m, \mathsf{HIBE}.m) \in \mathcal{M}^2$ uniformly at random, subject to $\mathsf{RIBE}.m + \mathsf{HIBE}.m = m$.
4. Run $\mathsf{HIBE.ct} \leftarrow \mathsf{HIBE.Enc}(\mathsf{HIBE.mpk}, (\mathsf{id}||\mathsf{t}), \mathsf{HIBE}.m)$.
5. $(\tilde{Q}, \overline{\mathsf{lab}}) \xleftarrow{\$} \mathsf{GCircuit}(1^\lambda, \mathsf{Q}[\mathsf{RIBE}.m])$.
6. For $i = n-1$ to 0,
$(\tilde{P}^i, \overline{\mathsf{lab}}') \xleftarrow{\$} \mathsf{GCircuit}(1^\lambda, \mathsf{P}[\mathsf{id}[i+1], k_i, \overline{\mathsf{lab}}])$ and set $\overline{\mathsf{lab}} := \overline{\mathsf{lab}}'$.
7. Output $\mathsf{RIBE.ct} := \left(\left\{\mathsf{lab}_{j,h_{\varepsilon,j}^{(\mathsf{t})}}\right\}_{j\in[\lambda]}, \{\tilde{P}^0, \cdots, \tilde{P}^{n-1}, \tilde{Q}\}\right)$, where $h_{\varepsilon,j}^{(\mathsf{t})}$ is the j^{th} bit of $h_{\varepsilon}^{(\mathsf{t})}$.
8. Output $\mathsf{ct} := (\mathsf{RIBE.ct}, \mathsf{HIBE.ct})$.

- $\mathsf{Transform}(\mathsf{mpk}, \mathsf{tk}_{\mathsf{id}}^{(\mathsf{t})}, \mathsf{ct})$

1. $\mathsf{W} := \{\varepsilon, \mathsf{id}[1], \cdots, \mathsf{id}[1\cdots n-1]\}$, where ε is the empty string.
2. Parse $\mathsf{mpk} = (k_0, \cdots, k_{n-1}, \ell, \mathsf{HIBE.mpk})$, $\mathsf{ct} = (\mathsf{RIBE.ct}, \mathsf{HIBE.ct})$ and $\mathsf{sk}_{\mathsf{id}}^{(\mathsf{t})} = (\mathsf{t}, \mathsf{id}, \{\mathsf{lk}_v^{(\mathsf{t})}\}_{v\in\mathsf{W}})$, where $\mathsf{lk}_v^{(\mathsf{t})} = (h_v^{(\mathsf{t})}, h_{v||0}^{(\mathsf{t})}, h_{v||1}^{(\mathsf{t})}, r_v^{(\mathsf{t})})$.
3. Parse $\mathsf{RIBE.ct} := \left(\left\{\mathsf{lab}_{j,h_{\varepsilon,j}^{(\mathsf{t})}}\right\}_{j\in[\lambda]}, \{\tilde{P}^0, \cdots, \tilde{P}^{n-1}, \tilde{Q}\}\right)$
4. Set $y := h_{\varepsilon}^{(\mathsf{t})}$.
5. For $i = 0$ to $n-1$:
Set $v := \mathsf{id}[1\cdots i]$ (Recall $\mathsf{id}[1\cdots 0] = \varepsilon$);
$\{c_{j,b}\}_{j\in[\lambda],b\in\{0,1\}} \leftarrow \mathsf{Eval}(\tilde{P}^i, \{\mathsf{lab}_{j,y_j}\}_{j\in[\lambda]})$;
If $i \neq n-1$, set $v' := \mathsf{id}[1\cdots i+1]$ and $y := h_{v'}^{(\mathsf{t})}$, and for each $j \in [\lambda]$,

$$\{\mathsf{lab}_{j,y_j}\}_{j\in[\lambda]} \leftarrow \mathsf{HDec}(k_i, c_{j,y_j}, (h_{v||0}^{(\mathsf{t})}||h_{v||1}^{(\mathsf{t})}), r_v^{(\mathsf{t})}).$$

If $i = n-1$, set $y := \mathsf{ek}_{\mathsf{id}}$ and for each $j \in [\lambda]$, compute

$$\{\mathsf{lab}_{j,y_j}\}_{j\in[\lambda]} \leftarrow \mathsf{HDec}(k_i, c_{j,y_j}, (\mathsf{ek}_{v||0}||\mathsf{ek}_{v||1}) = (h_{v||0}^{(\mathsf{t})}||h_{v||1}^{(\mathsf{t})}), r_v^{(\mathsf{t})}).$$

6. Compute $f \leftarrow \mathsf{Eval}(\tilde{Q}, \{\mathsf{lab}_{j,y_j}\}_{j\in[\lambda]})$.
7. Output $\mathsf{ct}' := (f, \mathsf{HIBE.ct})$
   // This algorithm is almost the same as the Decryption algorithm in Section 4 except that this algorithm omits the last step, i.e., it does not recover RIBE.m from f.

- $\mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_{\mathsf{id}}^{(\mathsf{t})}, \mathsf{ct}')$

1. Parse $\mathsf{mpk} = (k_0, \cdots, k_{n-1}, \ell, \mathsf{HIBE.mpk})$, $\mathsf{dk}_{\mathsf{id}}^{(\mathsf{t})} = (\mathsf{dk}_{\mathsf{id}}, \mathsf{HIBE.sk}_{\mathsf{id}||\mathsf{t}})$ and $\mathsf{ct}' = (f, \mathsf{HIBE.ct})$.
2. Run $\mathsf{HIBE.m} \leftarrow \mathsf{HIBE.Dec}(\mathsf{HIBE.mpk}, \mathsf{HIBE.sk}_{\mathsf{id}||\mathsf{t}}, \mathsf{HIBE}.ct,)$.
3. Run $\mathsf{RIBE.m} \leftarrow \mathsf{D}(\mathsf{dk}_{\mathsf{id}}, f)$.
4. Output $m := \mathsf{RIBE.m} + \mathsf{HIBE.m}$

- $\mathsf{R}(\mathsf{id}, \mathsf{t}, \mathsf{st})$:

1. Parse $\mathsf{st} := \{\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU}\}$.
2. Update the revocation list by $\mathsf{RL} := \mathsf{RL} \cup \{(\mathsf{id}, \mathsf{t})\}$.
3. $\mathsf{st} := \{\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU}\}$.
4. Output st.

Obviously, the correctness of this scheme Σ follows from the correctness of the RIBE scheme described in Section 4 and the HIBE scheme used as the building block. The security of scheme Σ is guaranteed by the following theorem.

Theorem 3. *If HIBE is the hierarchal IBE constructed in [8], the above server-aided revocable IBE scheme Σ is selective-SR-ID-CPA secure (with decryption key exposure resistance) based on the CDH assumption.*

Proof. The full proof of Theorem 3 is in Appendix B.2. □

7. Analysis of Key Updating Size

In this section, we analyze the key updating efficiency of our revocable IBE scheme. Different from an IBE scheme, a revocable IBE scheme has enormous cost on the publishing updating keys at each time slot. In our RIBE, the number of updating keys is linear to the number of updated nodes. Therefore, we focus on the number of updated nodes for the performance. The advantage of our RIBE lies in the fact that the nodes that needs to updated is only related to the number Δr of newly revoked users in the past time slot. More precisely, in all the three schemes proposed in this paper, the number of nodes needs to be updated in each time plot is at most $\Delta r(\log N - \log(\Delta r))$. Thus the key updating size of our scheme is at most $O(\Delta r(\log N - \log(\Delta r)))$. If there is no new users revoked in the previous time slot, then key updating is not necessary at all.

Recall that in the most of RIBE schemes, the size of updating keys is closely related to the total number r of *all* the revoked users across *all* the past slots. For example, in [10] the size of updated key during each time slot is of order $O(r \log(N/r))$, where N is the number of users. In addition, in [14], the size of updated key during each time slot is of order $O(r)$.

For simulation, we use Poisson distribution to simulate the number of revoked users at each time period, where α denotes the expected number of revoked users in each time slot. At a time slot t, we sample a random number Δr_{t} following the Poisson distribution parameterized by α, and Δr_{t} denotes the number of revoked users at time slot t. The total number $r_{\mathsf{t}'}$ of the revoked users up to time slot t' is given by $\sum_{\mathsf{t}=0}^{\mathsf{t}'} \Delta r_{\mathsf{t}}$. We evaluate the key updating sizes in our RIBE, the RIBE in [10] and the RIBE in [14]. Since all the our three schemes share the same updating complexity in each time plot, we only simulate our RIBE scheme without DKER and compare the results with the RIBE scheme in [10] and the RIBE scheme in [14]. The simulation results for $N = 2^{20}$ and $N = 2^{25}$ are shown in Figures 9 and 10 respectively.

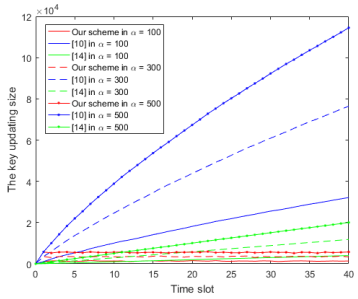

Figure 9. $N = 20$.

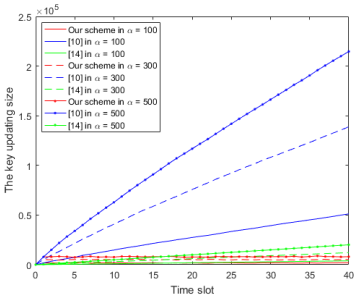

Figure 10. $N = 25$.

Author Contributions: Conceptualization, Z.H.; Methodology, Z.H. and S.L.; Simulation, K.C. and Z.H.; Validation, J.K.L.; Formal Analysis, Z.H. and S.L.; Investigation, K.C.and J.K.L.; Writing—Original Draft Preparation, Z.H. and S.L.; Writing—Review & Editing, K.C.and J.K.L.; Visualization, K.C.; Supervision, J.K.L.; Project Administration, S.L.; Funding Acquisition, S.L. and K.C.

Funding: Ziyuan Hu and Shengli Liu were supported by the National Natural Science Foundation of China (NSFC Grant No. 61672346). Kefei Chen was supported by National Key R&D Program of China (Grant No. 2017YFB0802000), NSFC (Grant No. U1705264) and (Grant No. 61472114).

Acknowledgments: The authors thank the anonymous reviewers for their helpful comments. Special thanks go to Atsushi Takayasu who helped us to give a better presentation of this paper and told us their work of converting a RIBE scheme without DKER to a RIBE scheme with DKER.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Hierarchical Identity Based Encryption

We formally define Hierarchical Identity Based Encryption (HIBE). A HIBE scheme consists of four PPT algorithms $\mathsf{HIBE} = (\mathsf{HIBE.Setup}, \mathsf{HIBE.KG}, \mathsf{HIBE.Enc}, \mathsf{HIBE.Dec})$. Let $\mathsf{HIBE}.\mathcal{M}$ denote the message space and $\mathsf{HIBE}.\mathcal{ID}$ the identity space.

- **Setup**: The setup algorithm $\mathsf{HIBE.Setup}$ is run by the key authority. The input of the algorithm is a security parameter λ. The output of this algorithm consists of a pair of key $(\mathsf{HIBE.mpk}, \mathsf{HIBE.msk})$. In formula, $(\mathsf{HIBE.mpk}, \mathsf{HIBE.msk}) \leftarrow \mathsf{HIBE.Setup}(1^\lambda)$.
- **Key Generation**: The key generation algorithm $\mathsf{HIBE.KG}$ is run by the key authority. It takes a secret key $\mathsf{HIBE.sk_{id}}$ ($\mathsf{HIBE.msk}$ for ϵ) and an identity id' as the input. The output of this algorithm is $\mathsf{HIBE.sk_{id}}$. In formula, $\mathsf{HIBE.sk_{id||id'}} \leftarrow \mathsf{HIBE.KG}(\mathsf{HIBE.sk_{id}}, \mathsf{id}')$.
- **Encryption**: The encryption algorithm $\mathsf{HIBE.Enc}$ is run by the sender. It takes the master public key $\mathsf{HIBE.mpk}$, an identity id and a plaintext message m as the input. The output of this algorithm is the ciphertext $\mathsf{HIBE.ct}$. In formula, $\mathsf{HIBE.ct} \leftarrow \mathsf{HIBE.Enc}(\mathsf{HIBE.mpk}, \mathsf{id}, \mathsf{m})$.
- **Decryption**: The decryption algorithm $\mathsf{HIBE.Dec}$ is run by the receiver. The input of this algorithm consists of the master public key $\mathsf{HIBE.mpk}$, the secret key $\mathsf{HIBE.sk_{id}}$ and the ciphertext $\mathsf{HIBE.ct}$. The output of this algorithm is the plaintext m. In formula, $\mathsf{m} \leftarrow \mathsf{HIBE.Dec}(\mathsf{HIBE.mpk}, \mathsf{HIBE.sk_{id}}, \mathsf{HIBE.ct})$.

Correctness. For all $(\mathsf{HIBE.mpk}, \mathsf{HIBE.msk}) \leftarrow \mathsf{HIBE.Setup}(1^\lambda)$, all $(\mathsf{id}, \mathsf{id}') \in (\mathsf{HIBE}.\mathcal{ID})^2$, $\mathsf{HIBE.sk_{id}} \leftarrow \mathsf{HIBE.KG}(\mathsf{HIBE.msk}, \mathsf{id})$, $\mathsf{HIBE.sk_{id||id'}} \leftarrow \mathsf{HIBE.KG}(\mathsf{HIBE.sk_{id}}, \mathsf{id}')$, all $\mathsf{m} \in \mathsf{HIBE}.\mathcal{M}$ and $\mathsf{HIBE.ct} \leftarrow \mathsf{HIBE.Enc}(\mathsf{HIBE.mpk}, \mathsf{id||id}', \mathsf{m})$, it holds that $\mathsf{m} \leftarrow \mathsf{HIBE.Dec}(\mathsf{HIBE.mpk}, \mathsf{HIBE.sk_{id||id'}}, \mathsf{HIBE.ct})$.

Security. Now we formalize the security of a revocable IBE. We first consider a oracle key generation oracle $\mathrm{HIBE.KG}(\cdot)$. This oracle takes an identity id as the input and outputs $\mathsf{HIBE.sk_{id}} \leftarrow \mathsf{HIBE.KG}(\mathsf{HIBE.msk}, \mathsf{id})$.

Definition 5. *Let* $\mathit{HIBE} = (\mathit{HIBE.Setup}, \mathit{HIBE.KG}, \mathit{HIBE.Enc}, \mathit{HIBE.Dec})$ *be a hierarchical IBE scheme. Below describes an experiment between a challenger* $\mathcal{C}$ *and a PPT adversary* $\mathcal{A}$*.*

$\boldsymbol{EXP}_{\mathcal{A}}^{\textit{selective-IND-ID-CPA}}(\lambda)$:
$(m_0, m_1, id^*) \leftarrow \mathcal{A}$;
$(\mathit{HIBE.mpk}, \mathit{HIBE.msk}) \leftarrow \mathit{HIBE.Setup}(1^\lambda)$;
$\theta \xleftarrow{\$} \{0,1\}$;
$\mathit{HIBE.ct}^* \leftarrow \mathit{HIBE.Enc}(\mathit{HIBE.mpk}, id^*, m_\theta)$
$\theta' \leftarrow \mathcal{A}^{\mathrm{HIBE.KG}(\cdot)}(\mathit{HIBE.mpk}, \mathit{HIBE.ct}^*)$;
If $\theta = \theta'$ *Return* 1; *If* $\theta \neq \theta'$ *Return* 0.

The experiment has the following requirements for $\mathcal{A}$*.*

- *The two plaintexts submitted by* $\mathcal{A}$ *have the same length, i.e.,* $|m_0| = |m_1|$.
- *If* $\mathcal{A}$ *has queried* id *to oracle* $\mathrm{HIBE.KG}(\cdot)$*, then* id *cannot be the a prefix of* id^*.

A hierarchical IBE scheme is selective-IND-ID-CPA secure if for all PPT adversary $\mathcal{A}$*, the following advantage is negligible in the security parameter* λ*, i.e.,*

$$\mathbf{Adv}_{\mathit{HIBE},\mathcal{A}}^{\textit{selective-IND-ID-CPA}}(\lambda) = |\Pr[\mathbf{EXP}_{\mathcal{A}}^{\textit{selective-IND-ID-CPA}}(\lambda) = 1] - 1/2| = \mathit{negl}(\lambda).$$

Appendix B. Proofs of Theorems

Appendix B.1. Proof of Theorem 1

Proof. The proof consists of $5n + 4$ hybrids, $\mathcal{H}_{-1}$, $\{\mathcal{H}_0, \mathcal{H}_{0,1}, \mathcal{H}_{0,2}, \mathcal{H}_{0,3}, \mathcal{H}_{0,4}\}$, $\cdots$, $\{\mathcal{H}_{n-1}, \mathcal{H}_{n-1,1}, \mathcal{H}_{n-1,2}, \mathcal{H}_{n-1,3}, \mathcal{H}_{n-1,4}\}$, $\mathcal{H}_n, \mathcal{H}_{n+1}, \mathcal{H}_{n+2}$, and we will show that adjacent hybrids are computational indistinguishable. Compared with $\mathcal{H}_i$, hybrid $\mathcal{H}_{i+1}$ has small changes in how the oracle queries are answered and/or the challenge ciphertext is generated. Denote by $\mathcal{H}_i \Rightarrow 1$ the event that the hybrid outputs 1.

- $\mathcal{H}_{-1}$: This hybrid is just the original experiment $\mathbf{EXP}_{\mathcal{A}}^{\text{IND-ID-CPA}}(\lambda)$ (without oracle $\mathrm{DK}(\cdot,\cdot)$) as shown in Definition 1. Thus

$$\mathbf{Adv}_{\mathcal{A}}^{\text{IND-ID-CPA}}(\lambda) = |\Pr[\mathcal{H}_{-1} \Rightarrow 1] - 1/2]|. \tag{A1}$$

  Specifically, in this hybrid, the challenger $\mathcal{C}$ will first invoke $\mathsf{RIBE.Setup}(1^\lambda, 1^n)$ to obtain $(\mathsf{mpk}, \mathsf{msk}, \mathsf{st})$ where $\mathsf{st} = (\mathsf{KL}, \mathsf{PL}, \mathsf{RL})$. $\mathcal{C}$ sends mpk to the adversary $\mathcal{A}$ and answers oracle queries as follows.

  1. Private key generation oracle $\mathrm{KG}(\mathsf{id})$. Upon receiving $\mathcal{A}$'s query id, the challenger invokes $(\mathsf{sk_{id}}, \mathsf{st}') \leftarrow \mathsf{RIBE.KG}(\mathsf{msk}, \mathsf{id}, \mathsf{KL}, \mathsf{st})$ and returns $\mathsf{sk_{id}}$ to $\mathcal{A}$.
  2. Key update oracle $\mathrm{KU}(\mathsf{t})$. Upon receiving $\mathcal{A}$'s query t, $\mathcal{C}$ does as follows:
     For t_i from 1 to t:
       $\mathsf{st}' \leftarrow \mathsf{RIBE.KU}(\mathsf{msk}, \mathsf{t}_i, \mathsf{st})$
       $\mathsf{st} := \mathsf{st}'$.
     Parse $\mathsf{st} = (\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU})$
     Returns $(\mathsf{KU}, \mathsf{PL})$ to $\mathcal{A}$.

3. Revocation oracle $\text{RVK}(\mathsf{id}, \mathsf{t})$. Upon receiving $\mathcal{A}$'s query an id and a t, $\mathcal{C}$ invokes $\mathsf{st}' \leftarrow \mathsf{RIBE.R}(\mathsf{id}, \mathsf{t}, \mathsf{st})$ and parses $\mathsf{st}' = (\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU})$. It returns RL to $\mathcal{A}$.

- $\mathcal{H}_0$: In this hybrid, we change how the challenger $\mathcal{C}$ answers oracle $\text{KG}(\mathsf{id})$ and oracle $\text{KU}(\mathsf{t})$. Recall that in $\mathcal{H}_{-1}$, the subroutines NodeGen and LeafGen are involved in RIBE.KG when answering queries to the oracle KG, and the subroutines NodeChange and LeafChange are involved in RIBE.KU when answering queries to the oracle KU. A pseudo-random subroutine $\mathsf{PRF}(s, \cdot)$ is invoked in all the four subroutines. Now in $\mathcal{H}_0$, this $\mathsf{PRF}(s, \cdot)$ will be replaces by a truly subroutine $\mathsf{RF}(\cdot)$. Note that $\mathcal{C}$ can efficiently implement the truly subroutine $\mathsf{RF}(\cdot)$: Given a fresh input x, $\mathcal{C}$ chooses a random element R in $\{0,1\}^\lambda$ as the output of $\mathsf{RF}(\cdot)$. $\mathcal{C}$ records (x, R) locally. If x is not fresh, $\mathcal{C}$ retrieves R from its records.

  Any difference between $\mathcal{H}_{-1}$ and $\mathcal{H}_0$ will lead to a distinguisher $\mathcal{B}_1$, who can distinguish PRF from RF. Hence

$$|\Pr[\mathcal{H}_{-1} \Rightarrow 1] - \Pr[\mathcal{H}_0 \Rightarrow 1]| \leq \mathbf{Adv}^{PRF}_{\mathcal{B}_1}(\lambda). \tag{A2}$$

- $\mathcal{H}_\gamma$ for $\gamma \in \{0, 1, \cdots, n\}$: In this hybrid, challenger $\mathcal{C}$ changes the generation of the challenge ciphertext. Recall that the challenge ciphertext $\mathsf{ct} := \left(\left\{\mathsf{lab}_{j,h^{(\mathsf{t})}_{\varepsilon,j}}\right\}_{j\in[\lambda]}, \{\tilde{P}^0, \cdots, \tilde{P}^{n-1}, \tilde{Q}\}\right)$. In hybrid $\mathcal{H}_\gamma$, $\mathcal{C}$ invokes the simulator Sim provided by the garbled circuit scheme to generate the first γ garbled circuits $\{\tilde{P}^0, \cdots, \tilde{P}^{\gamma-1}\}$. Meanwhile, for $i \in [0, \gamma-1]$, the input of the Sim is the 2λ chameleon encryption ciphertexts $\{\mathsf{HEnc}(k_i, (h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots i]}, j, b), \mathsf{lab}_{j,h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots i+1],j}})\}_{j\in[\lambda],b\in\{0,1\}}$. We stress that the 2λ labels satisfy $\mathsf{lab}_{j,0} = \mathsf{lab}_{j,1} = \mathsf{lab}_{j,h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots i+1],j}}$. Please note that Sim needs the hash value $h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots i]}$ with $i \in [0, \gamma]$. Therefore the challenger $\mathcal{C}$ has to determine $h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots i]}$ first with $i \in [0, \gamma]$. Then $\mathcal{C}$ invokes sim to generate $\{\tilde{P}^0, \cdots, \tilde{P}^{\gamma-1}\}$, and invokes GCircuit to generate the rest circuits. Below is the detailed description of the generation of the challenge ciphertext by $\mathcal{C}$.
  Assume $\mathcal{A}$'s challenge query as $(\mathsf{id}^*, \mathsf{t}^*, M_0, M_1)$. $\mathcal{C}$ first chooses a random bit θ, and encrypts M_θ under id^* in t^* as follows:

  1. Define $\mathsf{W}^* := \{\varepsilon, \mathsf{id}^*[1], \cdots, \mathsf{id}^*[1\cdots n-1]\}$, where ε is the empty string. Determine the values $(h_\varepsilon, h_{\mathsf{id}^*[1]}, \cdots, h_{\mathsf{id}^*[1\cdots n-1]}, \mathsf{ek}_{\mathsf{id}^*})$ which are the values attached to all nodes on the path from the root to id^*.
     - $\mathsf{sk}_{\mathsf{id}^*} \leftarrow \mathsf{RIBE.KG}(\mathsf{msk}, \mathsf{id}^*)$.
     - $\mathsf{st}' \leftarrow \mathsf{RIBE.KU}(\mathsf{msk}, \mathsf{t}^*, \mathsf{st})$.
       $\mathsf{st} := \mathsf{st}' = (\mathsf{KL}, \mathsf{PL}, \mathsf{RL}, \mathsf{KU})$
       Retrieve $(\mathsf{t}^*, h^{(\mathsf{t}^*)}_\varepsilon)$ from PL.
     - $\mathsf{sk}^{(\mathsf{t}^*)}_{\mathsf{id}^*} \leftarrow \mathsf{RIBE.DK}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{id}^*}, \mathsf{KU}, \mathsf{t}^*)$.
       Parse $\mathsf{sk}^{(\mathsf{t}^*)}_{\mathsf{id}^*} = (\mathsf{id}^*, \{(h^{(\mathsf{t}^*)}_v, h^{(\mathsf{t}^*)}_{v||0}, h^{(\mathsf{t}^*)}_{v||1}, r^{(\mathsf{t}^*)}_v)\}_{v\in\mathsf{W}^*}, \mathsf{dk}_{\mathsf{id}^*})$, where $(h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots n-1]||0}, h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots n-1]||1}) = (\mathsf{ek}_{\mathsf{id}^*[1\cdots n-1]||0}, \mathsf{ek}_{\mathsf{id}^*[1\cdots n-1]||1})$. Please note that $\mathsf{dk}_{\mathsf{id}^*} = \perp$ if id^* has been revoked before t^*.
  2. $(\tilde{Q}, \overline{\mathsf{lab}}) \xleftarrow{\$} \mathsf{GCircuit}(1^\lambda, \mathsf{Q}[M_\theta])$, where $\overline{\mathsf{lab}} = \{\mathsf{lab}_{j,b}\}_{i\in[\lambda],b\in\{0,1\}}$.
  3. For $i = n-1$ to γ,

$$(\tilde{P}^i, \overline{\mathsf{lab}}') \xleftarrow{\$} \mathsf{GCircuit}(1^\lambda, \mathsf{P}[\mathsf{id}^*[i+1], k_i, \overline{\mathsf{lab}}]). \tag{A3}$$

     Set $\overline{\mathsf{lab}} := \overline{\mathsf{lab}}'$.

4. For $i = \gamma - 1$ to 0, set $v = \mathsf{id}^*[1 \cdots i]$, where $v = \varepsilon$ if $i = 0$.

$$(\tilde{P}^i, \{\mathsf{lab}'_{j,h_{v,j}^{(\mathsf{t}^*)}}\}_{j\in[\lambda]}) \xleftarrow{\$} \mathsf{Sim}(1^\lambda, \{\mathsf{HEnc}(k_i, (h_v^{(\mathsf{t}^*)}, j, b), \mathsf{lab}_{j,h^{(\mathsf{t}^*)}_{\mathbf{id}^*[1\cdots i+1],j}})\}_{j\in[\lambda],b\in\{0,1\}}) \tag{A4}$$

and set $\overline{\mathsf{lab}} := \{\mathsf{lab}'_{j,h_{v,j}^{(\mathsf{t}^*)}}, \mathsf{lab}'_{j,h_{v,j}^{(\mathsf{t}^*)}}\}_{j\in[\lambda]}$.

5. Output $\mathsf{ct}^* := \left(\left\{\mathsf{lab}_{\ell,h_{\varepsilon,\ell}^{(\mathsf{t}^*)}}\right\}_{\ell\in[\lambda]}, \{\tilde{P}^0, \cdots, \tilde{P}^{n-1}, \tilde{Q}\}\right)$, where $h_{\varepsilon,\ell}^{(\mathsf{t}^*)}$ is the ℓ^{th} bit of $h_\varepsilon^{(\mathsf{t}^*)}$.

Please note that the randomnesses in RIBE.KG and RIBE.KU are generated by random subroutines instead of the PRF.

- $\mathcal{H}_{\gamma,1}$ for $\gamma \in \{0, 1, \cdots, n-1\}$: This hybrid is the same as $\mathcal{H}_\gamma$ except that the challenger $\mathcal{C}$ changes the way of generating h_v and r_v when answering private key queries and the way of generating $h_v^{(\mathsf{t})}$ and $r_v^{(\mathsf{t})}$ when answering key update queries for $v \in \{0,1\}^\gamma$. For $\gamma = n-1$, set $(h_{v||0}, h_{v||1}) := (\mathsf{ek}_{v||0}, \mathsf{ek}_{v||1})$. In addition, set

$$h_{v||b}^{(\mathsf{t})} := \begin{cases} \mathsf{ek}_{v||b} & \text{if } v \text{ is not revoked,} \\ \mathsf{ek}'_{v||b} & \text{if } v \text{ is revoked,} \end{cases}$$

where $(\mathsf{ek}'_v, \mathsf{dk}'_v) \leftarrow \mathsf{G}(1^\lambda)$ and $b \in \{0,1\}$. Specifically, $\mathcal{C}$ changes the generation process of (h_v, r_v) from

$$\begin{aligned} h_v &:= \mathsf{H}(k_\gamma, 0^{2\lambda}; \mathsf{RF}(v)) \\ r_v &:= \mathsf{H}^{-1}(td_\gamma, (0^{2\lambda}, \mathsf{RF}(v)), h_{v||0}||h_{v||1}) \end{aligned}$$

to

$$\begin{aligned} r_v &\xleftarrow{\$} \mathbb{Z}_p \\ h_v &:= \mathsf{H}(k_\gamma, h_{v||0}||h_{v||1}; r_v) \end{aligned}$$

$\mathcal{C}$ uses the same way to generate $(h_v^{(\mathsf{t})}, r_v^{(\mathsf{t})})$, i.e., $r_v^{(\mathsf{t})}$ is chosen randomly, and $h_v^{(\mathsf{t})} := \mathsf{H}(k_\gamma, h_{v||0}^{(\mathsf{t})}||h_{v||1}^{(\mathsf{t})}; r_v^{(\mathsf{t})})$.

Due to uniformity properties of the chameleon hash, hybrids $\mathcal{H}_\gamma$ and $\mathcal{H}_{\gamma,1}$ are statistical indistinguishable. So

$$|\Pr[\mathcal{H}_\gamma \Rightarrow 1] - \Pr[\mathcal{H}_{\gamma,1} \Rightarrow 1]| = \mathsf{negl}. \tag{A5}$$

- $\mathcal{H}_{\gamma,2}$ for $\gamma \in \{0, 1, \cdots, n-1\}$: This hybrid is the same as $\mathcal{H}_{\gamma,1}$ except step 3 (as shown in $\mathcal{H}_\gamma$ in detail). Specifically, set $v := \mathsf{id}[0 \cdots \gamma]$. $\mathcal{C}$ changes the generation process of garbled circuits $\tilde{P}^\gamma$ from

$$(\tilde{P}^\gamma, \overline{\mathsf{lab}}') \xleftarrow{\$} \mathsf{GCircuit}(1^\lambda, \mathsf{P}[\mathsf{id}^*[\gamma+1], k_\gamma, \overline{\mathsf{lab}}])$$

and setting $\overline{\mathsf{lab}} := \overline{\mathsf{lab}}'$ to

$$(\tilde{P}^\gamma, \{\mathsf{lab}'_{j,h_{v,j}^{(\mathsf{t}^*)}}\}_{j\in[\lambda]}) \xleftarrow{\$} \mathsf{Sim}(1^\lambda, \{\mathsf{HEnc}(k_\gamma, (h_v^{(\mathsf{t}^*)}, j, b), \mathsf{lab}_{j,b})\}_{j\in[\lambda],b\in\{0,1\}})$$

and setting $\overline{\mathsf{lab}} := \{\mathsf{lab}'_{j,h_{v,j}^{(\mathsf{t}^*)}}, \mathsf{lab}'_{j,h_{v,j}^{(\mathsf{t}^*)}}\}_{j\in[\lambda]}$.

Since $\{\mathsf{HEnc}(k_\gamma, (h_v^{(\mathsf{t}^*)}, j, b), \mathsf{lab}_{j,b})\}_{j\in[\lambda],b\in\{0,1\}}$ is exactly the output of $\mathsf{P}[\mathsf{id}^*[\gamma+1], k_\gamma, \overline{\mathsf{lab}}](h_v^{(\mathsf{t}^*)})$, the indistinguishability of $\mathcal{H}_{\gamma,1}$ and $\mathcal{H}_{\gamma,2}$ directly follows from the security of the garbled circuit scheme. If there is a PPT adversary $\mathcal{A}$ who can distinguish $\mathcal{H}_{\gamma,1}$ and $\mathcal{H}_{\gamma,2}$ with advantage ϵ,

then we can construct a PPT algorithm $\mathcal{B}_2$ who can break the security of the garbled circuit scheme with same advantage ϵ. Please note that $\mathcal{B}_2$ can generate msk itself and simulate all the oracles for $\mathcal{A}$ perfectly. $\mathcal{B}_2$ embeds its own challenge to $(\tilde{P}^\gamma, \overline{\mathsf{lab}})$. If $(\tilde{P}^\gamma, \overline{\mathsf{lab}})$ is generated by GCircuit, $\mathcal{B}_2$ perfectly simulates $\mathcal{H}_{\gamma,1}$. If $(\tilde{P}^\gamma, \overline{\mathsf{lab}})$ is generated by Sim, $\mathcal{B}_2$ perfectly simulates $\mathcal{H}_{\gamma,2}$. Hence

$$|\Pr[\mathcal{H}_{\gamma,1} \Rightarrow 1] - \Pr[\mathcal{H}_{\gamma,2} \Rightarrow 1]| \leq \mathbf{Adv}^{GC}_{\mathcal{B}_2}(\lambda). \tag{A6}$$

- $\mathcal{H}_{\gamma,3}$ for $\gamma \in [0, n-1]$: This hybrid is the same as $\mathcal{H}_{\gamma,2}$ except step 4 (as shown in $\mathcal{H}_\gamma$). Challenger $\mathcal{C}$ changes

$$(\tilde{P}^\gamma, \{\mathsf{lab}'_{j,h^{(\mathsf{t}^*)}_{v,j}}\}_{j\in[\lambda]}) \xleftarrow{\$} \mathsf{Sim}(1^\lambda, \{\mathsf{HEnc}(k_\gamma, (h_v^{(\mathsf{t}^*)}, j, b), \mathsf{lab}_{j,b})\}_{j\in[\lambda],b\in\{0,1\}})$$

to

$$(\tilde{P}^\gamma, \{\mathsf{lab}'_{j,h^{(\mathsf{t}^*)}_{v,j}}\}_{j\in[\lambda]}) \xleftarrow{\$} \mathsf{Sim}(1^\lambda, \{\mathsf{HEnc}(k_\gamma, (h_v^{(\mathsf{t}^*)}, j, b), \mathsf{lab}_{j,h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots\gamma+1],j}})\}_{j\in[\lambda],b\in\{0,1\}}).$$

Please note that td_γ is not used any more, the indistinguishability between $\mathcal{H}_{\gamma,2}$ and $\mathcal{H}_{\gamma,3}$ can be reduced to the security of the chameleon encryption scheme defined in Section 2.7. We need λ hybrids, $\{\mathcal{H}^0_{\gamma,2}, \mathcal{H}^1_{\gamma,2}, \cdots, \mathcal{H}^\lambda_{\gamma,2}\}$, to prove this, where $\mathcal{H}^i_{\gamma,2}, i \in [0,\lambda]$ is the same as $\mathcal{H}_{\gamma,2}$ except the generation of $\tilde{P}^\gamma$. Set

$$ct_{j,b} := \begin{cases} \mathsf{HEnc}(k_\gamma, (h_v^{(\mathsf{t}^*)}, j, b), \mathsf{lab}_{j,b}) & \text{if} \quad j > i \\ \mathsf{HEnc}(k_\gamma, (h_v^{(\mathsf{t}^*)}, j, b), \mathsf{lab}_{j,h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots\gamma+1],j}}) & \text{if} \quad j \leq i \end{cases}$$

In $\mathcal{H}^i_{\gamma,2}$,

$$(\tilde{P}^\gamma, \{\mathsf{lab}'_{j,h^{(\mathsf{t}^*)}_{v,j}}\}_{j\in[\lambda]}) \xleftarrow{\$} \mathsf{Sim}(1^\lambda, \{ct_{j,b}\}_{j\in[\lambda],b\in\{0,1\}}). \tag{A7}$$

Obviously, $\mathcal{H}^0_{\gamma,2}$ is the same as $\mathcal{H}_{\gamma,2}$ and $\mathcal{H}^\lambda_{\gamma,2}$ is the same as $\mathcal{H}_{\gamma,3}$. Please note that the only difference between $\mathcal{H}^{i-1}_{\gamma,2}$ and $\mathcal{H}^i_{\gamma,2}$ is $ct_{i,1-h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots\gamma+1],i}}$, where

$$ct_{i,1-h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots\gamma+1],i}} := \begin{cases} \mathsf{HEnc}(k_\gamma, (h_v^{(\mathsf{t}^*)}, i, 1-h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots\gamma+1],i}), \mathsf{lab}_{i,1-h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots\gamma+1],i}}) & \text{in} \quad \mathcal{H}^{(i-1)}_\gamma \\ \mathsf{HEnc}(k_\gamma, (h_v^{(\mathsf{t}^*)}, i, 1-h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots\gamma+1],i}), \mathsf{lab}_{i,h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots\gamma+1],i}}) & \text{in} \quad \mathcal{H}^{(i)}_\gamma \end{cases}.$$

If there is a PPT adversary $\mathcal{A}$ who can distinguish $\mathcal{H}^{i-1}_{\gamma,2}$ and $\mathcal{H}^i_{\gamma,2}$ with a advantage ϵ for $i \in [\lambda]$, we can construct a PPT distinguisher $\mathcal{B}_3$ can use this adversary to break the security of the chameleon encryption scheme with the same advantage ϵ. $\mathcal{B}_3$ simulates $\mathcal{H}^{i-1}_{\gamma,2}$ ($\mathcal{H}^i_{\gamma,2}$) for $\mathcal{A}$ as follows:

1. $\mathcal{B}_3$ receives a hash key k^* from it own challenger $\mathcal{C}^{\text{IND-CE}}$ of the chameleon encryption scheme.
2. $\mathcal{B}_3$ generates $(\mathsf{mpk}, \mathsf{msk}, \mathsf{st}) \leftarrow \mathsf{RIBE.Setup}(1^\lambda, n)\}$. $\mathcal{B}_3$ resets $k_\gamma := k^*$. Now $\mathcal{B}_3$ does not know the corresponding chameleon hash trapdoor td_γ. Then $\mathcal{B}_3$ sends mpk to $\mathcal{A}$.
3. $\mathcal{B}_3$ can perfectly simulates all oracles for $\mathcal{A}$ since these oracles do not need the trapdoor td_γ anymore.
4. When receiving the challenge query $(\mathsf{id}^*, \mathsf{t}^*, M^*_0, M^*_1)$, $\mathcal{B}_3$ sets $x^* = h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots\gamma]||0} || h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots\gamma]||1}$, $r^* := r^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots\gamma]}$(Please note that $r^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots\gamma]}$ is chosen randomly).
   - $\mathcal{B}_3$ generates $\{\tilde{Q}, \tilde{P}^{n-1}, \cdots, \tilde{P}^{\gamma+1}\}$ according to (A3), just like $\mathcal{H}^{i-1}_{\gamma,2}$ ($\mathcal{H}^i_{\gamma,2}$).
   - To generate $\tilde{P}^\gamma$, $\mathcal{B}_3$ does the following. It computes $\{ct_{j,b}\}_{j\in[\lambda],b\in\{0,1\}}$ but leaves $ct_{i,1-h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots\gamma+1],i}}$ undefined. Set $m^*_0 := \mathsf{lab}_{i,h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots\gamma+1],i}}$ and $m^*_1 := \mathsf{lab}_{i,1-h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots\gamma+1],i}}$, where

$(\tilde{P}^{\gamma+1}, \{\mathsf{lab}_{j,b}\}_{j\in[\lambda],b\in\{0,1\}}) \xleftarrow{\$} \mathsf{GCircuit}(1^\lambda, \mathsf{P}\ [\mathsf{id}^*[\gamma+2], k_{\gamma+1}, \overline{\mathsf{lab}}''])$. $\mathcal{B}_3$ sets its own challenge query as $(x^*, r^*, i, m_0^*, m_1^*)$. Then the challenger of $\mathcal{B}_3$ generates a challenge ciphertext ct^* and sends it to $\mathcal{B}_3$. $\mathcal{B}_3$ sets $ct_{i,1-h^{(\mathsf{t}^*)}_{\mathsf{id}^*[1\cdots\gamma+1],i}} := ct^*$. In addition, it computes $\tilde{P}^\gamma$ as (A7).

- $\mathcal{B}_3$ generates $\{\tilde{P}^{\gamma-1}, \cdots, \tilde{P}^0\}$ according to (A4), just like $\mathcal{H}^{i-1}_{\gamma,2}$ ($\mathcal{H}^{i}_{\gamma,2}$).
- $\mathcal{B}_3$ sends $\mathcal{A}$ the challenge ciphertext

  $$\mathsf{ct} := \left(\left\{ \mathsf{lab}_{\ell, h^{(\mathsf{t}^*)}_{\varepsilon,\ell}} \right\}_{\ell\in[\lambda]}, \{\tilde{P}^0, \cdots, \tilde{P}^{n-1}, \tilde{Q}\} \right).$$

5. If $\mathcal{A}$ returns a guessing bit θ' to $\mathcal{B}_3$, $\mathcal{B}_3$ returns θ' to its own challenger.

If ct^* is the chameleon encryption of m_0^*, $\mathcal{B}_3$ simulates $\mathcal{H}^i_{\gamma,2}$ perfectly for $\mathcal{A}$. If ct^* is the chameleon encryption of m_1^*, $\mathcal{B}_3$ simulates $\mathcal{H}^{i-1}_{\gamma,2}$ perfectly for $\mathcal{A}$. If $\mathcal{A}$ can distinguish these two hybrids with advantage ϵ, $\mathcal{B}_3$ can break the security of the multi-bit chameleon encryption with the same advantage ϵ. Therefore,

$$|\Pr[\mathcal{H}^{i-1}_{\gamma,2} \Rightarrow 1] - \Pr[\mathcal{H}^{i}_{\gamma,2} \Rightarrow 1]| \leq \mathbf{Adv}^{CE}_{\mathcal{B}_3}(\lambda) \text{ and}$$

$$|\Pr[\mathcal{H}^{0}_{\gamma,2} \Rightarrow 1] - \Pr[\mathcal{H}^{\lambda}_{\gamma,2} \Rightarrow 1]| \leq \lambda \cdot \mathbf{Adv}^{CE}_{\mathcal{B}_3}(\lambda).$$

Recall that $\mathcal{H}^0_{\gamma,2}$ is the same as $\mathcal{H}_{\gamma,2}$ and $\mathcal{H}^\lambda_{\gamma,2}$ is the same as $\mathcal{H}_{\gamma,3}$. We have

$$|\Pr[\mathcal{H}_{\gamma,2} \Rightarrow 1] - \Pr[\mathcal{H}_{\gamma,3} \Rightarrow 1]| \leq \lambda \cdot \mathbf{Adv}^{CE}_{\mathcal{B}_3}(\lambda). \tag{A8}$$

- $\mathcal{H}_{\gamma,4}$ for $\gamma \in \{0, 1, \cdots, n-1\}$: In this hybrid, challenger $\mathcal{C}$ undoes the changes made from $\mathcal{H}_\gamma$ and $\mathcal{H}_{\gamma,1}$. It is obvious that the computational indistinguishability between $\mathcal{H}_{\gamma,3}$ and $\mathcal{H}_{\gamma,4}$ also follows from uniformity properties of the chameleon hash. Please note that $\mathcal{H}_{\gamma,4}$ is the same as $\mathcal{H}_{\gamma+1}$. We have

  $$|\Pr[\mathcal{H}_{\gamma,3} \Rightarrow 1] - \Pr[\mathcal{H}_{\gamma,4} \Rightarrow 1]| = \mathsf{negl}. \tag{A9}$$

  Therefore, from Equations (A5), (A6), (A8) and (A9) and the fact that $\mathcal{H}_{\gamma,4}$ is the same to $\mathcal{H}_{\gamma+1}$, it holds

  $$|\Pr[\mathcal{H}_\gamma \Rightarrow 1] - \Pr[\mathcal{H}_{\gamma+1} \Rightarrow 1]| = \mathsf{negl} + \mathbf{Adv}^{GC}_{\mathcal{B}_2}(\lambda) + \lambda \cdot \mathbf{Adv}^{CE}_{\mathcal{B}_3}(\lambda). \tag{A10}$$

  $$|\Pr[\mathcal{H}_0 \Rightarrow 1] - \Pr[\mathcal{H}_n \Rightarrow 1]| = \mathsf{negl} + n \cdot \mathbf{Adv}^{GC}_{\mathcal{B}_2}(\lambda) + n \cdot \lambda \cdot \mathbf{Adv}^{CE}_{\mathcal{B}_3}(\lambda). \tag{A11}$$

- $\mathcal{H}_{n+1}$: This hybrid is the same as $\mathcal{H}_n$ except that the challenger $\mathcal{C}$ changes the way of generating $\tilde{Q}$. More Formally, $\mathcal{C}$ changes the generation process of garbled circuits $\tilde{Q}$ from

  $$(\tilde{Q}, \overline{\mathsf{lab}}) \leftarrow \mathsf{GCircuit}(1^\lambda, \mathsf{Q}[M_{b'}])$$

  to

  $$(\tilde{Q}, \{\mathsf{lab}_{j, h^{(\mathsf{t}^*)}_{\mathsf{id}^*,j}}\}) \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{E}_{h^{(\mathsf{t}^*)}_{\mathsf{id}^*}}(M_\theta))$$

  This indistinguishability between $\mathcal{H}_n$ and $\mathcal{H}_{n+1}$ follows by the security of the garble circuit scheme. The proof is similar to the indistinguishability between $\mathcal{H}_{\gamma,1}$ and $\mathcal{H}_{\gamma,2}$.Hence,

  $$|\Pr[\mathcal{H}_n \Rightarrow 1] - \Pr[\mathcal{H}_{n+1} \Rightarrow 1]| \leq \mathbf{Adv}^{GC}_{\mathcal{B}_2}(\lambda). \tag{A12}$$

- $\mathcal{H}_{n+2}$: This hybrid is the same as $\mathcal{H}_{n+1}$ except that the challenger $\mathcal{C}$ replaces the ciphertext $\mathsf{E}_{h^{(\mathsf{t}^*)}_{\mathsf{id}^*||0}}(M_\theta)$ hard-coded in the circuit $\tilde{Q}$ with $\mathsf{E}_{h^{(\mathsf{t}^*)}_{\mathsf{id}^*||0}}(0)$.
  If there is a PPT adversary $\mathcal{A}$ who can distinguish between $\mathcal{H}_{n+1}$ and $\mathcal{H}_{n+2}$ with advantage ϵ,

there is a PPT distinguisher $\mathcal{B}_4$ who can break the IND-CPA security of $\mathsf{PKE} = (\mathsf{G}, \mathsf{E}, \mathsf{D})$ with advantage $\epsilon/(2q+1)$. First of all, We consider two kinds of adversaries:

Type-I : $\mathcal{A}_I$ never queries id^* to key generation oracle $\mathrm{KG}(\cdot)$ for $\mathsf{sk}_{\mathsf{id}^*}$.
Type-II : $\mathcal{A}_{II}$ queries id^* to $\mathrm{KG}(\cdot)$ and obtains $\mathsf{sk}_{\mathsf{id}^*}$. In this case id^* should be revoked before t^*.

$$\begin{aligned} & |\Pr[\mathcal{H}_{n+1} \Rightarrow 1] - \Pr[\mathcal{H}_{n+2} \Rightarrow 1]| \\ = \; & |\Pr[\mathcal{H}_{n+1} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_I] - \Pr[\mathcal{H}_{n+2} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_I]| \cdot \Pr[\mathcal{A} = \mathcal{A}_I] \\ & + |\Pr[\mathcal{H}_{n+1} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_{II}] - \Pr[\mathcal{H}_{n+2} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_{II}]| \cdot \Pr[\mathcal{A} = \mathcal{A}_{II}] \\ \leq \; & |\Pr[\mathcal{H}_{n+1} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_I] - \Pr[\mathcal{H}_{n+2} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_I]| \\ & + |\Pr[\mathcal{H}_{n+1} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_{II}] - \Pr[\mathcal{H}_{n+2} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_{II}]| \end{aligned} \tag{A13}$$

Claim 1 $|\Pr[\mathcal{H}_{n+1} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_I] - \Pr[\mathcal{H}_{n+2} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_I]| \leq (q+1) \cdot \mathbf{Adv}^{PKE}_{\mathcal{B}_4}(\lambda)$.

Proof. Suppose that $\mathcal{A}_I$ issues q_K queries to oracle $\mathrm{KG}(\cdot)$, and let $\mathsf{id}^{(j)}$ be $\mathcal{A}_I$'s j-th query to $\mathrm{KG}(\cdot)$. For each PPT $\mathcal{A}_I$ such that $|\Pr[\mathcal{H}_{n+1} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_I] - \Pr[\mathcal{H}_{n+2} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_I]| = \epsilon$, we build a PPT algorithm $\mathcal{B}_4$ breaking the IND-CPA security of PKE with the advantage $q_K \cdot \epsilon$. $\mathcal{B}_4$ simulates $\mathcal{H}_{n+1}$ ($\mathcal{H}_{n+2}$) to $\mathcal{A}_I$ as follows:

1. $\mathcal{B}_4$ generates $(\mathsf{mpk}, \mathsf{msk}, \mathsf{st}) \leftarrow \mathsf{RIBE.Setup}(1^\lambda, n)$. $\mathcal{B}_4$ sends mpk to $\mathcal{A}_I$.
2. $\mathcal{B}_4$ receives encryption key ek^* from its own challenger.
3. $\mathcal{B}_4$ chooses $j \xleftarrow{\$} [0, q_K]$.
4. If $j \neq 0$, since $\mathcal{B}_4$ has msk, $\mathcal{B}_4$ can perfectly simulate all oracles for $\mathcal{A}_{II}$ as in $\mathcal{H}_{n+1}$ ($\mathcal{H}_{n+2}$). $\mathcal{B}_4$ embeds ek^* in the private key generation of identity $\mathsf{id}^{(j)}$ (the output of $\mathrm{KG}(\mathsf{id}^{(j)})$). More specifically, $\mathcal{B}_4$ invokes $\mathsf{RIBE.KG}(\mathsf{msk}, \mathsf{id}^{(j)}, \mathsf{st})$ with a little change (framed parts are added) in the fourth step of the algorithm as follows:

   – For $v = \mathsf{id}^{(j)}[1 \cdots n-1]$:
   $h_v \leftarrow \mathsf{H}(k_n, 0^{2\lambda}; w_v)$, where $w_r \leftarrow \mathsf{RF}$.
   $(\mathsf{ek}_{v||\mathsf{id}^{(j)}[n]}, \mathsf{dk}_{v||\mathsf{id}^{(j)}[n]}) \leftarrow \mathsf{G}(1^\lambda)$,
   $(h_v, \mathsf{ek}_{v||0}, \mathsf{ek}_{v||1}, r_v, \mathsf{dk}_{v||0}, \mathsf{dk}_{v||1}) \leftarrow \mathsf{LeafGen}(k_{n-1}, (td_{n-1}, s), v, \ell)$
   $\boxed{\mathsf{ek}_{v||(1-\mathsf{id}^{(j)}[n])} := \mathsf{ek}^*.}$
   $\boxed{r_v \leftarrow \mathsf{H}^{-1}(td_{n-1}, (0^{2\lambda}, w_v), \mathsf{ek}_{v||0}||\mathsf{ek}_{v||1}).}$
   $\mathsf{KL} := \mathsf{KL} \cup \{(v, h_v, \mathsf{ek}_{v||0}, \mathsf{ek}_{v||1}, r_v), (v||0, \mathsf{ek}_{v||0}, \perp), (v||1, \mathsf{ek}_{v||1}, \perp)\}$,
   $\mathsf{lk}_v := (h_v, \mathsf{ek}_{v||0}, \mathsf{ek}_{v||1}, r_v)$.
   Recall that $\mathsf{ek}_{v||(1-\mathsf{id}^{(j)}[n])}$ is generated by
   $(\mathsf{ek}_{v||(1-\mathsf{id}^{(j)}[n])}, \mathsf{dk}_{v||(1-\mathsf{id}^{(j)}[n])}) \leftarrow \mathsf{G}(1^\lambda)$ in $\mathsf{LeafGen}$ algorithm, hence $\mathsf{ek}_{v||(1-\mathsf{id}^{(j)}[n])}$ has identical distribution with ek^*. As a result, $\mathcal{B}_4$ perfectly simulates the oracle KG on for $\mathcal{A}_I$ just like $\mathcal{H}_{n+1}$ ($\mathcal{H}_{n+2}$).
5. $\mathcal{B}_4$ receives the challenge query $(\mathsf{id}^*, \mathsf{t}^*, M_0, M_1)$ from $\mathcal{A}_I$.
   If $j \neq 0$ and $\mathsf{id}^* \neq \mathsf{id}^{(j)}[1 \cdots n-1]||(1 - \mathsf{id}^{(j)}[n])$, $\mathcal{B}_4$ aborts the game.
   If $j = 0$ and there exists $i \in [q_K]$ such that $\mathsf{id}^* = \mathsf{id}^{(i)}[1 \cdots n-1]||(1 - \mathsf{id}^{(i)}[n])$, $\mathcal{B}_4$ aborts the game.
   Otherwise:
   $\mathcal{B}_4$ chooses $\theta \xleftarrow{\$} \{0, 1\}$.
   $\mathcal{B}_4$ sets its own challenge query as $m_0^* := 0$ and $m_1^* = M_\theta$.
   After receiving the challenge ciphertext ct^* from its own challenger, $\mathcal{B}_4$ computes

$(\tilde{Q}, \{\mathsf{lab}_{j,\mathsf{ek}_{\mathsf{id}'^*,j}}\}) \leftarrow \mathsf{Sim}(1^\lambda, ct^*)$, and continues to generate $\{\tilde{P}^{n-1}, \cdots, \tilde{P}^0\}$ according to (A4), just like $\mathcal{H}_{n+1}$ ($\mathcal{H}_{n+2}$).

$\mathcal{B}_4$ sends the challenge ciphertext $\mathsf{ct} := \left(\left\{ \mathsf{lab}_{\ell, h_{\varepsilon,\ell}^{(\mathsf{t}^*)}} \right\}_{\ell \in [\lambda]}, \{\tilde{P}^0, \cdots, \tilde{P}^{n-1}, \tilde{Q}\} \right)$ to $\mathcal{A}_{II}$.

6. Finally, $\mathcal{B}_4$ outputs what $\mathcal{A}_I$ outputs.

As long as $\mathcal{B}_4$ does not abort the game, it simulates $\mathcal{H}_{n+1}$ when ct^* is the encryption of $M_{b''}$ and $\mathcal{B}_4$ simulates $\mathcal{H}_{n+2}$ when ct^* is the encryption of 0. Therefore, we have

$$\begin{aligned}
\mathbf{Adv}_{\mathcal{B}_4}^{PKE}(\lambda) &= |\Pr[\neg\text{abort} \wedge \mathcal{H}_{n+1} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_I] - \Pr[\neg\text{abort} \wedge \mathcal{H}_{n+2} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_I]| \\
&= |\Pr[\mathcal{H}_{n+1} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_I] - \Pr[\mathcal{H}_{n+2} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_I]| \cdot \Pr[\neg\text{abort}] \qquad \text{(A14)} \\
&= 1/(q_K + 1) \cdot |\Pr[\mathcal{H}_{n+1} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_I] - \Pr[\mathcal{H}_{n+2} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_I]| \\
&\geq 1/(q+1) \cdot |\Pr[\mathcal{H}_{n+1} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_{II}] - \Pr[\mathcal{H}_{n+2} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_{II}]|. \qquad \text{(A15)}
\end{aligned}$$

(A14) holds due to the fact that $j \leftarrow [0, q_K]$ is independently chosen while (A15) holds due to $q_K \leq q$. □

Claim 2 $|\Pr[\mathcal{H}_{n+1} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_{II}] - \Pr[\mathcal{H}_{n+2} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_{II}]| \leq q \cdot \mathbf{Adv}_{\mathcal{B}_4}^{PKE}(\lambda)$.

Proof. Suppose that $\mathcal{A}_{II}$ issues q_R queries to oracle $\textsc{Rvk}(\cdot)$, and let $(\mathsf{id}^{(j)}, \mathsf{t}^{(j)})$ be $\mathcal{A}_{II}$'s j-th query to $\textsc{Rvk}(\cdot)$. For each PPT $\mathcal{A}_{II}$ such that $|\Pr[\mathcal{H}_{n+1} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_{II}] - \Pr[\mathcal{H}_{n+2} \Rightarrow 1 | \mathcal{A} = \mathcal{A}_{II}]| = \epsilon$, we build a PPT algorithm $\mathcal{B}_4$ breaking the IND-CPA security of PKE with the advantage $q_R \cdot \epsilon$. $\mathcal{B}_4$ simulates $\mathcal{H}_{n+1}$ ($\mathcal{H}_{n+2}$) to $\mathcal{A}_{II}$ as follows:

1. $\mathcal{B}_4$ generates $(\mathsf{mpk}, \mathsf{msk}, \mathsf{st}) \leftarrow \mathsf{RIBE.Setup}(1^\lambda, n)\}$. $\mathcal{B}_4$ sends mpk to $\mathcal{A}_{II}$.
2. $\mathcal{B}_4$ receives encryption key ek^* from its own challenger.
3. $\mathcal{B}_4$ chooses $j \xleftarrow{\$} [q_R]$.
4. Since $\mathcal{B}_4$ has msk, $\mathcal{B}_4$ can perfectly simulate all oracles for $\mathcal{A}_{II}$ as in $\mathcal{H}_{n+1}$ ($\mathcal{H}_{n+2}$). $\mathcal{B}_4$ embeds ek^* in the key update generation (the output of KU) of time $\mathsf{t}^{(j)}$. More specifically, $\mathcal{B}_4$ invokes $\mathsf{RIBE.KU}(\mathsf{msk}, \mathsf{t}^{(j)}, \mathsf{st})$ with a little change (a framed part is added) in the fifth step of the algorithm as follows:

   – For all node $v \in \mathsf{Y}$ such that $|v| = n$: $(\mathsf{ek}_v^{\mathsf{t}^{(j)}}, \perp) \leftarrow \mathsf{LeafChange}(v, \mathsf{t}^{(j)}, s)$,
   $\boxed{\text{If } v = \mathsf{id}^{(j)}, \mathsf{ek}_v^{\mathsf{t}^{(j)}} := \mathsf{ek}^*.}$
   $\mathsf{KU}^{\mathsf{t}^{(j)}} := \mathsf{KU}^{\mathsf{t}^{(j)}} \cup \{(v, \mathsf{ek}_v^{\mathsf{t}^{(j)}}, \perp)\}$.
   $h_v^{\mathsf{t}^{(j)}} := \mathsf{ek}_v^{\mathsf{t}^{(j)}}$.
   Recall that $\mathsf{ek}_v^{(j)}$ is generated by $(\mathsf{ek}_v^{(j)}, \mathsf{dk}_v^{(j)}) \leftarrow \mathsf{G}(1^\lambda)$ in LeafChange algorithm, hence $\mathsf{ek}_v^{(j)}$ has an identical distribution with ek^*. As a result, $\mathcal{B}_4$ perfectly simulates the oracle KU for $\mathcal{A}_{II}$ just like $\mathcal{H}_{n+1}$ ($\mathcal{H}_{n+2}$).
5. $\mathcal{B}_4$ receives the challenge query parsed as $(\mathsf{id}^*, \mathsf{t}^*, M_0, M_1)$ from $\mathcal{A}_{II}$.
   If $\mathsf{id}^* \neq \mathsf{id}^{(j)}$, $\mathcal{B}$ aborts the game.
   Else:
   $\mathcal{B}_4$ chooses $\theta \xleftarrow{\$} \{0,1\}$.
   $\mathcal{B}_4$ sets its own challenge query as $m_0^* := 0$ and $m_1^* = M_\theta$.
   After receiving the challenge ciphertext ct^* from its own challenger, $\mathcal{B}_4$ computes $(\tilde{Q}, \{\mathsf{lab}_{j,\mathsf{ek}_{\mathsf{id}'^*,j}}\}) \leftarrow \mathsf{Sim}(1^\lambda, ct^*)$, and continues to generate $\{\tilde{P}^{n-1}, \cdots, \tilde{P}^0\}$ according to (A4), just like $\mathcal{H}_{n+1}$ ($\mathcal{H}_{n+2}$).
   $\mathcal{B}_4$ sends the challenge ciphertext $\mathsf{ct} := \left(\left\{ \mathsf{lab}_{\ell, h_{\varepsilon,\ell}^{(\mathsf{t}^*)}} \right\}_{\ell \in [\lambda]}, \{\tilde{P}^0, \cdots, \tilde{P}^{n-1}, \tilde{Q}\} \right)$ to $\mathcal{A}_{II}$.

6. Finally, $\mathcal{B}_4$ outputs what $\mathcal{A}_{II}$ outputs.

Given that $\mathsf{id}^* = \mathsf{id}^{(j)}$, $\mathcal{B}_4$ simulates $\mathcal{H}_{n+1}$ when ct^* is the encryption of $M_{b''}$ and $\mathcal{B}_4$ simulates $\mathcal{H}_{n+2}$ when ct^* is the encryption of 0. Therefore, we have

$$\begin{aligned}
\mathbf{Adv}^{PKE}_{\mathcal{B}_4}(\lambda) &= |\Pr[\mathsf{id}^* = \mathsf{id}^{(j)} \wedge \mathcal{H}_{n+1} \Rightarrow 1|\mathcal{A} = \mathcal{A}_{II}] - \Pr[\mathsf{id}^* = \mathsf{id}^{(j)} \wedge \mathcal{H}_{n+2} \Rightarrow 1|\mathcal{A} = \mathcal{A}_{II}]| \\
&= |\Pr[\mathcal{H}_{n+1} \Rightarrow 1|\mathcal{A} = \mathcal{A}_{II}] - \Pr[\mathcal{H}_{n+2} \Rightarrow 1|\mathcal{A} = \mathcal{A}_{II}]| \cdot \Pr[\mathsf{id}^* = \mathsf{id}^{(j)}] \quad \text{(A16)} \\
&= 1/q_R \cdot |\Pr[\mathcal{H}_{n+1} \Rightarrow 1|\mathcal{A} = \mathcal{A}_{II}] - \Pr[\mathcal{H}_{n+2} \Rightarrow 1|\mathcal{A} = \mathcal{A}_{II}]| \\
&\geq 1/q \cdot |\Pr[\mathcal{H}_{n+1} \Rightarrow 1|\mathcal{A} = \mathcal{A}_{II}] - \Pr[\mathcal{H}_{n+2} \Rightarrow 1|\mathcal{A} = \mathcal{A}_{II}]|. \quad \text{(A17)}
\end{aligned}$$

(A16) holds due to the fact that $j \leftarrow [q_R]$ is independently chosen while (A17) holds due to $q_R \leq q$. □

According to (A13), Claim 1 and Claim 2, we have

$$|\Pr[\mathcal{H}_{n+1} \Rightarrow 1] - \Pr[\mathcal{H}_{n+2} \Rightarrow 1]| \leq (2q+1) \cdot \mathbf{Adv}^{PKE}_{\mathcal{B}_4}(\lambda). \quad \text{(A18)}$$

- Please note that in $\mathcal{H}_{n+2}$, the challenge ciphertext is information theoretically independent of the plaintexts submitted by $\mathcal{A}$. So we have

$$\Pr[\mathcal{H}_{n+2} \Rightarrow 1] = 1/2. \quad \text{(A19)}$$

Combining (A1), (A2), (A11), (A12), (A18) and (A19), we have

$$\mathbf{Adv}^{\text{IND-ID-CPA}}_{\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{PRF}_{\mathcal{B}_1}(\lambda) + (n+1) \cdot \mathbf{Adv}^{GC}_{\mathcal{B}_2}(\lambda) + n \cdot \lambda \cdot \mathbf{Adv}^{CE}_{\mathcal{B}_3}(\lambda) + (2q+1) \cdot \mathbf{Adv}^{PKE}_{\mathcal{B}_4}(\lambda).$$

□

Appendix B.2. Proof of Theorem 3

Proof. This theorem can be derived from Corollary 1. For any PPT adversary $\mathcal{A}$ in the selective-SR-ID-CPA security game of SR-IBE Σ, we can construct a PPT algorithm $\mathcal{B}$ breaking the selective-IND-ID-CPA security of RIBE Π such that

$$\mathbf{Adv}^{\text{selective-SR-ID-CPA}}_{\Sigma,\mathcal{A}}(\lambda) = \mathbf{Adv}^{\text{selective-IND-ID-CPA}}_{\Pi,\mathcal{B}}(\lambda).$$

$\mathcal{B}$ simulates $\mathbf{EXP}^{\text{selective-SR-ID-CPA}}_{\mathcal{A}}(\lambda)$ as follows:

1. $\mathcal{A}$ generates the challenge identity id^* and time slot t^* to $\mathcal{B}$. Then $\mathcal{B}$ sends the same challenge pair $(\mathsf{id}^*, \mathsf{t}^*)$ to its own challenger.
2. $\mathcal{B}$ receives $\Pi.\mathsf{mpk}$ from its own challenger and sets $\Sigma.\mathsf{mpk} := \Pi.\mathsf{mpk}$. Then $\mathcal{B}$ sends $\Sigma.\mathsf{mpk}$ to $\mathcal{A}$.
3. When $\mathcal{A}$ queries for the oracle PUBKG with an identity id, $\mathcal{B}$ queries for his oracle KG with the same identity id to its own challenger. Set $\mathsf{W} := \{\varepsilon, \mathsf{id}[1], \cdots, \mathsf{id}[1 \cdots n-1]\}$, where ε is the empty string. Upon receiving the $\Pi.\mathsf{sk}_{\mathsf{id}}$, $\mathcal{B}$ parses $\Pi.\mathsf{sk}_{\mathsf{id}} := (\Sigma.\mathsf{pk}_{\mathsf{id}}, \Sigma.\mathsf{sk}_{\mathsf{id}})$, where $\Sigma.\mathsf{pk}_{\mathsf{id}} := (\mathsf{t} = 0, \mathsf{id}, \{\mathsf{lk}_v\}_{v \in \mathsf{W}})$ and $\Sigma.\mathsf{sk}_{\mathsf{id}} := (\mathsf{dk}_{\mathsf{id}}, \mathsf{HIBE.sk}_{\mathsf{id}})$. $\mathcal{B}$ sends $\Sigma.\mathsf{pk}_{\mathsf{id}}$ to $\mathcal{A}$.
4. When $\mathcal{A}$ queries for the oracle PRIVKG with an identity id, $\mathcal{B}$ queries for his oracle KG with the same identity id to its own challenger. Set $\mathsf{W} := \{\varepsilon, \mathsf{id}[1], \cdots, \mathsf{id}[1 \cdots n-1]\}$, where ε is the empty string. Upon receiving the $\Pi.\mathsf{sk}_{\mathsf{id}}$, $\mathcal{B}$ parses $\Pi.\mathsf{sk}_{\mathsf{id}} := (\Sigma.\mathsf{pk}_{\mathsf{id}}, \Sigma.\mathsf{sk}_{\mathsf{id}})$, where $\Sigma.\mathsf{pk}_{\mathsf{id}} := (\mathsf{t} = 0, \mathsf{id}, \{\mathsf{lk}_v\}_{v \in \mathsf{W}})$ and $\Sigma.\mathsf{sk}_{\mathsf{id}} := (\mathsf{dk}_{\mathsf{id}}, \mathsf{HIBE.sk}_{\mathsf{id}})$. $\mathcal{B}$ sends $\Sigma.\mathsf{sk}_{\mathsf{id}}$ to $\mathcal{A}$.
5. When $\mathcal{A}$ queries for the oracle KU, $\mathcal{B}$ queries for the oracle KU to its own challenger. Upon receiving the $(\mathsf{KU}, \mathsf{PL})$, $\mathcal{B}$ sends $(\mathsf{KU}, \mathsf{PL})$ to $\mathcal{A}$.
6. When $\mathcal{A}$ queries for the oracle DK with an identity id and a time slot t, $\mathcal{B}$ queries for his oracle DK with the same pair $(\mathsf{id}, \mathsf{t})$ to its own challenger. Set $\mathsf{W} := \{\varepsilon, \mathsf{id}[1], \cdots, \mathsf{id}[1 \cdots n-1]\}$, where ε is the

empty string. Upon receiving the $\Pi.\mathsf{sk}_{\mathsf{id}}^{(\mathsf{t})}$, $\mathcal{B}$ parses $\Pi.\mathsf{sk}_{\mathsf{id}}^{(\mathsf{t})} := (\mathsf{t}, \mathsf{id}, \{\mathsf{lk}_v\}_{v \in \mathsf{W}}, \mathsf{dk}_{\mathsf{id}}, \mathsf{HIBE.sk}_{\mathsf{id}||\mathsf{t}})$. $\mathcal{B}$ sends $\Sigma.\mathsf{dk}_{\mathsf{id}}^{(t)} := (\mathsf{dk}_{\mathsf{id}}, \mathsf{HIBE.sk}_{\mathsf{id}||\mathsf{t}})$ to $\mathcal{A}$.

7. When $\mathcal{A}$ queries for the oracle RVK with an identity id and a time slot t, $\mathcal{B}$ queries for his oracle RVK with the same pair $(\mathsf{id}, \mathsf{t})$ to its own challenger.
8. When $\mathcal{A}$ submits the challenge query $(\mathsf{m}_0, \mathsf{m}_1)$, $\mathcal{B}$ sends the same challenge query $(\mathsf{m}_0, \mathsf{m}_1)$ to its own challenger. Upon receiving the challenge ciphertext $\Pi.\mathsf{ct}_\theta^*$, $\mathcal{B}$ sends $\Sigma.\mathsf{ct}_\theta^* := \Pi.\mathsf{ct}_\theta^*$ to $\mathcal{A}$.
9. Finally, $\mathcal{B}$ outputs what $\mathcal{A}$ outputs.

Since $\mathbf{EXP}_{\mathcal{B}}^{\text{selective-IND-ID-CPA}}(\lambda)$ has the same requirements as $\mathbf{EXP}_{\mathcal{A}}^{\text{selective-SR-ID-CPA}}(\lambda)$, $\mathcal{B}$ simulates $\mathbf{EXP}_{\mathcal{A}}^{\text{selective-SR-ID-CPA}}(\lambda)$ to $\mathcal{A}$ perfectly. Therefore, we have

$$\mathbf{Adv}_{\Sigma,\mathcal{A}}^{\text{selective-SR-ID-CPA}}(\lambda) = \mathbf{Adv}_{\Pi,\mathcal{B}}^{\text{selective-IND-ID-CPA}}(\lambda). \tag{A20}$$

According to Corollary 1, we have that for PPT adversary $\mathcal{B}$, the advantage is negligible:

$$\mathbf{Adv}_{\Pi,\mathcal{B}}^{\text{selective-IND-ID-CPA}}(\lambda) = \mathsf{negl}(\lambda). \tag{A21}$$

Combining (A20) and (A21), for any PPT adversary $\mathcal{A}$, we have

$$\mathbf{Adv}_{\Sigma,\mathcal{A}}^{\text{selective-SR-ID-CPA}}(\lambda) = \mathsf{negl}(\lambda).$$

Thus we finish the proof. □

References

1. Shamir, A. Identity-Based Cryptosystems and Signature Schemes. In Proceedings of the CRYPTO 1984, Advances in Cryptology, Santa Barbara, CA, USA, 19–22 August 1984; pp. 47–53.10.1007/3-540-39568-7_5. [CrossRef]
2. Waters, B. Efficient Identity-Based Encryption Without Random Oracles. In Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 2005), Aarhus, Denmark, 22–26 May 2005; pp. 114–127.10.1007/11426639_7. [CrossRef]
3. Gentry, C. Practical Identity-Based Encryption Without Random Oracles. In Proceedings of the 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 2006), St. Petersburg, Russia, 28 May–1 June 2006; pp. 445–464. 11761679_27. [CrossRef]
4. Okamoto, T.; Takashima, K. Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In Proceedings of the 30th Annual Cryptology Conference, Advances in Cryptology (CRYPTO 2010), Santa Barbara, CA, USA, 15–19 August 2010; pp. 191–208. 978-3-642-14623-7_11. [CrossRef]
5. Gentry, C.; Peikert, C.; Vaikuntanathan, V. Trapdoors for hard lattices and new cryptographic constructions. In Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, BC, Canada, 17–20 May 2008; pp. 197–206.10.1145/1374376.1374407. [CrossRef]
6. Agrawal, S.; Boneh, D.; Boyen, X. Efficient Lattice (H)IBE in the Standard Model. In Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology: (EUROCRYPT 2010), Monaco/Nice, France, 30 May–3 June 2010; pp. 553–572.10.1007/978-3-642-13190-5_28. [CrossRef]
7. Cash, D.; Hofheinz, D.; Kiltz, E.; Peikert, C. Bonsai Trees, or How to Delegate a Lattice Basis. In Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 2010), Monaco/Nice, France, 30 May–3 June 2010; pp. 523–552.10.1007/978-3-642-13190-5_27. [CrossRef]

8. Döttling, N.; Garg, S. Identity-Based Encryption from the Diffie-Hellman Assumption. In Proceedings of the 37th Annual International Cryptology Conference, Advances in Cryptology (CRYPTO 2017), Santa Barbara, CA, USA, 20–24 August 2017; pp. 537–569.10.1007/978-3-319-63688-7_18. [CrossRef]
9. Boneh, D.; Franklin, M.K. Identity-Based Encryption from the Weil Pairing. *SIAM J. Comput.* **2003**, *32*, 586–615.10.1137/S0097539701398521. [CrossRef]
10. Boldyreva, A.; Goyal, V.; Kumar, V. Identity-based encryption with efficient revocation. In Proceedings of the 2008 ACM Conference on Computer and Communications Security (CCS 2008), Alexandria, VA, USA, 27–31 October 2008; pp. 417–426.10.1145/1455770.1455823. [CrossRef]
11. Sahai, A.; Waters, B. Fuzzy Identity-Based Encryption. In Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 2005), Aarhus, Denmark, 22–26 May 2005; pp. 457–473.10.1007/11426639_27. [CrossRef]
12. Libert, B.; Vergnaud, D. Adaptive-ID Secure Revocable Identity-Based Encryption. In Proceedings of the Cryptographers' Track at the RSA Conference on Topics in Cryptology (CT-RSA 2009), San Francisco, CA, USA, 20–24 April 2009; pp. 1–15.10.1007/978-3-642-00862-7_1. [CrossRef]
13. Seo, J.H.; Emura, K. Revocable Identity-Based Encryption Revisited: Security Model and Construction. In Proceedings of the 16th International Conference on Practice and Theory in Public-Key Cryptography: Public-Key Cryptography (PKC 2013), Nara, Japan, 26 February–1 March 2013; pp. 216–234. 978-3-642-36362-7_14. [CrossRef]
14. Lee, K.; Lee, D.H.; Park, J.H. Efficient revocable identity-based encryption via subset difference methods. *Des. Codes Cryptogr.* **2017**, *85*, 39–76.10.1007/s10623-016-0287-3. [CrossRef]
15. Watanabe, Y.; Emura, K.; Seo, J.H. New Revocable IBE in Prime-Order Groups: Adaptively Secure, Decryption Key Exposure Resistant, and with Short Public Parameters. In Proceedings of the Cryptographers' Track at the RSA Conference on Topics in Cryptology (CT-RSA 2017), San Francisco, CA, USA, 14–17 February 2017; pp. 432–449.10.1007/978-3-319-52153-4_25. [CrossRef]
16. Park, S.; Lee, K.; Lee, D.H. New Constructions of Revocable Identity-Based Encryption from Multilinear Maps. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 1564–1577.10.1109/TIFS.2015.2419180. [CrossRef]
17. Chen, J.; Lim, H.W.; Ling, S.; Wang, H.; Nguyen, K. Revocable Identity-Based Encryption from Lattices. In Proceedings of the 17th Australasian Conference on Information Security and Privacy (ACISP 2012), Wollongong, NSW, Australia, 9–11 July 2012; pp. 390–403.10.1007/978-3-642-31448-3_29. [CrossRef]
18. Takayasu, A.; Watanabe, Y. Lattice-Based Revocable Identity-Based Encryption with Bounded Decryption Key Exposure Resistance. In Proceedings of the 22nd Australasian Conference on Information Security and Privacy (ACISP 2017), Auckland, New Zealand, 3–5 July 2017; pp. 184–204. 60055-0_10. [CrossRef]
19. Qin, B.; Deng, R.H.; Li, Y.; Liu, S. Server-Aided Revocable Identity-Based Encryption. In Proceedings of the 20th European Symposium on Research in Computer Security (ESORICS 2015), Vienna, Austria, 21–25 September 2015; pp. 286–304.10.1007/978-3-319-24174-6_15. [CrossRef]
20. Liang, K.; Liu, J.K.; Wong, D.S.; Susilo, W. An Efficient Cloud-Based Revocable Identity-Based Proxy Re-encryption Scheme for Public Clouds Data Sharing. In Proceedings of the 19th European Symposium on Research in Computer Security (ESORICS 2014), Wroclaw, Poland, 7–11 September 2014; pp. 257–272.10.1007/978-3-319-11203-9_15. [CrossRef]
21. Yang, Y.; Liu, J.K.; Liang, K.; Choo, K.R.; Zhou, J. Extended Proxy-Assisted Approach: Achieving Revocable Fine-Grained Encryption of Cloud Data. In Proceedings of the 20th European Symposium on Research in Computer Security (ESORICS 2015), Vienna, Austria, 21–25 September 2015; pp. 146–166. 978-3-319-24177-7_8. [CrossRef]
22. Yang, Y.; Liu, J.K.; Wei, Z.; Huang, X. Towards Revocable Fine-Grained Encryption of Cloud Data: Reducing Trust upon Cloud. In Proceedings of the 22nd Australasian Conference on Information Security and Privacy (ACISP 2017), Auckland, New Zealand, 3–5 July 2017; pp. 127–144.10.1007/978-3-319-60055-0_7. [CrossRef]
23. Liu, J.K.; Yuen, T.H.; Zhang, P.; Liang, K. Time-Based Direct Revocable Ciphertext-Policy Attribute-Based Encryption with Short Revocation List. In Proceedings of the 16th International Conference on Applied Cryptography and Network Security (ACNS 2018), Leuven, Belgium, 2–4 July 2018; pp. 516–534.10.1007/978-3-319-93387-0_27. [CrossRef]

24. Katsumata, S.; Matsuda, T.; Takayasu, A. Lattice-based Revocable (Hierarchical) IBE with Decryption Key Exposure Resistance. *IACR Cryptol. ePrint Arch.* **2018**, *2018*, 420.