Lattice-based Revocable (Hierarchical) IBE with Decryption Key Exposure Resistance

[*]Shuichi Katsumata [†]Takahiro Matsuda [‡]Atsushi Takayasu

May 8, 2018

Abstract

Revocable identity-based encryption (RIBE) is an extension of IBE that supports a key revocation mechanism; an indispensable feature for practical cryptographic schemes. Due to this extra feature, RIBE is often required to satisfy a strong security notion unique to the revocation setting called *decryption key exposure resistance* (DKER). Additionally, *hierarchal* IBE (HIBE) is another orthogonal extension of IBE that supports key delegation functionalities allowing for scalable deployments of cryptographic schemes. Thus far, R(H)IBE constructions with DKER are only known from bilinear maps, where all constructions rely heavily on the so-called *key re-randomization* property to achieve the DKER and/or hierarchal feature. Since lattice-based schemes seem to be inherently ill-fit with the key re-randomization property, we currently do not know of any lattice-based R(H)IBE schemes with DKER.

In this paper, we propose the first lattice-based RHIBE scheme with DKER *without* relying on the key re-randomization property, departing from all the previously known methods. We start our work by providing a generic construction of RIBE schemes with DKER, which uses as building blocks any two-level standard HIBE scheme and (weak) RIBE scheme *without* DKER. Based on previous lattice-based RIBE constructions, our result implies the first lattice-based RIBE scheme with DKER. Then, building on top of our generic construction, we construct the first lattice-based RHIBE scheme with DKER, by further exploiting the algebraic structure of lattices. To this end, we prepare a new tool called the *level conversion keys*, which allows us to achieve the hierarchal feature without relying on the key re-randomization property.

1 Introduction

Background. *Identity-based encryption* (IBE) is an advanced form of public key encryption, where an arbitrary string can be used as user's public keys. One extension of IBE is *hierarchical* IBE (HIBE), which further supports a key delegation functionality; an attractive feature for scalable deployments of IBE. However, as opposed to ordinary public key encryption, (H)IBE does not support a key/user revocation mechanism due to the absence of the public key infrastructures and there are no trivial ways to drive malicious users out from an ordinary (H)IBE system. Therefore, adding a key revocation mechanism to (H)IBE is considered to be one of the important research themes when considering practical deployments of (H)IBE. For instance, Boneh and

[*]The University of Tokyo, Japan, and National Institute of Advanced Industrial Science and Technology (AIST), Japan. The author is supported by a JSPS Fellowship for Young Scientists.

[†]National Institute of Advanced Industrial Science and Technology (AIST), Japan.

[‡]The University of Tokyo, Japan, and National Institute of Advanced Industrial Science and Technology (AIST), Japan. e-mail: takayasu@mist.i.u-tokyo.ac.jp

Franklin [BF03] proposed a method for adding a simple revocation mechanism to any IBE systems. However, the bottleneck of their proposal was its efficiency. The number of keys generated for every time period was proportional to the number of all users in the IBE system and the scheme did not scale if the number of users became too large. Since then, constructing an (H)IBE scheme with a scalable revocation mechanism has been a sought-after goal. Below, we refer to an (H)IBE that allows for such a scalable revocation mechanism as *revocable (H)IBE*.

The first revocable IBE (RIBE) scheme was proposed by Boldyreva et al. [BGK08]. RIBE requires three types of keys: a *secret key*, a *key update*, and a *decryption key*. As in IBE, each user is issued a secret key that is associated with his/her identity. However, in order to achieve the key revocation mechanism, each user's secret key itself does not allow him/her to decrypt ciphertexts. To allow the users to decrypt, the key generation center (KGC) broadcasts *key updates* for every time period through a public channel. Roughly, the key update incorporates public information of the users that are currently allowed in the system. Therefore, although the key update is useless for already revoked users, it will allow non-revoked users to combine the key updates with their secret keys to derive a *decryption key*, which allow them to properly decrypt ciphertexts. To achieve a scaleable revocation mechanism, Boldyreva et al. utilized a subset cover framework called the complete subtree (CS) method [NNL01], so that the size of the key update sent by the KGC in each time period will be logarithmic in the number of system users. After the work of [BGK08], many works [CLL$^+$12a, IWS15, ISW17, LLP17, LV09, SE13b, SE14b, WES17] followed and their RIBE construction was also extended to revocable *hierarchical* IBE (RHIBE) that simultaneously supports scalable key revocation and key delegation functionalities [ESY16, Lee16, LP16, RLPL15, SE13a, SE14a, SE15, SE16].

Since RIBE and RHIBE were introduced by envisioning the real-world use of (H)IBE systems, their security definitions should take into account as many realistic threats and attack scenarios as possible. For example, leakage of decryption keys due to social/cyber attacks or unexpected human errors are common incidents in practice. Motivated by this, Seo and Emura [SE13b, SE14b, SE15, SE16] introduced a security notion for R(H)IBE called *decryption key exposure resistance* (DKER). Roughly speaking, this security notion guarantees that an exposure of a user's decryption key at some period will not compromise the confidentiality of ciphertexts that are encrypted for different time periods — a clearly desirable security guarantee in practice. After the introduction of the new security notion DKER, it has quickly become one of the default security requirements for R(H)IBE and attracted many followup works concerning R(H)IBE schemes with DKER [ESY16, IWS15, ISW17, Lee16, MLC$^+$15, LLP17, LP16, PLL15, PLL16, RLPL15, SE15, SE16, WES17]. However, so far all of the constructions of R(H)IBE schemes with DKER have been based on bilinear or multilinear maps.

Although lattice-based cryptography has been paid much attention in the last decade, mainly due to their quantum resistance, construction of R(H)IBE schemes with DKER has been rather unsuccessful. Chen et al. [CLL$^+$12b] proposed the first lattice-based RIBE scheme *without* DKER, which was a work before the notion of DKER was formalized by [SE13b, SE14b]. Recently, there was a followup work by Takayasu and Watanabe [TW17] who partially solved the open problem of achieving RIBE with DKER by proposing a variant of [CLL$^+$12b] that achieved *bounded* DKER. This is a strictly weaker notion than DKER which only allows a bounded number of decryption keys to be leaked. However, we still do not have any lattice-based RIBE scheme that fully achieves DKER. Furthermore, the situation on lattice-based RHIBE schemes is much grimmer since we do not know how to achieve them even in the weaker security model that does not require DKER.

One of the main reasons why we do not know how to construct R(H)IBE schemes with DKER is because the algebraic structure of lattices are ill-fit with the so-called *key re-randomization* property. Thus far, all RIBE schemes [IWS15, ISW17, LLP17, MLC$^+$15, PLL15, SE13b, SE14b, WES17] and RHIBE schemes with DKER [ESY16, Lee16, LP16, PLL16, RLPL15, SE15, SE16] are based on number theoretical assumptions, e.g., bilinear maps and multilinear maps, which they all rely heavily on this key re-randomization property. At a high level, this is the property with which each user can re-randomize his/her key so that the re-randomized key is distributed identically to (or at least statistically close to) a key generated using a fresh randomness. In essence, this is the unique property that enables us to achieve DKER. Furthermore, this property is also heavily utilized when generating his/her children's secret keys for fixed randomness without using any secret information, hence, achieving the hierarchal feature. However, unfortunately, due to the difference in the algebraic structure of bilinear, multilinear maps and lattices, we are currently unaware of any way of achieving the key re-randomization property from lattices.[1] Therefore, to construct lattice-based R(H)IBE schemes with DKER, it seems that we must deviate from prior methodologies and develop new techniques.

Our Contributions. In this paper, we propose the first lattice-based R(H)IBE scheme with DKER. Namely, our scheme is secure under the learning with errors (LWE) assumption. The techniques used in this work highly depart from previous works that rely on the key re-randomization property to achieve DKER and the key delegation functionalities. Specifically, we show a method to generically construct an RIBE scheme with DKER from any two-level standard HIBE scheme and RIBE scheme without DKER, thus bypassing the necessity of the key re-randomization property. Then, building on top of the idea of our generic construction, we further exploit the algebraic structure of lattices to construct an RHIBE scheme with DKER. We provide a brief summary of our work below and refer the detailed technical overview to Section 2.

Our first contribution is a generic construction of RIBE *with* DKER from any RIBE *without* DKER and two-level HIBE. The new tools we introduce to circumvent the necessity of the key re-randomization property are called *leveled ciphertexts* and *leveled decryption keys*. At a high level, each "level" for the leveled ciphertexts and decryption keys is associated to the RIBE scheme without DKER and the two-level HIBE scheme, respectively; one level is responsible for achieving the revocation mechanism and the other is responsible for the key re-randomization mechanism. Therefore, informally, our leveled structure allows for a *partial* key re-randomization mechanism. Using the lattice-based RIBE scheme without DKER of Chen et al. [CLL$^+$12b] and any lattice-based HIBE scheme, e.g., [ABB10, CHKP12], our result implies the first lattice-based RIBE scheme with DKER. Furthermore, since any IBE schemes can be converted to an HIBE scheme [DG17] (in the selective-identity model) and any RIBE scheme without DKER implies an IBE scheme, our result also implies a generic conversion of any RIBE scheme without DKER into an RIBE scheme with DKER.

Our second contribution is the construction of the first lattice-based RHIBE scheme with DKER. It is built on top of the idea of our generic construction and further exploits the algebraic structure unique to lattices. Namely, to achieve the key delegation functionality, i.e., hierarchal feature, we additionally introduce a new tool called *level conversion keys*. In essence, this tool enables a user to convert his (secret) decryption key to a (public) key update for users of different hierarchal levels. In other words, the level conversion key allows one to delegate his key to its children without requiring to re-randomize his key. Although the idea is simple, the concrete machinery to blend the level conversion keys securely into the construction is rather contrived and we refer the details to Section 2.

[1] If the reader is familiar with lattice-based cryptography, he/she may think that the existing RIBE schemes [CLL$^+$12b, TW17] can be easily modified to support the property by using short trapdoor bases. However, a simple modification is insufficient for R(H)IBE. We provide the detailed discussion of the difficulty in Section 2.

Finally, we state some side contributions worth highlighting in our paper. Firstly, we reformalize the syntax and security definitions for R(H)IBE. For instance, regarding the security definitions, since previous security definitions [BGK08, SE13b, SE14a, SE16] had some ambiguous treatments (e.g. in some cases it is not clear when the values such as secret keys and key updates, are generated during the security game), it was up to the readers to interpret the definitions and the proofs. Therefore, in our work we provide a refined security definition for R(H)IBE which in particular is a more rigorous and explicit treatment than the previous definitions. Secondly, we provide a formal treatment on an implicit argument that has been frequently adopted in the R(H)IBE literature. In particular, we introduce a simple yet handy "strategy-dividing lemma", which helps us simplify the security proofs for R(H)IBE schemes in general. For the details, see Section 4.

Related Works. Boldyreva et al. [BGK08] proposed the first RIBE scheme that achieved selective-identity security from bilinear maps and Libert and Vergnaud [LV09] extended their results to the adaptive setting. Chen et al. [CLL$^+$12a] proposed the first anonymous RIBE scheme. The first lattice-based RIBE scheme was proposed by Chen et al. [CLL$^+$12b] and the first RHIBE scheme was proposed by Seo and Emura [SE13a, SE14a] based on bilinear maps. Recently, Chang et al. [CCKS18] proposed an RIBE scheme from codes with rank metric in the random oracle model.

After Seo and Emura [SE13b, SE14b] introduced the notion of DKER and proposed the first RIBE scheme with DKER, several improvements and variants have been proposed. These works consist of RIBE [IWS15, ISW17, LLP17, WES17] and RHIBE [ESY16, Lee16, LP16, SE15, SE16] from bilinear maps, and those from multilinear maps [MLC$^+$15, PLL16, PLL15]. From lattices, Takayasu and Watanabe [TW17] proposed an RIBE scheme with bounded DKER; a strictly weaker notion then DKER.

Server-aided RIBE [CDLQ16, NWZ16, QDLL15] is a variant of RIBE where most of the computation of the users are delegated to an untrusted server. The revocation mechanism we study in this paper is sometimes referred to as indirect revocation. A direct revocation mechanism does not require key updates and has been discussed for attribute-based encryption [AI09a, AI09b] and predicate encryption [NMS12]. Recently, Ling et al. proposed the first lattice-based directly revocable predicate encryption scheme [LNWZ17] and its server-aided variant [LNWZ18].

Roadmap. In Section 2, we provide an overview of our constructions. In Section 3, we recall basic tools for lattice-based cryptography. In Section 4, we introduce formal definitions for RHIBE. In Section 5, we show a generic construction of RIBE with DKER. Finally, in Section 6, we show our main result concerning the first lattice-based RHIBE scheme with DKER.

Notations. Before diving into the technical details, we prepare some notations. Let $\mathbb{N}$ be the set of all natural numbers. For non-negative integers $n, n' \in \mathbb{N}$ with $n \leq n'$, we define $[n, n'] := \{n, n+1, \dots, n'\}$, and we extend the definition for $n > n'$ by $[n, n'] = \emptyset$. For notational convenience, for $n \in \mathbb{N}$, we define $[n] := [1, n]$. Throughout the paper, $\lambda \in \mathbb{N}$ denotes the security parameter.

As usual in the literature of (R)HIBE, an identity ID of a user at level ℓ in the hierarchy in an RHIBE scheme is expressed as a length-ℓ vector $\mathsf{ID} = (\mathsf{id}_1, \cdots, \mathsf{id}_\ell)$. In order not to mix up with an identity $\mathsf{ID} = (\mathsf{id}_1, \mathsf{id}_2, \dots)$ treated in an RHIBE scheme and its element id_i, we sometimes call the former a *hierarchical identity* and the latter an *element identity*. We refer to the set of all element identities as the *element identity space* and denote it by $\mathcal{ID}$. We assume that the element identity space is determined only by the security parameter λ. Thus, for example, the space to which level-ℓ identities belong is expressed as $(\mathcal{ID})^\ell$. For notational convenience, for $\ell \in \mathbb{N}$ we define $(\mathcal{ID})^{\leq \ell} := \bigcup_{i \in [\ell]} (\mathcal{ID})^i$, and the hierarchal identity space $\mathcal{ID}_{\mathtt{h}} := (\mathcal{ID})^{\leq L}$. We denote by "$\mathsf{kgc}$" the special hierarchical identity for the level-0 user, i.e., the key generation center (KGC).

Like an ordinary vector, we consider a prefix of hierarchical identities. For example, for a level-ℓ hierarchical identity $\mathsf{ID} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$ and $t \leq \ell$, $\mathsf{ID}_{[t]}$ represents the length-t prefix of ID, i.e., $\mathsf{ID}_{[t]} = (\mathsf{id}_1, \ldots, \mathsf{id}_t)$. We denote by "$\mathsf{pa}(\mathsf{ID})$" the identity of its parent (i.e. the direct ancestor), namely, if $\mathsf{ID} \in (\mathcal{ID})^\ell$, then $\mathsf{pa}(\mathsf{ID}) := \mathsf{ID}_{[\ell-1]} = (\mathsf{id}_1, \ldots, \mathsf{id}_{\ell-1})$, and $\mathsf{pa}(\mathsf{ID})$ for a level-1 identity $\mathsf{ID} \in \mathcal{ID}$ is defined to be kgc. Furthermore, we denote by "$\mathsf{prefix}(\mathsf{ID})$" the set consisting of itself and all of its ancestors, namely, $\mathsf{prefix}(\mathsf{ID}) := \{\mathsf{ID}_{[1]}, \mathsf{ID}_{[2]}, \ldots, \mathsf{ID}_{[|\mathsf{ID}|]} = \mathsf{ID}\}$. Also, for $\mathsf{ID} \in (\mathcal{ID})^\ell$, we denote by "$\mathsf{ID} \| \mathcal{ID}$" as the subset of $(\mathcal{ID})^{\ell+1}$ that contains all the members who have ID as its parent.

2 Technical Overview

In this section, we provide the technical overview of our results. In order to make the lattice-based RHIBE overview easier to follow, we present the details of our generic construction of RIBE with DKER using lattice terminologies in the selective-identity model. The general idea presented below translates naturally to our generic construction. To this end, we first prepare two standard hash functions used in lattice-based cryptography: one for the users $\mathsf{ID} \in \mathcal{ID}_{\mathtt{h}} = \mathcal{ID}^{\leq L}$, where each element identity space is defined by $\mathcal{ID} = \mathbb{Z}_q^n \setminus \{\mathbf{0}_n\}$, and another for the time period[2] $\mathsf{t} \in \mathcal{T} \subset \mathbb{Z}_q^n \setminus \{\mathbf{0}_n\}$. In particular, for a user $\mathsf{ID} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell) \in (\mathbb{Z}_q^n \setminus \{\mathbf{0}_n\})^{\leq L}$ and time period $\mathsf{t} \in \mathbb{Z}_q^n \setminus \{\mathbf{0}_n\}$ we use the following hash functions $\mathbf{E}(\cdot)$ and $\mathbf{F}(\cdot)$:

- $\mathbf{E}(\mathsf{ID}) := [\mathbf{B}_1 + H(\mathsf{id}_1)\mathbf{G} | \cdots | \mathbf{B}_\ell + H(\mathsf{id}_\ell)\mathbf{G}] \in \mathbb{Z}_q^{n \times \ell m}$,
- $\mathbf{F}(\mathsf{t}) := \mathbf{B}_{L+1} + H(\mathsf{t})\mathbf{G} \in \mathbb{Z}_q^{n \times m}$,

where $(\mathbf{B}_j)_{j \in [L+1]}$ are random matrices in $\mathbb{Z}_q^{n \times m}$ chosen at setup of the scheme and $\mathbf{G}$ is the gadget matrix [MP12]. Here, $H : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n \times n}$ is a specific hash function whose definition is provided in Section 3. Notice that for any $\mathsf{ID} \in (\mathbb{Z}_q^n \setminus \{\mathbf{0}_n\})^\ell$ and $\mathsf{id}_{\ell+1} \in \mathbb{Z}_q^n \setminus \{\mathbf{0}_n\}$, we have $\mathbf{E}(\mathsf{ID} \| \mathsf{id}_{\ell+1}) = [\mathbf{E}(\mathsf{ID}) | \mathbf{B}_{\ell+1} + H(\mathsf{id}_{\ell+1})\mathbf{G}]$. Finally, we define $\mathbf{E}(\mathsf{kgc}) := \emptyset$.

Review of RIBE *without* DKER. At first, we recall Chen et al.'s lattice-based RIBE scheme *without* DKER [CLL+12b] in Figure 1. Here, we see why the scheme realizes the revocation mechanism while it does not satisfy DKER. One feature of RIBE construction is that the KGC maintains a binary tree where each user is assigned to a randomly selected leaf. Furthermore, a random vector $\mathbf{u}_\theta \in \mathbb{Z}_q^n$ is uniquely assigned to each node θ of the binary tree. Then, we explain three types of keys which are core tools to realize the revocation mechanism. A *secret key* for a

$$
\boxed{
\begin{array}{ll}
\mathsf{PP} := (\mathbf{A}, \mathbf{u}, \text{hash functions } \mathbf{E}(\cdot), \mathbf{F}(\cdot)), & \mathsf{sk}_{\mathsf{kgc}} := \mathbf{T}_{\mathbf{A}} \\
\multicolumn{2}{l}{\mathsf{ct} := (c_0 := \mathbf{u}^\top \mathbf{s} + \mathsf{noise} + \mathsf{M} \left\lfloor \frac{q}{2} \right\rfloor, \mathbf{c}_1 := [\mathbf{A} | \mathbf{E}(\mathsf{ID}) | \mathbf{F}(\mathsf{t})]^\top \mathbf{s} + \mathsf{noise})} \\
\mathsf{sk}_{\mathsf{ID}} := (\mathbf{e}_{\mathsf{ID},\theta})_\theta & \text{s.t. } [\mathbf{A} | \mathbf{E}(\mathsf{ID})] \mathbf{e}_{\mathsf{ID},\theta} = \mathbf{u}_\theta \\
\mathsf{ku}_{\mathsf{t}} := (\mathbf{e}_{\mathsf{t},\theta})_\theta & \text{s.t. } [\mathbf{A} | \mathbf{F}(\mathsf{t})] \mathbf{e}_{\mathsf{t},\theta} = \mathbf{u} - \mathbf{u}_\theta \\
\mathsf{dk}_{\mathsf{ID},\mathsf{t}} := \mathbf{d}_{\mathsf{ID},\mathsf{t}} & \text{s.t. } [\mathbf{A} | \mathbf{E}(\mathsf{ID}) | \mathbf{F}(\mathsf{t})] \mathbf{d}_{\mathsf{ID},\mathsf{t}} = \mathbf{u}
\end{array}
}
$$

Figure 1: **Chen et al.'s RIBE Scheme**

[2] As we will show in Section 4, the time period space is a set of natural numbers $\{1, 2, \ldots\}$. Here, we assume that there is an efficient hash function that maps each natural number to a distinct vector in $\mathbb{Z}_q^n \setminus \{\mathbf{0}_n\}$.

user ID is a tuple of short vectors $\mathsf{sk}_{\mathsf{ID}} = (\mathbf{e}_{\mathsf{ID},\theta})_\theta$, where each short vector $\mathbf{e}_{\mathsf{ID},\theta}$ is associated to a random vector $\mathbf{u}_\theta$ such that

$$[\mathbf{A}|\mathbf{E}(\mathsf{ID})]\mathbf{e}_{\mathsf{ID},\theta} = \mathbf{u}_\theta.$$

Since $\mathbf{u}_\theta$ is an independent random vector and the ciphertext c_0 only depends on $\mathbf{u}$, $\mathbf{e}_{\mathsf{ID},\theta}$ in $\mathsf{sk}_{\mathsf{ID}}$ itself is useless for decrypting a ciphertext ct. Hence, in each time period the KGC broadcasts a *key update* which is also a tuple of short vectors $\mathsf{ku}_{\mathsf{t}} = (\mathbf{e}_{\mathsf{t},\theta})_\theta$, where each short vector $\mathbf{e}_{\mathsf{t},\theta}$ is associated to a random vector $\mathbf{u}_\theta$ such that

$$[\mathbf{A}|\mathbf{F}(\mathsf{t})]\mathbf{e}_{\mathsf{t},\theta} = \mathbf{u} - \mathbf{u}_\theta.$$

As same as above, $\mathbf{e}_{\mathsf{t},\theta}$ in ku_{t} itself is useless for decrypting a ciphertext ct. Now, we explain how the revocation mechanism works. By utilizing the complete subtree (CS) method [NNL01], the KGC is able to broadcast key updates so that there is no common node θ in ku_{t} and $\mathsf{sk}_{\mathsf{ID}}$ of *revoked* IDs, while there are at least one common node θ in ku_{t} and $\mathsf{sk}_{\mathsf{ID}}$ of *non-revoked* IDs. Then, $\mathbf{e}_{\mathsf{ID},\theta}$ in $\mathsf{sk}_{\mathsf{ID}}$ and $\mathbf{e}_{\mathsf{t},\theta}$ in ku_{t} of the common node θ enable a non-revoked ID to derive a well-formed *decryption key* which is a short vector satisfying

$$[\mathbf{A}|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]\mathbf{d}_{\mathsf{ID},\mathsf{t}} = \mathbf{u}.$$

It can be easily checked that $\mathbf{d}_{\mathsf{ID},\mathsf{t}}$ can be obtained by simply adding $\mathbf{e}_{\mathsf{ID},\theta}$ and $\mathbf{e}_{\mathsf{t},\theta}$ in a component-wise fashion. Also note that if $\mathbf{e}_{\mathsf{ID},\theta}$ and $\mathbf{e}_{\mathsf{t},\theta}$ are short vectors, then so is $\mathbf{d}_{\mathsf{ID},\mathsf{t}}$. Then, the vector enables us to recover the plaintext by computing

$$c_0 - \mathbf{c}_1^\top \mathbf{d}_{\mathsf{ID},\mathsf{t}} \approx \mathsf{M} \left\lfloor \frac{q}{2} \right\rfloor.$$

The main insight of this construction is that only non-revoked users can use the key updates to eliminate the random factor $\mathbf{u}_\theta$ to obtain a short vector $\mathbf{d}_{\mathsf{ID},\mathsf{t}}$ that is bound to the the public matrix $[\mathbf{A}|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]$ and public vector $\mathbf{u}$ for which the ciphertexts ct are created with.

Although the scheme is proven to be a secure RIBE scheme *without* DKER, it clearly does not satisfy DKER. Indeed, there is a concrete attack even with a single decryption key query on the target ID^*. The attack is as follows: assume that the adversary obtains a decryption key $\mathsf{dk}_{\mathsf{ID}^*,\mathsf{t}}$ for the target ID^* and a time period $\mathsf{t} \neq \mathsf{t}^*$. Since key updates are publicly broadcast, the adversary also obtains ku_{t} and $\mathsf{ku}_{\mathsf{t}^*}$. Since user ID^* will not be revoked unless $\mathsf{sk}_{\mathsf{ID}^*}$ was revealed to the adversary, the key updates ku_{t} and $\mathsf{ku}_{\mathsf{t}^*}$ will share a common node θ^* with the secret key.[3] Therefore, recalling that $\mathsf{dk}_{\mathsf{ID}^*,\mathsf{t}}$ was a simple component-wise addition of $\mathbf{e}_{\mathsf{ID}^*,\theta^*}$ in $\mathsf{sk}_{\mathsf{ID}^*}$ and $\mathbf{e}_{\mathsf{t},\theta^*}$ in ku_{t}, $\mathcal{A}$ can first recover the secret key component $\mathbf{e}_{\mathsf{ID}^*,\theta^*}$ from $(\mathsf{dk}_{\mathsf{ID}^*,\mathsf{t}}, \mathbf{e}_{\mathsf{t},\theta^*})$, which he can then combine it with $\mathbf{e}_{\mathsf{t}^*,\theta^*}$ in $\mathsf{ku}_{\mathsf{t}^*}$ to create the decryption key $\mathbf{d}_{\mathsf{ID}^*,\mathsf{t}^*}$ for the challenge time period t^*. Specifically, this decryption key allows the adversary to completely break the scheme. In reality, this corresponds to the fact that once a decryption key for a certain time period is exposed to an adversary, then all the messages of distinct time periods may also be compromised. In essence, this attack relies on the fact that the decryption key leaks partial information on the secret key, which can then be used to construct decryption keys of all distinct time periods.

In all the previous bilinear map-based constructions, the above problem was circumvented by relying on the so-called *key re-randomization property.* Informally, this property allows one to

[3] To be more precise, there are cases ku_{t} and $\mathsf{ku}_{\mathsf{t}^*}$ might not share a common node, however, $\mathcal{A}$ can always adaptively revoke other users so that this holds.

re-randomize the decryption key, hence even if the decryption key is leaked, it would be impossible to restore the original secret key. In the above construction, this idea would correspond to re-sampling a short random vector $\bar{\mathbf{d}}_{\mathsf{ID},\mathsf{t}}$ such that

$$[\mathbf{A}|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]\bar{\mathbf{d}}_{\mathsf{ID},\mathsf{t}} = \mathbf{u}$$

using his original decryption key $\mathbf{d}_{\mathsf{ID},\mathsf{t}}$. Indeed, if the distribution of $\bar{\mathbf{d}}_{\mathsf{ID},\mathsf{t}}$ is independent of the original decryption key $\mathbf{d}_{\mathsf{ID},\mathsf{t}}$, this modification would prevent the above attack, since the adversary will not be able to recover the secret key component $\mathbf{e}_{\mathsf{ID}^*,\theta^*}$ anymore using the above strategy. However, such a re-sampling procedure is computationally infeasible, since otherwise we would be able to trivially solve the small integer solution (SIS) problem.

Readers familiar with lattice-based constructions of (non-revocable) HIBE may think that we may achieve the key re-randomization property by simply using a short trapdoor basis as the secret key instead of a vector. Indeed, if we add a short trapdoor basis $\mathbf{T}_{[\mathbf{A}|\mathbf{E}(\mathsf{ID})]}$ as a part of the secret key $\mathsf{sk}_{\mathsf{ID}}$, the user ID will be able to sample a short vector $\bar{\mathbf{d}}_{\mathsf{ID},\mathsf{t}} \neq \mathbf{d}_{\mathsf{ID},\mathsf{t}}$, since anybody can efficiently extend the trapdoor basis $\mathbf{T}_{[\mathbf{A}|\mathbf{E}(\mathsf{ID})]}$ to $\mathbf{T}_{[\mathbf{A}|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]}$ and thus sample a random vector $\bar{\mathbf{d}}_{\mathsf{ID},\mathsf{t}}$ such that $[\mathbf{A}|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]\bar{\mathbf{d}}_{\mathsf{ID},\mathsf{t}} = \mathbf{u}$. However, this approach does not mesh well with the above revocation mechanism, since now the user ID can derive decryption keys $\mathbf{d}_{\mathsf{ID},\mathsf{t}}$ for every time period without requiring the key updates ku_{t}. Therefore, adding a short trapdoor basis to the secret key is too powerful and we completely lose the mechanism for supporting revocation.

Constructing RIBE *with* DKER. To summarize thus far, the main bottleneck of Chen et al.'s RIBE scheme without DKER is that it satisfies the key revocation mechanism, but seems challenging to extend it to satisfy DKER. On the other hand, adding a short trapdoor basis would definitely be useful for achieving DKER, however, it seems to contradict with the revocation mechanism. In the following, we show that we can carefully combine these two seemingly conflicting ideas together. The concrete construction of our lattice-based RIBE scheme *with* DKER is illustrated in Figure 2. The boxed items denote the changes made from the previous figure.

$$\begin{array}{ll}
\mathsf{PP} := (\mathbf{A}, \boxed{\bar{\mathbf{A}}}, \mathbf{u}, \text{hash functions } \mathbf{E}(\cdot), \mathbf{F}(\cdot)), & \mathsf{sk}_{\mathsf{kgc}} := (\mathbf{T}_{\mathbf{A}}, \boxed{\mathbf{T}_{\bar{\mathbf{A}}}}) \\
\mathsf{ct} := \left(\begin{array}{c} c_0 := \mathbf{u}^\top(\mathbf{s} + \boxed{\bar{\mathbf{s}}}) + \mathsf{noise} + \mathsf{M}\left\lfloor \frac{q}{2} \right\rfloor, \\ \mathbf{c}_1 := [\mathbf{A}|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]^\top\mathbf{s} + \mathsf{noise}, \ \boxed{\bar{\mathbf{c}}_1 := [\bar{\mathbf{A}}|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]^\top\bar{\mathbf{s}} + \mathsf{noise}} \end{array} \right) & \\
\mathsf{sk}_{\mathsf{ID}} := \left((\mathbf{e}_{\mathsf{ID},\theta})_\theta, \boxed{\mathbf{T}_{[\bar{\mathbf{A}}|\mathbf{E}(\mathsf{ID})]}}\right) & \text{s.t. } [\mathbf{A}|\mathbf{E}(\mathsf{ID})]\mathbf{e}_{\mathsf{ID},\theta} = \mathbf{u}_\theta \\
\mathsf{ku}_{\mathsf{t}} := (\mathbf{e}_{\mathsf{t},\theta})_\theta & \text{s.t. } [\mathbf{A}|\mathbf{F}(\mathsf{t})]\mathbf{e}_{\mathsf{t},\theta} = \mathbf{u} - \mathbf{u}_\theta \\
\mathsf{dk}_{\mathsf{ID},\mathsf{t}} := \left(\mathbf{d}_{\mathsf{ID},\mathsf{t}}, \boxed{\bar{\mathbf{d}}_{\mathsf{ID},\mathsf{t}}}\right) & \text{s.t. } [\mathbf{A}|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]\mathbf{d}_{\mathsf{ID},\mathsf{t}} = \mathbf{u}, \ \boxed{[\bar{\mathbf{A}}|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]\bar{\mathbf{d}}_{\mathsf{ID},\mathsf{t}} = \mathbf{u}}
\end{array}$$

Figure 2: **Our RIBE Scheme with DKER**

Our construction relies on a tool we call *leveled ciphertexts* and *leveled decryption keys*; the terminology should become more intuitive and helpful in the hierarchical setting that will be explained later. Here, we call an element associated with a matrix $\mathbf{A}$ and $\bar{\mathbf{A}}$ level-1 and level-2, respectively. In particular, $\mathbf{c}_1, \bar{\mathbf{c}}_1$ and $\mathbf{d}_{\mathsf{ID},\mathsf{t}}, \bar{\mathbf{d}}_{\mathsf{ID},\mathsf{t}}$ in Figure 2 are the level-1, level-2 ciphertexts and decryption keys, respectively. Here, the level-1 components $\mathbf{c}_1$ and $\mathbf{d}_{\mathsf{ID},\mathsf{t}}$ correspond to Chen et al.'s RIBE scheme without DKER and are responsible for achieving the revocation mechanism. On the other hand, the level-2 components $\bar{\mathbf{c}}_1$ and $\bar{\mathbf{d}}_{\mathsf{ID},\mathsf{t}}$ are the newly introduced elements that will help us achieve DKER. Since the two decryption keys for levels-1 and 2 are in one-to-one correspondence with the ciphertexts $(\mathbf{c}_1, \bar{\mathbf{c}}_1)$ for levels-1 and 2, both of the decryption keys are required to recover the underlying message as follows:

$$c_0 - \underbrace{\mathbf{c}_1^\top \mathbf{d}_{\mathsf{ID},\mathsf{t}}}_{\text{level-1 component}} - \underbrace{\bar{\mathbf{c}}_1^\top \bar{\mathbf{d}}_{\mathsf{ID},\mathsf{t}}}_{\text{level-2 component}} \approx \mathsf{M} \left\lfloor \frac{q}{2} \right\rfloor .$$

In particular, if either level of the decryption key is missing, one will not be able to recover the message. Separating the role of the decryption keys is the main idea that allows us to associate the two seemingly conflicting properties of revocation and key re-randomization to each level of the decryption keys.

First, we observe that the above RIBE scheme achieves the revocation mechanism since it simply inherits this property from the underlying Chen et al.'s RIBE scheme without DKER. Furthermore, we achieve DKER by incorporating the aforementioned trapdoor idea; we add a trapdoor $\mathbf{T}_{[\bar{\mathbf{A}}|\mathbf{E}(\mathsf{ID})]}$ to the secret key $\mathsf{sk}_{\mathsf{ID}}$. Using this short trapdoor basis $\mathbf{T}_{[\bar{\mathbf{A}}|\mathbf{E}(\mathsf{ID})]}$, we can now sample a level-2 decryption key $\bar{\mathbf{d}}_{\mathsf{ID},\mathsf{t}}$ for each time period independently from the previous time periods. Namely, using $\mathbf{T}_{[\bar{\mathbf{A}}|\mathbf{E}(\mathsf{ID})]}$, we can sample a short vector $\bar{\mathbf{d}}_{\mathsf{ID},\mathsf{t}}$ such that

$$[\bar{\mathbf{A}}|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]\bar{\mathbf{d}}_{\mathsf{ID},\mathsf{t}} = \mathbf{u},$$

where $\bar{\mathbf{d}}_{\mathsf{ID},\mathsf{t}}$ leaks no information of the secret key $\mathsf{sk}_{\mathsf{ID}}$. Hence, although we are not able to completely re-randomize the decryption key $\mathsf{dk}_{\mathsf{ID},\mathsf{t}} = (\mathbf{d}_{\mathsf{ID},\mathsf{t}}, \bar{\mathbf{d}}_{\mathsf{ID},\mathsf{t}})$, we can *partially* re-randomize the decryption key by sampling a new level-2 decryption key $\bar{\mathbf{d}}_{\mathsf{ID},\mathsf{t}}$ for each time period; even if $\mathsf{dk}_{\mathsf{ID},\mathsf{t}}$ is compromised, this alone will not be sufficient for constructing decryption keys for other time periods. Indeed, we show that this partial key re-randomization property is sufficient to prove the DKER security.

In Section 5, we formalize and prove the above idea by providing a generic construction of RIBE with DKER, using as building blocks any RIBE without DKER and 2-level HIBE. At a high level, the 2-level HIBE scheme is responsible for the key re-randomization property and is the core component that allows us to convert non-DKER secure RIBE schemes into DKER secure RIBE schemes.

Constructing RHIBE from Lattices. Next, we show an overview of our lattice-based RHIBE construction. For simplicity of presentation, we do not take into account DKER in the following technical overview. Hence, we explain how to construct an RHIBE scheme by modifying Chen et al.'s RIBE scheme.

Before getting into detail, we prepare some notations used for the hierarchal setting. In the following, let L be the maximum depth of the hierarchy, where we treat the KGC as level-0. In RHIBE, all level-i users ID for $i \in [0, L-1]$, including the KGC, maintain a binary tree $\mathsf{BT}_{\mathsf{ID}}$ to manage their children users in $\mathsf{ID}\|\mathcal{ID}$. Furthermore, a random vector $\mathbf{u}_{\mathsf{ID},\theta} \in \mathbb{Z}_q^n$ is uniquely assigned to each node θ of the binary tree $\mathsf{BT}_{\mathsf{ID}}$. The level-$(\ell-1)$ user $\mathsf{pa}(\mathsf{ID})$ creates the secret key $\mathsf{sk}_{\mathsf{ID}}$ of the level-ℓ user ID, and the user ID derives his own decryption key $\mathsf{dk}_{\mathsf{ID},\mathsf{t}}$ by combining his own secret key $\mathsf{sk}_{\mathsf{ID}}$ and the key updates $\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}$ that are broadcast by the parent user $\mathsf{pa}(\mathsf{ID})$. Throughout the overview, we assume ID represents an level-ℓ user.

At first, we briefly explain the difficulty for constructing a lattice-based RHIBE scheme. For the purpose, the simplest approach should be extending a hash function $\mathbf{E}(\cdot)$ to encode $\mathsf{ID} \in \mathcal{ID}_{\mathtt{h}} = \mathcal{ID}^{\leq L}$. However, it seems difficult for constructing an RHIBE scheme without further modifications. As Chen et al.'s RIBE scheme, if we use a set of short vectors $(\mathbf{e}_{\mathsf{ID},\theta})_\theta$ as $\mathsf{sk}_{\mathsf{ID}}$, the scheme does not support the key delegation property since level-$(\ell-1)$ users $\mathsf{pa}(\mathsf{ID})$ cannot create $\mathsf{sk}_{\mathsf{ID}}$ of level-ℓ users. On the other hand, if we use a short trapdoor basis $\mathbf{T}_{[\mathbf{A}|\mathbf{E}(\mathsf{ID})]}$ as

$\mathsf{sk}_{\mathsf{ID}}$, the scheme does not satisfy the key revocation mechanism since now the user ID can derive decryption keys $\mathbf{d}_{\mathsf{ID},\mathsf{t}}$ for every time period without requiring the key updates $\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}$. Hence, the key delegation functionality and the key revocation mechanism does not seem to blend well.

Introducing Leveld Secret Keys: Due to the complex nature of our scheme, we believe it to be helpful to provide the intuition of our scheme following a series of modifications, where our final scheme without DKER is depicted in Figure 6. Our starting point is illustrated in Figure 3, where as before, the box indicates the changes made from the prior scheme.

$$\begin{array}{ll}
\mathsf{PP} := (\boxed{(\mathbf{A}_i)_{i\in[L]}}, \mathbf{u}, \text{hash functions } \mathbf{E}(\cdot), \mathbf{F}(\cdot)), & \mathsf{sk}_{\mathsf{kgc}} := \boxed{(\mathbf{T}_{\mathbf{A}_i})_{i\in[L]}} \\
\multicolumn{2}{l}{\mathsf{ct} := \left(c_0 := \mathbf{u}^\top \mathbf{s} + \mathsf{noise} + \mathsf{M}\left\lfloor \frac{q}{2} \right\rfloor, \mathbf{c}_1 := [\boxed{\mathbf{A}_\ell}|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]^\top \mathbf{s} + \mathsf{noise}\right)} \\
\mathsf{sk}_{\mathsf{ID}} := \left((\mathbf{e}_{\mathsf{ID},\theta})_\theta, \boxed{(\mathbf{T}_{[\mathbf{A}_i|\mathbf{E}(\mathsf{ID})]})_{i\in[\ell+1,L]}}\right) & \text{s.t. } [\boxed{\mathbf{A}_\ell}|\mathbf{E}(\mathsf{ID})]\mathbf{e}_{\mathsf{ID},\theta} = \mathbf{u}_{\mathsf{pa}(\mathsf{ID}),\theta} \\
\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}} := (\mathbf{e}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t},\theta})_\theta & \text{s.t. } [\boxed{\mathbf{A}_\ell}|\mathbf{E}(\mathsf{pa}(\mathsf{ID}))|\mathbf{F}(\mathsf{t})]\mathbf{e}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t},\theta} = \mathbf{u} - \mathbf{u}_{\mathsf{pa}(\mathsf{ID}),\theta} \\
\mathsf{dk}_{\mathsf{ID},\mathsf{t}} := \mathbf{d}_{\mathsf{ID},\mathsf{t}} & \text{s.t. } [\boxed{\mathbf{A}_\ell}|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]\mathbf{d}_{\mathsf{ID},\mathsf{t}} = \mathbf{u}
\end{array}$$

Figure 3: **Leveled Secret Key and i-Leveled Ciphertext**

Towards resolving the incompatibility of the key delegation property and the key revocation mechanism, the scheme in Figure 3 utilizes leveled ciphertexts as the prior non-hierarchal scheme in Figure 2. Furthermore, we introduce a new tool called *leveled secret keys* in this scheme. Here, we call an element associated with a matrix $\mathbf{A}_i$ level-i, respectively. In particular, the ciphertext ct of a level-ℓ user ID is a level-ℓ ciphertext since $\mathbf{c}_1$ is associated with $\mathbf{A}_\ell$. The main trick of the scheme in Figure 3 is that a secret key $\mathsf{sk}_{\mathsf{ID}}$ for a level-ℓ user consists of level-i secret keys for $i \in [\ell, L]$, where the level-ℓ secret key and the other level-i secret keys for $i \in [\ell+1, L]$ serve a different purpose. The level-ℓ secret key in $\mathsf{sk}_{\mathsf{ID}}$ is a tuple of short vectors of the form $(\mathbf{e}_{\mathsf{ID},\theta})_\theta$ each of which satisfies

$$[\mathbf{A}_\ell|\mathbf{E}(\mathsf{ID})]\mathbf{e}_{\mathsf{ID},\theta} = [\mathbf{A}_\ell|\mathbf{E}(\mathsf{pa}(\mathsf{ID}))|\mathbf{B}_\ell + H(\mathsf{id}_\ell)\mathbf{G}]\mathbf{e}_{\mathsf{ID},\theta} = \mathbf{u}_{\mathsf{pa}(\mathsf{ID}),\theta},$$

and serves the same purpose as the original Chen et al.'s RIBE scheme. Namely, the level-ℓ secret key of a level-ℓ user is used for decrypting its own level-ℓ ciphertext, where the detailed procedure will be explained later. The remaining level-i secret keys in $\mathsf{sk}_{\mathsf{ID}}$ for $i \in [\ell+1, L]$ are trapdoors of the form $\mathbf{T}_{[\mathbf{A}_i|\mathbf{E}(\mathsf{ID})]}$ in $\mathsf{sk}_{\mathsf{ID}}$ and serves the purpose of delegation. Concretely, using the trapdoor $\mathbf{T}_{[\mathbf{A}_i|\mathbf{E}(\mathsf{ID})]}$ for $i \in [\ell+1, L]$, the level-ℓ user ID can sample all level-i secret keys for his children $\mathsf{ID}\|\mathsf{id}_{\ell+1} \in \mathsf{ID}\|\mathcal{ID}$; a set of short vectors $(\mathbf{e}_{\mathsf{ID}\|\mathsf{id}_{\ell+1},\theta})_\theta$ such that $[\mathbf{A}_i|\mathbf{E}(\mathsf{ID}\|\mathsf{id}_{\ell+1})]\mathbf{e}_{\mathsf{ID}\|\mathsf{id}_{\ell+1},\theta} = \mathbf{u}_{\mathsf{ID},\theta}$ and trapdoors $\mathbf{T}_{[\mathbf{A}_i|\mathbf{E}(\mathsf{ID}\|\mathsf{id}_{\ell+1})]}$ for $i \in [\ell+2, L]$. In addition, the level-ℓ user ID also uses the level-$(\ell+1)$ trapdoor $\mathbf{T}_{[\mathbf{A}_{\ell+1}|\mathbf{E}(\mathsf{ID})]}$ in $\mathsf{sk}_{\mathsf{ID}}$ to derive key updates $\mathsf{ku}_{\mathsf{ID},\mathsf{t}}$, where level-$(\ell-1)$ user $\mathsf{pa}(\mathsf{ID})$'s key update $\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}$ is a tuple of short vectors $(\mathbf{e}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t},\theta})_\theta$ such that

$$[\mathbf{A}_\ell|\mathbf{E}(\mathsf{pa}(\mathsf{ID}))|\mathbf{F}(\mathsf{t})]\mathbf{e}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t},\theta} = \mathbf{u} - \mathbf{u}_{\mathsf{pa}(\mathsf{ID}),\theta}.$$

Then, the level-ℓ user ID is able to derive a well-formed decryption key $\mathsf{dk}_{\mathsf{ID},\mathsf{t}}$ which is a short vector of the form $\mathbf{d}_{\mathsf{ID},\mathsf{t}}$ which satisfies

$$[\mathbf{A}|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]\mathbf{d}_{\mathsf{ID},\mathsf{t}} = [\mathbf{A}|\mathbf{E}(\mathsf{pa}(\mathsf{ID}))|\mathbf{B}_\ell + H(\mathsf{id}_\ell)\mathbf{G}|\mathbf{F}(\mathsf{t})]\mathbf{d}_{\mathsf{ID},\mathsf{t}} = \mathbf{u}.$$

Hence, the scheme in Figure 3 properly supports the key delegation functionality.

Furthermore, the scheme also supports the key revocation mechanism *at first glance.* The level-ℓ secret key $(\mathbf{e}_{\mathsf{ID},\theta})_\theta$ of the level-ℓ user ID allows the above scheme to achieve the revocation mechanism in the sense that ID will not be able to decrypt his level-ℓ ciphertext without his parent's key update $\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}$. Note that this is the exact same reason why the original Chen et al.'s RIBE scheme achieved the revocation mechanism. However, this scheme is flawed and does not meet the standard security notion of RHIBE, where we require the user ID to be revoked once any of his ancestors $\mathsf{ID}_{[i]} \in \mathsf{prefix}(\mathsf{ID})$ for $i \in [\ell - 1]$ is revoked. In other words, we require that once a user in a hierarchy is revoked then that user and all of its descendants should be revoked from the system. It can be easily checked that since the level-ℓ user ID is provided with the full trapdoor $\mathbf{T}_{[\mathbf{A}_i|\mathbf{E}(\mathsf{ID})]}$ for $i \in [\ell + 1, L]$ as part of its secret key, nothing is preventing the ID from continuing on generating secret keys and key updates for his children. Hence the above scheme is not a secure RHIBE scheme.

Introducing Leveled Decryption Keys: For the above hierarchical revocation requirement to hold, we further modify the scheme as in Figure 4. From now on, we further modify the definition of level-i ciphertext, and call a tuple

$$\begin{array}{ll}
\mathsf{PP} := ((\mathbf{A}_i)_{i\in[L]}, \mathbf{u}, \text{hash functions } \mathbf{E}(\cdot), \mathbf{F}(\cdot)), & \mathsf{sk}_{\mathsf{kgc}} := (\mathbf{T}_{\mathbf{A}_i})_{i\in[L]} \\
\mathsf{ct} := \big(\ c_0 := \mathbf{u}^\top \boxed{(\mathbf{s}_1 + \cdots \mathbf{s}_\ell)} + \mathsf{noise} + \mathsf{M}\left\lfloor \frac{q}{2} \right\rfloor, & \boxed{(\mathbf{c}_i := [\mathbf{A}_i|\mathbf{E}(\mathsf{ID}_{[i]})|\mathbf{F}(\mathsf{t})]^\top \mathbf{s}_i + \mathsf{noise})_{i\in[\ell]}}\ \big) \\
\mathsf{sk}_{\mathsf{ID}} := ((\mathbf{e}_{\mathsf{ID},\theta}), (\mathbf{T}_{[\mathbf{A}_i|\mathbf{E}(\mathsf{ID})]})_{i\in[\ell+1,L]}) & \text{s.t. } [\mathbf{A}_\ell|\mathbf{E}(\mathsf{ID})]\mathbf{e}_{\mathsf{ID},\theta} = \mathbf{u}_\theta \\
\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}} := ((\mathbf{e}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t},\theta}), \boxed{(\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t}})_{i\in[\ell-1]}}) & \text{s.t. } [\mathbf{A}_\ell|\mathbf{E}(\mathsf{pa}(\mathsf{ID}))|\mathbf{F}(\mathsf{t})]\mathbf{e}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t},\theta} = \mathbf{u} - \mathbf{u}_\theta, \\
 & \boxed{[\mathbf{A}_i|\mathbf{E}(\mathsf{ID}_{[i]})|\mathbf{F}(\mathsf{t})]\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t}} = \mathbf{u}} \\
\mathsf{dk}_{\mathsf{ID},\mathsf{t}} := \big(\mathbf{d}_{\mathsf{ID},\mathsf{t}}, \boxed{(\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t}})_{i\in[\ell-1]}}\big) & \text{s.t. } [\mathbf{A}_\ell|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]\mathbf{d}_{\mathsf{ID},\mathsf{t}} = \mathbf{u}
\end{array}$$

Figure 4: **Multiple Leveled Ciphertext and Key Update**

$$(\mathbf{u}^\top \mathbf{s}_i + \mathsf{noise}, \quad \mathbf{c}_i = [\mathbf{A}_i|\mathbf{E}(\mathsf{ID}_{[i]})|\mathbf{F}(\mathsf{t})]^\top \mathbf{s}_i + \mathsf{noise})$$

a level-i ciphertext since $\mathbf{c}_i$ is associated with the public matrix $\mathbf{A}_i$ and both components are associated with the same secret vector $\mathbf{s}_i$. Thus, we modify the ciphertext for a level-ℓ user ID to contain all the level-i ciphertexts for $i \in [\ell]$, where each level-i ciphertext is associated with the public matrix $\mathbf{A}_i$ and an identity $\mathsf{ID}_{[i]}$. The idea behind this modification is to revoke any user ID whose ancestors were revoked by including some information specific to the ancestors in the ciphertext. In particular, if some ancestor at level $i \in [\ell - 1]$ were to be revoked, then the level-i ciphertext $\mathbf{c}_i$ should become undecryptable, hence maintaining the secrecy of the plaintext M. To make this idea work, we must now provide user ID with new components to allow decryption of the level-i ciphertexts for $i \in [\ell - 1]$. We achieve this by introducing a new tool called *leveled decryption keys*. A leveled decryption key for a level-ℓ user ID consists of level-i decryption keys for $i \in [\ell]$. Similarly to leveled secret keys, the level-ℓ decryption key and the other level-i decryption keys for $i \in [\ell - 1]$ serve a slightly different purpose. The level-ℓ decryption key is denoted as $\mathbf{d}_{\mathsf{ID},\mathsf{t}}$ in $\mathsf{dk}_{\mathsf{ID},\mathsf{t}}$, and is created and serves the same purpose as in the previous schemes. The level-i decryption key for $i \in [\ell - 1]$ is denoted as $\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t}}$ in $\mathsf{dk}_{\mathsf{ID},\mathsf{t}}$ and is the actual decryption key used by its ancestor at level-i; although we use a different notation, $\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t}}$ is equivalent to $\mathbf{d}_{\mathsf{ID}_{[i]},\mathsf{t}}$ such that

$$[\mathbf{A}_i|\mathbf{E}(\mathsf{ID}_{[i]})|\mathbf{F}(\mathsf{t})]\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t}} = [\mathbf{A}_i|\mathbf{E}(\mathsf{ID}_{[i]})|\mathbf{F}(\mathsf{t})]\mathbf{d}_{\mathsf{ID}_{[i]},\mathsf{t}} = \mathbf{u}.$$

In particular, each ancestor at level-i for $i \in [\ell - 1]$ broadcasts their own decryption key $\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t}}$ (See $\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}$ in Figure 4) and the user ID sets the level-i decryption key for $i \in [\ell - 1]$ as $\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t}}$.

It can be easily verified that user ID can correctly decrypt his ciphertext as follows:

$$c_0 - \underbrace{\mathbf{c}_\ell^\top \mathbf{d}_{\mathsf{ID},\mathsf{t}}}_{\text{level-}\ell\text{ component}} - \sum_{i=2}^{\ell} \underbrace{\mathbf{c}_i^\top \mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t}}}_{\text{level-}i\text{ component}} \approx \mathsf{M} \left\lfloor \frac{q}{2} \right\rfloor .$$

However, this scheme is obviously insecure, since the level-i ancestors are required to publicly broadcast their level-i decryption key $\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t}}(= \mathbf{d}_{\mathsf{ID}_{[i]},\mathsf{t}})$, which can in turn be used by anybody to decrypt the level-i ciphertexts of that particular ancestor.

Making the Levels Two-Dimensional: For the scheme in Figure 4 to be secure, decryption keys of the ancestors should not be made public via the key updates. Specifically, a ciphertext aimed for a user should not contain the same level as of his ancestors, since otherwise the decryption keys of the ancestors must be made public. For the purpose, we further modify the scheme as in Figure 5. To this end, we incorporate multiple public vectors $(\mathbf{u}_k)_{k\in[L]}$, and redefine the notion of leveled ciphertexts and leveled decryption keys to be *two-dimensional.* Here, we call an element associated with a vector $\mathbf{u}_k$ and a matrix $\mathbf{A}_i$ level-(k, i), respectively. For example, we call a tuple

$$\mathsf{PP} := ((\mathbf{A}_i)_{i\in[L]}, \boxed{(\mathbf{u}_k)_{k\in[L]}}, \text{hash functions } \mathbf{E}(\cdot), \mathbf{F}(\cdot)), \qquad \mathsf{sk}_{\mathsf{kgc}} := (\mathbf{T}_{\mathbf{A}_i})_{i\in[L]}$$

$$\mathsf{ct} := \Big(\; c_0 := \boxed{\mathbf{u}_\ell}^\top (\mathbf{s}_1 + \cdots \mathbf{s}_\ell) + \mathsf{noise} + \mathsf{M} \left\lfloor \tfrac{q}{2} \right\rfloor, \; (\mathbf{c}_i := [\mathbf{A}_i | \mathbf{E}(\mathsf{ID}_{[i]}) | \mathbf{F}(\mathsf{t})]^\top \mathbf{s}_i + \mathsf{noise})_{i\in[\ell]} \; \Big)$$

$$\mathsf{sk}_{\mathsf{ID}} := ((\mathbf{e}_{\mathsf{ID},\theta}), (\mathbf{T}_{[\mathbf{A}_i | \mathbf{E}(\mathsf{ID})]})_{i\in[\ell+1,L]}) \qquad \text{s.t. } [\mathbf{A}_\ell | \mathbf{E}(\mathsf{ID})] \mathbf{e}_{\mathsf{ID},\theta} = \mathbf{u}_\theta,$$

$$\mathsf{ku}_{\mathsf{pa(ID)},\mathsf{t}} := ((\mathbf{e}_{\mathsf{pa(ID)},\mathsf{t},\theta}), \boxed{(\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},k})_{i\in[\ell-1],k\in[\ell,L]}})$$

$$\text{s.t. } [\mathbf{A}_\ell | \mathbf{E}(\mathsf{pa(ID)}) | \mathbf{F}(\mathsf{t})] \mathbf{e}_{\mathsf{pa(ID)},\mathsf{t},\theta} = \mathbf{u}_\ell - \mathbf{u}_\theta, \qquad \boxed{[\mathbf{A}_i | \mathbf{E}(\mathsf{ID}_{[i]}) | \mathbf{F}(\mathsf{t})] \mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},k} = \mathbf{u}_k}$$

$$\mathsf{dk}_{\mathsf{ID},\mathsf{t}} := (\mathbf{d}_{\mathsf{ID},\mathsf{t}}, \boxed{(\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell})_{i\in[\ell-1]}}) \qquad \text{s.t. } [\mathbf{A}_\ell | \mathbf{E}(\mathsf{ID}) | \mathbf{F}(\mathsf{t})] \mathbf{d}_{\mathsf{ID},\mathsf{t}} = \boxed{\mathbf{u}_\ell}$$

Figure 5: (k, i)-**Leveled Ciphertext and Decryption Key**

$$(\mathbf{u}_k^\top \mathbf{s}_i + \mathsf{noise}, \quad \mathbf{c}_i = [\mathbf{A}_i | \mathbf{E}(\mathsf{ID}_{[i]}) | \mathbf{F}(\mathsf{t})]^\top \mathbf{s}_i + \mathsf{noise})$$

a level-(k, i) ciphertext since the first component is associated with the public vector $\mathbf{u}_k$, and the latter component $\mathbf{c}_i$ is associated with the public matrix $\mathbf{A}_i$, and both components are associated with the same secret vector $\mathbf{s}_i$. In particular, a ciphertext for a level-ℓ user ID consists of level-(ℓ, i) ciphertexts for $i \in [\ell]$. Accordingly, we must provide user ID with a redefined leveled decryption key to allow decryption of the two-dimensional leveled ciphertexts. Specifically, we provide a level-ℓ user ID with level-(ℓ, i) decryption keys for $i \in [\ell]$, where again the level-(ℓ, ℓ) decryption key and the other level-(ℓ, i) decryption keys for $i \in [\ell - 1]$ are defined slightly differently. The level-(ℓ, ℓ) decryption key is denoted as $\mathbf{d}_{\mathsf{ID},\mathsf{t}}$, and is constructed and serves the exact same purpose as in the previous scheme. The level-(ℓ, i) decryption keys for $i \in [\ell - 1]$ are denoted as $\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell}$. As before, these decryption keys $\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell}$ are broadcast as part of the parent's key updates $\mathsf{ku}_{\mathsf{pa(ID)},\mathsf{t}}$, while the way they are defined is slightly different from the previous scheme. Namely, the level-(ℓ, i) decryption key $\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell}$ satisfies

$$[\mathbf{A}_i | \mathbf{E}(\mathsf{ID}_{[i]}) | \mathbf{F}(\mathsf{t})] \mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell} = \mathbf{u}_\ell,$$

Using this, a level-ℓ user ID can decrypt its ciphertext as follows:

$$c_0 - \underbrace{\mathbf{c}_\ell^\top \mathbf{d}_{\mathsf{ID},\mathsf{t}}}_{\text{level-}(\ell,\ell)\text{ component}} - \sum_{i=2}^{\ell} \underbrace{\mathbf{c}_i^\top \mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell}}_{\text{level-}(\ell,i)\text{ component}} \approx \mathsf{M} \left\lfloor \frac{q}{2} \right\rfloor ,$$

where each level of the ciphertext and decryption keys are in one-to-one correspondence with each other. Note that the level-ℓ user ID uses only level-(ℓ, i) decryption keys $\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell}$ for $i \in [\ell-1]$ provided in the key update $\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}$ to decrypt his own ciphertext. He simply forwards the remaining level-(k, i) decryption keys $\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},k}$ for $(k, i) \in [\ell+1, L] \times [\ell-1]$ as part of his key update $\mathsf{ku}_{\mathsf{ID},\mathsf{t}}$.

One can see that the problem in the previous scheme of Figure 4 is now resolved, since the public term $\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell}$ can only be used in combination with the level-(ℓ, i) ciphertext. In other words, due to the two-dimensional level broadcast $\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell}$ is only useful for decrypting ciphertexts of level-ℓ users. Furthermore, since the level-(ℓ, ℓ) decryption key $\mathbf{d}_{\mathsf{ID},\mathsf{t}}$ still remains secret, the publicly broadcast decryption keys $\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell}$ for $i \in [\ell-1]$ alone are insufficient for decrypting the ciphertexts sent to user ID. The remaining problem with this approach is that there is currently no way for the level-$(\ell-1)$ ancestors $\mathsf{pa}(\mathsf{ID})$ to create the level-$(k, \ell-1)$ decryption keys $(\mathbf{f}_{\mathsf{ID}_{[\ell-1]},\mathsf{t},k})_{k\in[\ell,L]}$ that are broadcast as part of the key updates $\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}$. Specifically, since they do not have the trapdoor $\mathbf{T}_{[\mathbf{A}_{\ell-1}|\mathbf{E}(\mathsf{ID}_{[\ell-1]})]}$, they cannot simply sample the level-$(k, \ell-1)$ decryption keys $(\mathbf{f}_{\mathsf{ID}_{[\ell-1]},\mathsf{t},k})_{k\in[\ell,L]}$ for every time period.

Introducing Level Conversion Keys: Finally, we arrive at our proposed RHIBE scheme (without DKER) illustrated in Figure 6. We overcome our final obstacle by introducing a tool called *level conversion keys*. In the scheme of Figure 5, a level-ℓ parent user ID is able to create his level-(ℓ, ℓ) decryption key $\mathbf{d}_{\mathsf{ID},\mathsf{t}}$ by himself although he cannot compute the level-(k, ℓ) decryption keys $(\mathbf{f}_{\mathsf{ID},\mathsf{t},k})_{k\in[\ell+1,L]}$ in the key updates $\mathsf{ku}_{\mathsf{ID},\mathsf{t}}$ (which is analogous to $(\mathbf{f}_{\mathsf{ID}_{[\ell-1]},\mathsf{t},k})_{k\in[\ell,L]}$ in $\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}$ of level-$(\ell-1)$ users in the figure). To overcome the issue, we define a level-$[\ell, k]$ conversion key $(\mathbf{f}_{\mathsf{ID},k})_{k\in[\ell+1,L]}$ of a level-ℓ user ID satisfying

$$\begin{array}{ll}
\mathsf{PP} := ((\mathbf{A}_i)_{i\in[L]}, (\mathbf{u}_k)_{k\in[L]}, \text{hash functions } \mathbf{E}(\cdot), \mathbf{F}(\cdot)), & \mathsf{sk}_{\mathsf{kgc}} := (\mathbf{T}_{\mathbf{A}_i})_{i\in[L]} \\
\mathsf{ct} := \Big(\ c_0 := \mathbf{u}_\ell^\top(\mathbf{s}_1 + \cdots \mathbf{s}_\ell) + \mathsf{noise} + \mathsf{M}\left\lfloor \frac{q}{2} \right\rfloor,\ (\mathbf{c}_i := [\mathbf{A}_i|\mathbf{E}(\mathsf{ID}_{[i]})|\mathbf{F}(\mathsf{t})]^\top \mathbf{s}_i + \mathsf{noise})_{i\in[\ell]}\ \Big) & \\
\mathsf{sk}_{\mathsf{ID}} := ((\mathbf{e}_{\mathsf{ID},\theta}), \boxed{(\mathbf{f}_{\mathsf{ID},k})_{k\in[\ell+1,L]}}, (\mathbf{T}_{[\mathbf{A}_i|\mathbf{E}(\mathsf{ID})]})_{i\in[\ell+1,L]}) & \text{s.t. } [\mathbf{A}_\ell|\mathbf{E}(\mathsf{ID})]\mathbf{e}_{\mathsf{ID},\theta} = \mathbf{u}_\theta, \\
 & \boxed{[\mathbf{A}_\ell|\mathbf{E}(\mathsf{ID})]\mathbf{f}_{\mathsf{ID},k} = \mathbf{u}_k - \mathbf{u}_\ell} \\
\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}} := ((\mathbf{e}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t},\theta}), (\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},k})_{i\in[\ell-1],k\in[\ell,L]}) & \\
\quad \text{s.t. } [\mathbf{A}_\ell|\mathbf{E}(\mathsf{pa}(\mathsf{ID}))|\mathbf{F}(\mathsf{t})]\mathbf{e}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t},\theta} = \mathbf{u}_\ell - \mathbf{u}_\theta, & [\mathbf{A}_i|\mathbf{E}(\mathsf{ID}_{[i]})|\mathbf{F}(\mathsf{t})]\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},k} = \mathbf{u}_k \\
\mathsf{dk}_{\mathsf{ID},\mathsf{t}} := (\mathbf{d}_{\mathsf{ID},\mathsf{t}}, (\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell})_{i\in[\ell-1]}) & \text{s.t. } [\mathbf{A}_\ell|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]\mathbf{d}_{\mathsf{ID},\mathsf{t}} = \mathbf{u}_\ell
\end{array}$$

Figure 6: **Level Conversion Key**

$$[\mathbf{A}_\ell|\mathbf{E}(\mathsf{ID})]\mathbf{f}_{\mathsf{ID},k} = \mathbf{u}_k - \mathbf{u}_\ell.$$

To compute level-(k, ℓ) decryption keys $(\mathbf{f}_{\mathsf{ID},\mathsf{t},k})_{k\in[\ell+1,L]}$ in key updates $\mathsf{ku}_{\mathsf{ID},\mathsf{t}}$, the level-$[\ell, k]$ conversion key allows the user ID to convert his *secret* level-(ℓ, ℓ) decryption key $\mathbf{d}_{\mathsf{ID},\mathsf{t}}$ which satisfies

$$[\mathbf{A}_\ell|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]\mathbf{d}_{\mathsf{ID},\mathsf{t}} = \mathbf{u}_\ell$$

into a *public* level-(k, ℓ) decryption key $\mathbf{f}_{\mathsf{ID},\mathsf{t},k}$ which satisfies

$$[\mathbf{A}_\ell|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]\mathbf{f}_{\mathsf{ID},\mathsf{t},k} = \mathbf{u}_k,$$

where the conversion is a simple component-wise addition. Since the scheme supports both the key delegation functionality and the key revocation mechanism, it can be shown to be a secure RHIBE scheme *without* DKER.

Adding DKER to the Construction: To make the above lattice-based RHIBE scheme in Figure 6 satisfy DKER, we will use the same idea incorporated in our generic construction of RIBE with DKER. Specifically, we add one more level to the above scheme and wrap a standard HIBE scheme around it to manage the partial key re-randomization property. The concrete construction appears in Section 6.

3 Preliminaries

In this section, we briefly summarize basic tools for lattice-based cryptography. We treat vectors in their column form. For a vector $\mathbf{v} \in \mathbb{R}^n$, denote $\|\mathbf{v}\|$ as the standard Euclidean norm. For a matrix $\mathbf{R} \in \mathbb{R}^{n \times n}$, denote $\|\mathbf{R}\|_{\text{GS}}$ as the longest column of the Gram-Schmidt orthogonalization of $\mathbf{R}$ and denote $\|\mathbf{R}\|_2$ as the largest singular value. We denote $\mathbf{I}_m$ as the $m \times m$ identity matrix and $\mathbf{0}_{n\times m}$ as the $n \times m$ zero matrix. We sometimes simply write $\mathbf{0}_n$ to denote (column) zero vectors.

Lattices. A (full-rank-integer) m-dimensional lattice Λ in $\mathbb{Z}^m$ is a set of the form $\{\sum_{i\in[m]} x_i \mathbf{b}_i | x_i \in \mathbb{Z}\}$, where $\mathbf{B} = \{\mathbf{b}_1, \cdots, \mathbf{b}_m\}$ are m linearly independent vectors in $\mathbb{Z}^m$. We call $\mathbf{B}$ the basis of the lattice Λ. For any positive integers n, m and $q \geq 2$, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a vector $\mathbf{u} \in \mathbb{Z}_q^n$, we define $\Lambda_q^{\perp}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m | \mathbf{A}\mathbf{z} = \mathbf{0}_n \mod q\}$ and $\Lambda_q^{\mathbf{u}}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m | \mathbf{A}\mathbf{z} = \mathbf{u} \mod q\}$.

Gaussian Measures. Let $\mathcal{D}_{\Lambda,\sigma}$ denote the standard discrete Gaussian distribution over Λ with a Gaussian parameter σ. We summarize some basic properties of discrete Gaussian distributions.

Lemma 1 ([GPV08]). *Let Λ be an m-dimensional lattice. Let $\mathbf{T}$ be a basis for Λ, and suppose $\sigma \geq \|\mathbf{T}\|_{\text{GS}} \cdot \omega(\sqrt{\log m})$. Then* $\Pr[\|\mathbf{x}\|_2 > \sigma\sqrt{m} : \mathbf{x} \leftarrow \mathcal{D}_{\Lambda,\sigma}] \leq \mathsf{negl}(m)$.

Lemma 2 ([GPV08]). *Let n, m, q be positive integers such that $m \geq 2n \log q$ and q a prime. Let σ be any positive real such that $\sigma \geq \omega(\sqrt{\log n})$. Then for $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ and $\mathbf{e} \leftarrow D_{\mathbb{Z}^m,\sigma}$, the distribution of $\mathbf{u} = \mathbf{A}\mathbf{e} \mod q$ is statistically close to uniform over $\mathbb{Z}_q^n$. Furthermore, for a fixed $\mathbf{u} \in \mathbb{Z}_q^n$, the conditional distribution of $\mathbf{e} \leftarrow D_{\mathbb{Z}^m,\sigma}$, given $\mathbf{A}\mathbf{e} = \mathbf{u} \mod q$ for a uniformly random $\mathbf{A}$ in $\mathbb{Z}_q^{n\times m}$ is $D_{\Lambda_q^{\mathbf{u}}(\mathbf{A}),\sigma}$ with all but negligible probability.*

Sampling Algorithms. We review some of the algorithms for sampling short vectors from a given lattice.

Lemma 3. *Let $n, m, \bar{m}, q > 0$ be positive integers with $m \geq 2n\lceil \log q \rceil$ and q a prime. Then, we have the following polynomial time algorithms:*

$\mathsf{TrapGen}(1^n, 1^m, q) \rightarrow (\mathbf{A}, \mathbf{T}_\mathbf{A})$*([Ajt99, AP11, MP12]): a randomized algorithm that outputs a full rank matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a basis $\mathbf{T}_\mathbf{A} \in \mathbb{Z}^{m\times m}$ for $\Lambda_q^{\perp}(\mathbf{A})$ such that $\mathbf{A}$ is statistically close to uniform and $\|\mathbf{T}_\mathbf{A}\|_{\text{GS}} = O\left(\sqrt{n \log q}\right)$ with overwhelming probability in n.*

$\mathsf{SampleLeft}(\mathbf{A}, \mathbf{F}, \mathbf{u}, \mathbf{T}_\mathbf{A}, \sigma) \rightarrow \mathbf{e}$*([ABB10, MP12]): a randomized algorithm that, given as input a full rank matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, a matrix $\mathbf{F} \in \mathbb{Z}_q^{n\times \bar{m}}$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$, a basis $\mathbf{T}_\mathbf{A} \in \mathbb{Z}^{m\times m}$ of $\Lambda_q^{\perp}(\mathbf{A})$, and a Gaussian parameter $\sigma \geq \|\mathbf{T}_\mathbf{A}\|_{\text{GS}} \cdot \omega\left(\sqrt{\log m}\right)$, outputs a vector $\mathbf{e} \in \mathbb{Z}^{m+\bar{m}}$ sampled from a distribution statistically close to $\mathcal{D}_{\Lambda_q^{\mathbf{u}}([\mathbf{A}|\mathbf{F}]),\sigma}$.*

([MP12]): There exists a fixed full rank matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ such that the lattice $\Lambda_q^{\perp}(\mathbf{G})$ has a publicly known basis $\mathbf{T}_\mathbf{G} \in \mathbb{Z}^{m\times m}$ with $\|\mathbf{T}_\mathbf{G}\|_{\text{GS}} \leq \sqrt{5}$.

For simplicity, we omit the $\mathsf{SamplePre}$ algorithm of [ABB10], since in our paper it will be used as a public algorithm to sample from the lattice $\mathbb{Z}^m$. The following algorithms allow one to securely delegate a trapdoor of a lattice to an arbitrary higher-dimensional extension, with a slight loss in quality. It can be obtained by combining the works of [CHKP12] and [ABB10] in a straightforward manner.

Lemma 4. *Let $n, m, \bar{m}, q > 0$ be positive integers with $m > n$ and q a prime. Then, we have the following polynomial time algorithms:*

$\mathsf{ExtRndLeft}(\mathbf{A}, \mathbf{F}, \mathbf{T_A}, \sigma) \to \mathbf{T}_{[\mathbf{A}|\mathbf{F}]}$: *a randomized algorithm that, given as input matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{F} \in \mathbb{Z}_q^{n\times \bar{m}}$, a basis $\mathbf{T_A}$ of $\Lambda_q^{\perp}(\mathbf{A})$, and a Gaussian parameter $\sigma \geq \|\mathbf{T_A}\|_{\text{GS}} \cdot \omega(\sqrt{\log n})$, outputs a matrix $\mathbf{T}_{[\mathbf{A}|\mathbf{F}]} \in \mathbb{Z}^{(m+\bar{m})\times(m+\bar{m})}$ distributed statistically close to $(\mathcal{D}_{\Lambda_q^{\perp}([\mathbf{A}|\mathbf{F}]),\sigma})^{m+\bar{m}}$.*

$\mathsf{ExtRndRight}(\mathbf{A}, \mathbf{G}, \mathbf{R}, \mathbf{T_G}, \sigma) \to \mathbf{T}_{[\mathbf{A}|\mathbf{AR}+\mathbf{G}]}$: *a randomized algorithm that, given as input full rank matrices $\mathbf{A}, \mathbf{G} \in \mathbb{Z}_q^{n\times m}$, a matrix $\mathbf{R} \in \mathbb{Z}^{m\times m}$, a basis $\mathbf{T_G}$ of $\Lambda_q^{\perp}(\mathbf{G})$, and a Gaussian parameter $\sigma \geq \|\mathbf{R}\|_2 \cdot \|\mathbf{T_G}\|_2 \cdot \omega(\sqrt{\log n})$ outputs a matrix $\mathbf{T}_{[\mathbf{A}|\mathbf{AR}+\mathbf{G}]} \in \mathbb{Z}^{2m\times 2m}$ distributed statistically close to $(\mathcal{D}_{\Lambda_q^{\perp}([\mathbf{A}|\mathbf{AR}+\mathbf{G}]),\sigma})^{2m}$.*

Useful Facts. We recall some useful facts that will be used in our paper.

Lemma 5 (Leftover Hash Lemma). *Let $q > 2$ be a prime, m, n, k be positive integers such that $m > (n + 1)\log q + \omega(\log n)$, k is polynomial in n, and let $\mathbf{R} \leftarrow \{-1, 1\}^{m\times k}$. Let $\mathbf{A}$ and $\mathbf{B}$ be matrices chosen uniformly in $\mathbb{Z}_q^{n\times m}$ and $\mathbb{Z}_q^{n\times k}$, respectively. Then the distribution of $(\mathbf{A}, \mathbf{AR})$ is negligibly close in n to the distribution of $(\mathbf{A}, \mathbf{B})$.*

Lemma 6. *Let m, k be positive integers such that $k \geq m$. If $\mathbf{R}$ is sampled uniformly in $\{-1, 1\}^{m\times k}$ then there exists a universal constant C such that $\Pr\left[\|\mathbf{R}\|_2 > C\sqrt{m + k}\right] < e^{-m}$.*

Lemma 7 (Noise Re-randomization, [KY16], Lemma 1). *Let q, ℓ, m be positive integers and r a positive real satisfying $r > \max\{\omega(\sqrt{\log m}), \omega(\sqrt{\log \ell})\}$. Let $\mathbf{b} \in \mathbb{Z}_q^m$ be arbitrary and $\mathbf{z}$ chosen from $D_{\mathbb{Z}^m,r}$. Then there exists a PPT algorithm ReRand such that for any $\mathbf{V} \in \mathbb{Z}^{m\times \ell}$ and positive real $\sigma > \|\mathbf{V}\|_2$, $\mathsf{ReRand}(\mathbf{V}, \mathbf{b} + \mathbf{z}, r, \sigma)$ outputs $\mathbf{b}'^{\top} = \mathbf{b}^{\top}\mathbf{V} + \mathbf{z}'^{\top} \in \mathbb{Z}_q^{\ell}$ where $\mathbf{z}'$ is distributed statistically close to $D_{\mathbb{Z}^{\ell},2r\sigma}$.*

We use the standard map to encode identities as matrices in $\mathbb{Z}_q^{n\times n}$.

Definition 1 ([ABB10]). *Let n, q be positive integers with q a prime. We say that a function $H : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\times n}$ is a full-rank difference (FRD) map if: for all distinct $\mathsf{ID}, \mathsf{ID}' \in \mathbb{Z}_q^n$, the matrix $H(\mathsf{ID}) - H(\mathsf{ID}') \in \mathbb{Z}_q^{n\times n}$ is full rank, and H is computable in polynomial time in $n \log q$.*

Hardness Assumption. The security of our RIBE scheme is reduced to the learning with errors (LWE) assumption introduced by Regev [Reg05].

Assumption 1 (Learning with Errors). *For integers n, m, a prime q, a real $\alpha \in (0, 1)$ such that $\alpha q > 2\sqrt{n}$, and a PPT algorithm $\mathcal{A}$, the advantage for the learning with errors problem $\mathsf{LWE}_{n,m,q,\mathcal{D}_{\mathbb{Z}^m,\alpha q}}$ of $\mathcal{A}$ is defined as $\left|\Pr\left[\mathcal{A}(\mathbf{A}, \mathbf{A}^{\top}\mathbf{s} + \mathbf{x}) = 1\right] - \Pr\left[\mathcal{A}(\mathbf{A}, \mathbf{v} + \mathbf{x}) = 1\right]\right|$, where $\mathbf{A} \leftarrow \mathbb{Z}^{n\times m}$, $\mathbf{s} \leftarrow \mathbb{Z}^n$, $\mathbf{x} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\alpha q}$, $\mathbf{v} \leftarrow \mathbb{Z}^m$. We say that the LWE assumption holds if the above advantage is negligible for all PPT $\mathcal{A}$.*

4 Formal Definitions for Revocable Hierarchical Identity-Based Encryption and a Supporting Lemma

In this section, we give formal definitions for RHIBE in Section 4.1. Then, in Section 4.2, we explain a simple and yet handy lemma that we call the "strategy-dividing lemma", which helps us simplify security proofs of R(H)IBE schemes in general. Since RIBE is a special case of RHIBE, we provide its formal definitions in Section A.1.

4.1 Revocable Hierarchical Identity-Based Encryption

As mentioned in the introduction, we re-formalize the syntax of RHIBE. Compared to the existing works on RHIBE, our syntax of RHIBE treats each user's secret key, state information, and revocation list in a simplified manner. Thus, we first explain our treatments of them, and then proceed to introducing the formal syntax and security definitions.

On the Role of a Secret Key. In the literature of R(H)IBE, typically, the entity who has the power to derive a secret key for lower-level users (i.e., the KGC in RIBE, and non-leaf users in RHIBE), is modeled as a stateful entity, and is supposed to maintain a so-called "state", in addition to its own secret key. The state information typically contains the information with which the revocation mechanism is realized, and needs to be treated confidentially. Since it is after all another type of secret information, in our syntax, we merge the roles of the state information and a secret key. Hence, in our model, each user is supposed to maintain its own secret key that is generated by its parent, and it could be updated after performing the key generation algorithm (for generating a secret key for its child) and the key update information generation algorithm.

On the Treatment of Revocation Lists. Note that unlike in standard revocable (non-hierarchical) IBE, the key update information and revocation lists of users are maintained individually by their corresponding parent users in RHIBE. In our syntax of R(H)IBE, we treat a revocation list just as a subset of (the corresponding children's) identity space. More specifically, the revocation list of a user with identity $\mathsf{ID} \in (\mathcal{ID})^{\ell}$ contains identities that belong to the set $\mathsf{ID} \| \mathcal{ID} \subseteq (\mathcal{ID})^{\ell+1}$.

In the literature, for R(H)IBE, it is typical to consider the "revoke" algorithm whose role is to add an identity of a user to be revoked into the revocation list. We do not explicitly introduce such an algorithm as part of our syntax, since it is a simple operation of appending revoked users to a list.

Syntax. An RHIBE scheme Π consists of the six algorithms ($\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{GenSK}, \mathsf{KeyUp}, \mathsf{GenDK}, \mathsf{Decrypt}$) with the following interface:

$\mathsf{Setup}(1^{\lambda}, L) \to (\mathsf{PP}, \mathsf{sk}_{\mathsf{kgc}})$: This is the *setup* algorithm that takes the security parameter 1^{λ} and the maximum depth of the hierarchy $L \in \mathbb{N}$ as input, and outputs a public parameter PP and the KGC's secret key $\mathsf{sk}_{\mathsf{kgc}}$ (also called a master secret key).
We assume that the plaintext space $\mathcal{M}$, the time period space $\mathcal{T} := \{1, 2, \dots, \mathsf{t}_{\max}\}$, where $\mathsf{t}_{\max}$ is polynomial in λ, the element identity space $\mathcal{ID}$, and the hierarchical identity space $\mathcal{ID}_{\mathsf{h}} := (\mathcal{ID})^{\leq L}$ are determined only by the security parameter λ, and their descriptions are contained in PP.

$\mathsf{Encrypt}(\mathsf{PP}, \mathsf{ID}, \mathsf{t}, \mathsf{M}) \to \mathsf{ct}$: This is the *encryption* algorithm that takes a public parameter PP, an identity ID, a time period t, and a plaintext M as input, and outputs a ciphertext ct.

$\mathsf{GenSK}(\mathsf{PP}, \mathsf{sk}_{\mathsf{pa(ID)}}, \mathsf{ID}) \to (\mathsf{sk}_{\mathsf{ID}}, \mathsf{sk}'_{\mathsf{pa(ID)}})$: This is the *secret key generation* algorithm that takes a public parameter PP, a parent's secret key $\mathsf{sk}_{\mathsf{pa(ID)}}$, and an identity $\mathsf{ID} \in \mathcal{ID}_{\mathsf{h}}$ as input, and may update the parent's secret key $\mathsf{sk}_{\mathsf{pa(ID)}}$. Then, it outputs a secret key $\mathsf{sk}_{\mathsf{ID}}$ for the identity ID and also the parent's "updated" secret key $\mathsf{sk}'_{\mathsf{pa(ID)}}$.

$\mathsf{KeyUp}(\mathsf{PP}, \mathsf{t}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{RL}_{\mathsf{ID},\mathsf{t}}, \mathsf{ku}_{\mathsf{pa(ID)},\mathsf{t}}) \to (\mathsf{ku}_{\mathsf{ID},\mathsf{t}}, \mathsf{sk}'_{\mathsf{ID}})$: This is the *key update information generation* algorithm that takes a public parameter PP, a time period t, a secret key $\mathsf{sk}_{\mathsf{ID}}$ (of a user with $\mathsf{ID} \in (\mathcal{ID})^{\leq L-1} \cup \{\mathsf{kgc}\}$), a revocation list $\mathsf{RL}_{\mathsf{ID},\mathsf{t}} \subseteq \mathsf{ID} \| \mathcal{ID}$, and a parent's key update $\mathsf{ku}_{\mathsf{pa(ID)},\mathsf{t}}$ as input, and may update the secret key $\mathsf{sk}_{\mathsf{ID}}$. Then, it outputs a key update $\mathsf{ku}_{\mathsf{ID},\mathsf{t}}$ and also the "updated" secret key $\mathsf{sk}'_{\mathsf{ID}}$.

In the special case $\mathsf{ID} = \mathsf{kgc}$, we define $\mathsf{ku}_{\mathsf{pa(kgc),t}} := \bot$ for all $\mathsf{t} \in \mathcal{T}$, i.e., a key update is not needed for generating the KGC's key update $\mathsf{ku}_{\mathsf{kgc,t}}$.

$\mathsf{GenDK}(\mathsf{PP}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ku}_{\mathsf{pa(ID),t}}) \to \mathsf{dk}_{\mathsf{ID,t}}$ or $\bot$: This is the *decryption key generation* algorithm that takes a public parameter PP, a secret key $\mathsf{sk}_{\mathsf{ID}}$ (of a user with $\mathsf{ID} \in (\mathcal{ID})^{\leq L}$), and a parent's key update $\mathsf{ku}_{\mathsf{pa(ID),t}}$ as input, and outputs a decryption key $\mathsf{dk}_{\mathsf{ID,t}}$ for time period t or the special "invalid" symbol $\bot$ indicating that ID or some of its ancestor has been revoked.

$\mathsf{Decrypt}(\mathsf{PP}, \mathsf{dk}_{\mathsf{ID,t}}, \mathsf{ct}) \to \mathsf{M}$: This is the *decryption* algorithm that takes a public parameter PP, a decryption key $\mathsf{dk}_{\mathsf{ID,t}}$, and a ciphertext ct as input, and outputs the decryption result M.

Correctness. We require the following to hold for an RHIBE scheme. Informally, we require a ciphertext corresponding to a user ID for time t to be properly decrypted by user ID if the user or any of its ancestor is not revoked on time t. To fully capture this, we consider all the possible scenarios of creating the secret key for user ID. Namely, for all $\lambda \in \mathbb{N}$, $L \in \mathbb{N}$, $(\mathsf{PP}, \mathsf{sk}_{\mathsf{kgc}}) \leftarrow \mathsf{Setup}(1^\lambda, L)$, $\ell \in [L]$, $\mathsf{ID} \in (\mathcal{ID})^\ell$, $\mathsf{t} \in \mathcal{T}$, $\mathsf{M} \in \mathcal{M}$, $\mathsf{RL}_{\mathsf{kgc,t}} \subseteq \mathcal{ID}$, $\mathsf{RL}_{\mathsf{ID}_{[1]},\mathsf{t}} \subseteq \mathsf{ID}_{[1]} \| \mathcal{ID}, \dots, \mathsf{RL}_{\mathsf{ID}_{[\ell-1]},\mathsf{t}} \subseteq \mathsf{ID}_{[\ell-1]} \| \mathcal{ID}$, if $\mathsf{ID}' \notin \mathsf{RL}_{\mathsf{pa(ID'),t}}$ holds for all $\mathsf{ID}' \in \mathsf{prefix}(\mathsf{ID})$, then we require $\mathsf{M}' = \mathsf{M}$ to hold after executing the following procedures:

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

We note that, the most stringent way to define correctness would be to also capture the fact that the secret keys $\mathsf{sk}_{\mathsf{ID}}$ can be further updated after executing GenSK. In particular, the output of KeyUp, which takes as input the secret key $\mathsf{sk}_{\mathsf{ID}}$, may differ in general before and after GenSK is run. Therefore, to be more precise, we should also allow an arbitrary (polynomial) number of executions of GenSK in between steps (2.1) and (2.2). However, we defined correctness as above for the sake of simplicity and readability. We note that our scheme satisfies the more stringent correctness (which will be obvious from the construction).

Security Definition. Here, we give a formal security definition for RHIBE.

It seems to us that since the previous security definitions [BGK08, SE13b, SE14a, SE16] have some ambiguous treatment in the security game, it was up to the readers to interpret the definitions and the proofs. Therefore, in our work, we provide a refined security definition for RHIBE which in particular is a more rigorous and explicit treatment than the previous definitions. (We also provide refined security definitions for RIBE in Section A.1.)

Specifically, we explicitly separate the secret key generation and secret key reveal queries, so that we can capture a situation where some $\mathsf{sk}_{\mathsf{ID}}$ has been generated but not revealed to an adversary. Furthermore, we combine the "revoke" and "key update" queries in the previous definitions into the single "revoke & key update" query, and introduce the notion of the "current time period" $\mathsf{t}_{\mathsf{cu}} \in \mathcal{T}$ which is coordinated with the adversary's revoke & key update query. These make it possible to capture a situation where all key updates of non-revoked users are well-defined throughout the security game.

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

[5] Here, $\mathsf{sk}_{\mathsf{ID}}$ is the latest secret key that is the result of the step (2).

Formally, let $\Pi = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{GenSK}, \mathsf{KeyUp}, \mathsf{GenDK}, \mathsf{Decrypt})$ be an RHIBE scheme. We will only consider selective-identity security, which is defined via a game between an adversary $\mathcal{A}$ and the challenger $\mathcal{C}$. The game is parameterized by the security parameter λ and a polynomial $L = L(\lambda)$ representing the maximum depth of the identity hierarchy. Moreover, the game has the global counter t_{cu}, initialized with 1, that denotes the "current time period" with which $\mathcal{C}$'s responses to $\mathcal{A}$'s queries are controlled. The game proceeds as follows:

At the beginning, $\mathcal{A}$ sends the challenge identity/time period pair $(\mathsf{ID}^*, \mathsf{t}^*) \in (\mathcal{ID})^{\leq L} \times \mathcal{T}$ to $\mathcal{C}$. Next, $\mathcal{C}$ runs $(\mathsf{PP}, \mathsf{sk}_{\mathsf{kgc}}) \leftarrow \mathsf{Setup}(1^\lambda, L)$, and prepares a list $\texttt{SKList}$ that initially contains $(\mathsf{kgc}, \mathsf{sk}_{\mathsf{kgc}})$, and into which identity/secret key pairs $(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}})$ generated during the game will be stored. From this point on, whenever a new secret key is generated or an existing secret key is updated for an identity $\mathsf{ID} \in (\mathcal{ID})^{\leq L} \cup \{\mathsf{kgc}\}$ due to the execution of GenSK or KeyUp, $\mathcal{C}$ will store $(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}})$ or update the corresponding entry $(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}})$ in $\texttt{SKList}$, and we will not explicitly mention this addition/update. Then, $\mathcal{C}$ executes $(\mathsf{ku}_{\mathsf{kgc},1}, \mathsf{sk}'_{\mathsf{kgc}}) \leftarrow \mathsf{KeyUp}(\mathsf{PP}, \mathsf{t}_{\mathsf{cu}} = 1, \mathsf{sk}_{\mathsf{kgc}}, \mathsf{RL}_{\mathsf{kgc},1} = \emptyset, \perp)$ for generating a key update for the initial time period $\mathsf{t}_{\mathsf{cu}} = 1$. After that, $\mathcal{C}$ gives PP and $\mathsf{ku}_{\mathsf{kgc},1}$ to $\mathcal{A}$.

From this point on, $\mathcal{A}$ may adaptively make the following five types of queries to $\mathcal{C}$:

Secret Key Generation Query: Upon a query $\mathsf{ID} \in (\mathcal{ID})^{\leq L}$ from $\mathcal{A}$, $\mathcal{C}$ checks if $(\mathsf{ID}, *) \notin \texttt{SKList}$ and $(\mathsf{pa}(\mathsf{ID}), \mathsf{sk}_{\mathsf{pa}(\mathsf{ID})}) \in \texttt{SKList}$ for some $\mathsf{sk}_{\mathsf{pa}(\mathsf{ID})}$, and returns $\perp$ to $\mathcal{A}$ if this is *not* the case. Otherwise, $\mathcal{C}$ executes $(\mathsf{sk}_{\mathsf{ID}}, \mathsf{sk}'_{\mathsf{pa}(\mathsf{ID})}) \leftarrow \mathsf{GenSK}(\mathsf{PP}, \mathsf{sk}_{\mathsf{pa}(\mathsf{ID})}, \mathsf{ID})$. If $\mathsf{ID} \in (\mathcal{ID})^{\leq L-1}$, then $\mathcal{C}$ furthermore executes $(\mathsf{ku}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}}, \mathsf{sk}'_{\mathsf{ID}}) \leftarrow \mathsf{KeyUp}(\mathsf{PP}, \mathsf{t}_{\mathsf{cu}}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{RL}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}} = \emptyset, \mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}_{\mathsf{cu}}})$. Then, $\mathcal{C}$ returns $\mathsf{ku}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}}$ to $\mathcal{A}$ if $\mathsf{ID} \in (\mathcal{ID})^{\leq L-1}$, or returns nothing to $\mathcal{A}$ if $\mathsf{ID} \in (\mathcal{ID})^{L}$.[6]

We require that all identities ID appearing in the following queries (except the challenge query) be "activated", in the sense that $\mathsf{sk}_{\mathsf{ID}}$ is generated via this query and hence $(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}}) \in \texttt{SKList}$.

Secret Key Reveal Query: Upon a query $\mathsf{ID} \in (\mathcal{ID})^{\leq L}$ from $\mathcal{A}$, $\mathcal{C}$ checks if the following condition is satisfied:

- If $\mathsf{t}_{\mathsf{cu}} \geq \mathsf{t}^*$ and $\mathsf{ID}' \notin \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}'),\mathsf{t}^*}$ for all $\mathsf{ID}' \in \mathsf{prefix}(\mathsf{ID}^*)$, then $\mathsf{ID} \notin \mathsf{prefix}(\mathsf{ID}^*)$.[7]

If this condition is *not* satisfied, then $\mathcal{C}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ finds $\mathsf{sk}_{\mathsf{ID}}$ from $\texttt{SKList}$, and returns it to $\mathcal{A}$.

Revoke & Key Update Query: Upon a query $\mathsf{RL} \subseteq (\mathcal{ID})^{\leq L}$ (which denotes the set of identities that are going to be revoked in the next time period) from $\mathcal{A}$, $\mathcal{C}$ checks if the following conditions are satisfied simultaneously:

- $\mathsf{RL}_{\mathsf{ID},\mathsf{t}_{\mathsf{cu}}} \subseteq \mathsf{RL}$ for all $\mathsf{ID} \in \mathcal{ID}^{\leq L-1} \cup \{\mathsf{kgc}\}$ that appear in $\texttt{SKList}$.[8]
- For all identities ID such that $(\mathsf{ID}, *) \in \texttt{SKList}$ and $\mathsf{ID}' \in \mathsf{prefix}(\mathsf{ID})$, if $\mathsf{ID}' \in \mathsf{RL}$ then $\mathsf{ID} \in \mathsf{RL}$.[9]
- If $\mathsf{t}_{\mathsf{cu}} = \mathsf{t}^* - 1$ and $\mathsf{sk}_{\mathsf{ID}'}$ for some $\mathsf{ID}' \in \mathsf{prefix}(\mathsf{ID}^*)$ has already been revealed by the

[6] We stress that just making this query does not give the secret key $\mathsf{sk}_{\mathsf{ID}}$ to $\mathcal{A}$. It is captured by the "Secret Key Reveal Query" explained next. Furthermore, we provide the key updates to $\mathcal{A}$ unconditionally, since they are typically broadcast via an insecure channel and are not meant to be secret.

[7] In other words, this check ensures that if ID^* or any of its ancestors was *not* revoked before the challenge time period t^*, then $\mathsf{sk}_{\mathsf{ID}}$ will not be revealed for any $\mathsf{ID} \in \mathsf{prefix}(\mathsf{ID}^*)$. Without this condition, there is a trivial attack on any RHIBE scheme. See Section 4.1 for detailed explanations.

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

[9] In other words, this check ensures that if some ID is revoked, then all of its descendants are also revoked.

secret key reveal query ID', then $\mathsf{ID}' \in \mathsf{RL}$. [10]

If these conditions are *not* satisfied, then $\mathcal{C}$ returns $\perp$ to $\mathcal{A}$.
Otherwise $\mathcal{C}$ increments the current time period by $\mathsf{t_{cu}} \leftarrow \mathsf{t_{cu}} + 1$. Then, $\mathcal{C}$ executes the following operations (1) and (2) for all "activated" and non-revoked identities ID, i.e., $\mathsf{ID} \in (\mathcal{ID})^{\le L-1} \cup \{\mathsf{kgc}\}$, $(\mathsf{ID}, *) \in \mathtt{SKList}$, and $\mathsf{ID} \notin \mathsf{RL}$, in the breadth-first order in the identity hierarchy:

(1) Set $\mathsf{RL}_{\mathsf{ID},\mathsf{t_{cu}}} \leftarrow \mathsf{RL} \cap (\mathsf{ID} \| \mathcal{ID})$, where we define $\mathsf{kgc} \| \mathcal{ID} := \mathcal{ID}$.
(2) Run $(\mathsf{ku}_{\mathsf{ID},\mathsf{t_{cu}}}, \mathsf{sk}'_{\mathsf{ID}}) \leftarrow \mathsf{KeyUp}(\mathsf{PP}, \mathsf{t_{cu}}, \mathsf{sk_{ID}}, \mathsf{RL}_{\mathsf{ID},\mathsf{t_{cu}}}, \mathsf{ku}_{\mathsf{pa(ID)},\mathsf{t_{cu}}})$, where $\mathsf{ku}_{\mathsf{pa(kgc)},\mathsf{t_{cu}}} := \perp$.

Finally, $\mathcal{C}$ returns all the generated key updates $\{\mathsf{ku}_{\mathsf{ID},\mathsf{t_{cu}}}\}_{(\mathsf{ID},*) \in \mathtt{SKList}}$ to $\mathcal{A}$.

Decryption Key Reveal Query: Upon a query $(\mathsf{ID}, \mathsf{t}) \in (\mathcal{ID})^{\le L} \times \mathcal{T}$ from $\mathcal{A}$, $\mathcal{C}$ checks if the following conditions are simultaneously satisfied:

– $\mathsf{t} \le \mathsf{t_{cu}}$.
– $\mathsf{ID} \notin \mathsf{RL}_{\mathsf{pa(ID)},\mathsf{t}}$
– $(\mathsf{ID}, \mathsf{t}) \ne (\mathsf{ID}^*, \mathsf{t}^*)$. [11]

If these conditions are *not* satisfied, then $\mathcal{C}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ finds $\mathsf{sk_{ID}}$ from $\mathtt{SKList}$, runs $\mathsf{dk}_{\mathsf{ID},\mathsf{t}} \leftarrow \mathsf{GenDK}(\mathsf{PP}, \mathsf{sk_{ID}}, \mathsf{ku}_{\mathsf{pa(ID)},\mathsf{t}})$, and returns $\mathsf{dk}_{\mathsf{ID},\mathsf{t}}$ to $\mathcal{A}$.[12]

Challenge Query: $\mathcal{A}$ is allowed to make this query only once. Upon a query $(\mathsf{M}_0, \mathsf{M}_1)$ from $\mathcal{A}$, where it is required that $|\mathsf{M}_0| = |\mathsf{M}_1|$, $\mathcal{C}$ picks the challenge bit $b \in \{0,1\}$ uniformly at random, runs $\mathsf{ct}^* \leftarrow \mathsf{Encrypt}(\mathsf{PP}, \mathsf{ID}^*, \mathsf{t}^*, \mathsf{M}_b)$, and returns the challenge ciphertext ct^* to $\mathcal{A}$.

At some point, $\mathcal{A}$ outputs $b' \in \{0,1\}$ as its guess for b and terminates.

The above completes the description of the game. In this game, $\mathcal{A}$'s selective-identity security advantage $\mathsf{Adv}^{\mathtt{RHIBE\text{-}sel}}_{\Pi,L,\mathcal{A}}(\lambda)$ is defined by $\mathsf{Adv}^{\mathtt{RHIBE\text{-}sel}}_{\Pi,L,\mathcal{A}}(\lambda) := 2 \cdot |\Pr[b' = b] - 1/2|$.

Definition 2. *We say that an RHIBE scheme* Π *with depth* L *satisfies* selective-identity security*, if the advantage* $\mathsf{Adv}^{\mathtt{RHIBE\text{-}sel}}_{\Pi,L,\mathcal{A}}(\lambda)$ *is negligible for all PPT adversaries* $\mathcal{A}$.

On the Necessity of the Condition in the Secret Reveal Query. The condition is necessary, because if we do not have this condition, there is a trivial attack on any RHIBE scheme. Specifically, suppose there is some $\mathsf{ID} \in \mathsf{prefix}(\mathsf{ID}^*)$ such that $\mathsf{ID} \notin \mathsf{RL}_{\mathsf{pa(ID)},\mathsf{t}^*}$, and $\mathcal{A}$ obtains $\mathsf{sk_{ID}}$ after the challenge time period t^* via a secret key reveal query. Then, $\mathcal{A}$ can compute $\mathsf{sk}_{\mathsf{ID}'}$ for all $\mathsf{ID}' \in \mathsf{prefix}(\mathsf{ID}^*) \backslash \mathsf{prefix}(\mathsf{ID})$ (including $\mathsf{sk}_{\mathsf{ID}^*}$). Furthermore, since $\mathcal{A}$ owns $\mathsf{ku}_{\mathsf{pa(ID)},\mathsf{t}^*}$ that is generated by using $\mathsf{RL}_{\mathsf{pa(ID)},\mathsf{t}^*}$ that does not contain ID, letting $\widetilde{\mathsf{ku}}_{\mathsf{pa(ID)},\mathsf{t}^*} := \mathsf{ku}_{\mathsf{pa(ID)},\mathsf{t}^*}$, $\mathcal{A}$ can compute its own key update $\widetilde{\mathsf{ku}}_{\mathsf{ID}',\mathsf{t}^*}$ by sequentially executing $\mathsf{KeyUp}(\mathsf{PP}, \mathsf{t}^*, \mathsf{sk}_{\mathsf{ID}'}, \widetilde{\mathsf{RL}}_{\mathsf{pa(ID')},\mathsf{t}^*}, \widetilde{\mathsf{ku}}_{\mathsf{pa(ID')},\mathsf{t}^*})$, for each $\mathsf{ID}' \in \mathsf{prefix}(\mathsf{ID}^*) \backslash \mathsf{prefix}(\mathsf{ID})$, so that $\mathsf{ID}' \notin \widetilde{\mathsf{RL}}_{\mathsf{pa(ID')},\mathsf{t}^*}$. Therefore, $\mathcal{A}$ can obtain both $\mathsf{sk}_{\mathsf{ID}^*}$ and $\widetilde{\mathsf{ku}}_{\mathsf{pa(ID^*)},\mathsf{t}^*}$ where $\widetilde{\mathsf{ku}}_{\mathsf{pa(ID^*)},\mathsf{t}^*}$ is computed so that ID^* is not revoked at the time period t^*, from which $\mathcal{A}$ can derive a decryption key (using GenDK) that can decrypt the challenge ciphertext ct^*.

[10] In other words, this check is to ensure that if the secret key $\mathsf{sk}_{\mathsf{ID}'}$ of some ancestor ID' of ID^* (or ID^* itself) has been revealed to $\mathcal{A}$, then ID' is revoked in the next time period.

[11] In previous works [SE13b, SE16], $\mathcal{A}$ is disallowed to obtain not only $\mathsf{dk}_{\mathsf{ID}^*,\mathsf{t}^*}$ (which is clearly necessary to avoid a trivial attack), but also decryption keys $\mathsf{dk}_{\mathsf{ID}',\mathsf{t}^*}$ for all $\mathsf{ID}' \in \mathsf{prefix}(\mathsf{ID}^*)$. Our relaxed condition here makes the defined security stronger since $\mathcal{A}$ is able to obtain additional information without any restrictions.

[12] Note that $\mathsf{ku}_{\mathsf{pa(ID)},\mathsf{t}}$ must have been already generated at this point due to the condition $\mathsf{t} \le \mathsf{t_{cu}}$.

4.2 Strategy-Dividing Lemma

In the literature of R(H)IBE, a typical security proof for an R(H)IBE scheme goes as follows:

(1) classify an adversary's strategies into multiple pre-determined types, say Type-1 to Type-n for some $n \in \mathbb{N}$ that cover all possible strategies, and
(2) for each $i \in [n]$, prove that any adversary that is promised to follow the Type-i strategy (and never break the promise) has negligible advantage in attacking the considered scheme.

Here, it is implicitly assumed that the above mentioned "type-classification-based" security proof is sufficient for proving security against arbitrary adversaries that may decide their attack strategies adaptively during the game.

For completeness, we formalize the above implicit argument as a simple yet handy "strategy-dividing lemma", which helps us simplify security proofs for R(H)IBE schemes in general. We only state it for selective-identity security of an RIBE scheme for concreteness, but it can be similarly shown for R(H)IBE with all security notions considered in the paper.

Lemma 8 (Strategy-Dividing Lemma). *Let Π be an RIBE scheme, and let $\mathcal{A}$ be any PPT adversary against the selective-identity security of Π. Assume that there are n possible attack strategies for $\mathcal{A}$, Type-1, ... Type-n, that satisfy the following conditions:*

(1) Type-1, ..., Type-n cover all possible strategies, and each Type-i is mutually exclusive.
(2) For every $i \in [n]$, whether $\mathcal{A}$ has deviated from the Type-i strategy is "publicly detectable", in the sense that during the security game, as soon as $\mathcal{A}$ deviates from the Type-i strategy, it can be efficiently recognized given $\mathcal{A}$'s view at the moment it deviates from the strategy.

Then, there exist PPT adversaries $\mathcal{A}_1, \dots, \mathcal{A}_n$ against the selective-identity security of Π, such that $\mathcal{A}_i$ always follows the Type-i strategy for every $i \in [n]$, and

$$\mathsf{Adv}^{\texttt{RIBE-sel}}_{\Pi,\mathcal{A}}(\lambda) \le \sum_{i \in [n]} \mathsf{Adv}^{\texttt{RIBE-sel}}_{\Pi,\mathcal{A}_i}(\lambda). \tag{1}$$

In particular, if $\mathsf{Adv}^{\texttt{RIBE-sel}}_{\Pi,\mathcal{A}'_i}(\lambda)$ is negligible for all PPT adversaries $\mathcal{A}'_i$ that always follow the Type-i strategy and for all $i \in [n]$, then Π satisfies selective-identity security for any PPT adversary $\mathcal{A}$ following an arbitrary strategy.

Proof of Lemma 8. Let $\mathcal{A}$ be any PPT adversary that attacks the selective-identity security of an RIBE scheme Π, and suppose there are n attack strategies, Type-1, ..., Type-n, satisfying the conditions (1) and (2) stated in the lemma. We emphasize that $\mathcal{A}$ may decide its strategy adaptively during the security game.

In the selective-identity security game, let S be the event that $\mathcal{A}$ succeeds in guessing the challenge bit (i.e. $b' = b$ occurs). Furthermore, for each $i \in [n]$, let T_i be the event that $\mathcal{A}$ follows the Type-i strategy in the game. Since each Type-i is mutually exclusive and covers all possibilities, we have $\Pr[\bigvee_{i \in [n]} \mathsf{T}_i] = \sum_{i \in [n]} \Pr[\mathsf{T}_i] = 1$.

Using the definitions of the events, we can calculate $\mathcal{A}$'s advantage as follows:

$$\begin{aligned}\mathsf{Adv}^{\texttt{RIBE-sel}}_{\Pi,\mathcal{A}}(\lambda) &= 2 \cdot \Big|\Pr[\mathsf{S}] - \frac{1}{2}\Big| = 2 \cdot \Big|\sum_{i \in [n]} \Pr[\mathsf{S} \wedge \mathsf{T}_i] - \frac{1}{2}\sum_{i \in [n]} \Pr[\mathsf{T}_i]\Big| \\ &= 2 \cdot \Big|\sum_{i \in [n]} \Big(\Pr[\mathsf{S} \wedge \mathsf{T}_i] + \frac{1}{2}\Pr[\overline{\mathsf{T}_i}] - \frac{1}{2}\Big)\Big|\end{aligned}$$

$$\leq 2 \cdot \sum_{i \in [n]} \left| \Pr[\mathsf{S} \wedge \mathsf{T}_i] + \frac{1}{2} \Pr[\overline{\mathsf{T}_i}] - \frac{1}{2} \right|. \tag{2}$$

Now, for each $i \in [n]$, consider an adversary $\mathcal{A}_i$ against the selective-identity security of Π, which internally simulates the selective-identity security game for $\mathcal{A}$ while playing its own selective-identity security game with the challenger $\mathcal{C}$. Whenever $\mathcal{A}$ tries to send some value to the challenger, $\mathcal{A}_i$ forwards it to $\mathcal{A}_i$'s challenger $\mathcal{C}$, and when $\mathcal{C}$ sends some value to an adversary, $\mathcal{A}_i$ forwards it to $\mathcal{A}$, except that as soon as $\mathcal{A}_i$ detects that $\mathcal{A}$ has deviated from the Type-i strategy, $\mathcal{A}_i$ outputs a uniformly random bit and terminates. (A_i can detect it due to the public detectability condition.) Note that by design, this $\mathcal{A}_i$ is PPT and never deviates from the Type-i strategy.

Now, let S' be the event that $\mathcal{A}_i$ succeeds in guessing the challenge bit, and let T'_i be the event that $\mathcal{A}$ follows the Type-i strategy in the security game simulated by $\mathcal{A}_i$. Note that by design, $\mathcal{A}_i$ perfectly simulates the selective-identity security game for $\mathcal{A}$ so that $\mathcal{A}$'s challenge bit is that of $\mathcal{A}_i$, *until the point $\mathcal{A}$ deviates from the Type-i strategy.* This implies that we have $\Pr[\mathsf{S}' \wedge \mathsf{T}'_i] = \Pr[\mathsf{S} \wedge \mathsf{T}_i]$ and $\Pr[\mathsf{T}'_i] = \Pr[\mathsf{T}_i]$. Furthermore, whenever $\mathcal{A}$ deviates from the Type-i strategy, $\mathcal{A}_i$ always detects it and outputs a random bit. This means that we have $\Pr[\mathsf{S}' | \overline{\mathsf{T}'_i}] = 1/2$.

Hence, we can calculate $\mathcal{A}_i$'s selective-identity advantage as follows:

$$\begin{aligned} \mathsf{Adv}^{\texttt{RIBE-sel}}_{\Pi, \mathcal{A}_i}(\lambda) &= 2 \cdot \left| \Pr[\mathsf{S}'] - \frac{1}{2} \right| \\ &= 2 \cdot \left| \Pr[\mathsf{S}' \wedge \mathsf{T}'_i] + \Pr[\mathsf{S}' | \overline{\mathsf{T}'_i}] \cdot \Pr[\overline{\mathsf{T}'_i}] - \frac{1}{2} \right| \\ &= 2 \cdot \left| \Pr[\mathsf{S} \wedge \mathsf{T}_i] + \frac{1}{2} \Pr[\overline{\mathsf{T}_i}] - \frac{1}{2} \right|. \end{aligned} \tag{3}$$

Using Eq. (3) in Eq. (2), we can conclude that there exist PPT adversaries $\mathcal{A}_1, \dots, \mathcal{A}_n$ such that $\mathcal{A}_i$ always follows the Type-i strategy for every $i \in [n]$ and Eq. (1) holds, as desired. This completes the proof of Lemma 8. □

5 Generic Construction of RIBE with DKER

In this section, we show a "security-enhancing" generic construction for RIBE. Namely, we show how to construct an RIBE scheme with DKER by combining an RIBE scheme without DKER and a 2-level (non-revocable) HIBE scheme (where the formal definitions for HIBE are provided in Section A.2).

Let $\mathsf{r}.\Pi = (\mathsf{r.Setup}, \mathsf{r.Encrypt}, \mathsf{r.GenSK}, \mathsf{r.KeyUp}, \mathsf{r.GenDK}, \mathsf{r.Decrypt})$ be an RIBE scheme (without DKER) with identity space $\mathsf{r}.\mathcal{ID}$, plaintext space $\mathsf{r}.\mathcal{M}$, and time period space $\mathsf{r}.\mathcal{T}$. Let $\mathsf{h}.\Pi = (\mathsf{h.Setup}, \mathsf{h.Encrypt}, \mathsf{h.GenSK}, \mathsf{h.Delegate}, \mathsf{h.Decrypt})$ be a 2-level HIBE scheme with element identity space $\mathsf{h}.\mathcal{ID}$ and plaintext space $\mathsf{h}.\mathcal{M}$. We assume $\mathsf{r}.\mathcal{ID} = \mathsf{h}.\mathcal{ID}$, $\mathsf{r}.\mathcal{M} = \mathsf{h}.\mathcal{M}$, and $\mathsf{r}.\mathcal{T} \subseteq \mathsf{h}.\mathcal{ID}$. Furthermore, we assume that the plaintext space is finite and forms an abelian group with the addition "+" as the group operation.

Using these ingredients, we construct an RIBE scheme $\Pi = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{GenSK}, \mathsf{KeyUp}, \mathsf{GenDK}, \mathsf{Decrypt})$ with DKER as follows. The identity space $\mathcal{ID}$, the plaintext space $\mathcal{M}$, and the time period space $\mathcal{T}$ of the constructed RIBE scheme Π are, respectively, $\mathcal{ID} = \mathsf{r}.\mathcal{ID} = \mathsf{h}.\mathcal{ID}$, $\mathcal{M} = \mathsf{r}.\mathcal{M} = \mathsf{h}.\mathcal{M}$, and $\mathcal{T} = \mathsf{r}.\mathcal{T} \subseteq \mathsf{h}.\mathcal{ID}$.

$\mathsf{Setup}(1^\lambda) \to (\mathsf{PP}, \mathsf{sk_{kgc}})$: It takes the security parameter 1^λ as input, and runs $(\mathsf{r.PP}, \mathsf{r.sk_{kgc}}) \leftarrow \mathsf{r.Setup}(1^\lambda)$ and $(\mathsf{h.PP}, \mathsf{h.sk_{kgc}}) \leftarrow \mathsf{h.Setup}(1^\lambda)$. Then, it outputs a public parameter $\mathsf{PP} := (\mathsf{r.PP}, \mathsf{h.PP})$ and the KGC's secret key $\mathsf{sk_{kgc}} := (\mathsf{r.sk_{kgc}}, \mathsf{h.sk_{kgc}})$.

$\mathsf{Encrypt}(\mathsf{PP}, \mathsf{ID}, \mathsf{t}, \mathsf{M}) \to \mathsf{ct}$: It takes a public parameter $\mathsf{PP} = (\mathsf{r.PP}, \mathsf{h.PP})$, an identity $\mathsf{ID} \in \mathcal{ID}$, a time period $\mathsf{t} \in \mathcal{T}$, and a plaintext $\mathsf{M} \in \mathcal{M}$ as input, and samples a pair $(\mathsf{r.M}, \mathsf{h.M}) \in \mathcal{M}^2$ uniformly at random, subject to $\mathsf{r.M} + \mathsf{h.M} = \mathsf{M}$. Then, it runs $\mathsf{r.ct} \leftarrow \mathsf{r.Encrypt}(\mathsf{r.PP}, \mathsf{ID}, \mathsf{t}, \mathsf{r.M})$ and $\mathsf{h.ct} \leftarrow \mathsf{h.Encrypt}(\mathsf{h.PP}, (\mathsf{ID}, \mathsf{t}), \mathsf{h.M})$. Finally, it outputs a ciphertext $\mathsf{ct} := (\mathsf{r.ct}, \mathsf{h.ct})$.

$\mathsf{GenSK}(\mathsf{PP}, \mathsf{sk_{kgc}}, \mathsf{ID}) \to (\mathsf{sk_{ID}}, \mathsf{sk'_{kgc}})$: It takes a public parameter $\mathsf{PP} = (\mathsf{r.PP}, \mathsf{h.PP})$, the KGC's secret key $\mathsf{sk_{kgc}} = (\mathsf{r.sk_{kgc}}, \mathsf{h.sk_{kgc}})$, and an identity $\mathsf{ID} \in \mathcal{ID}$ as input, and runs $(\mathsf{r.sk_{ID}}, \mathsf{r.sk'_{kgc}}) \leftarrow \mathsf{r.GenSK}(\mathsf{r.PP}, \mathsf{r.sk_{kgc}}, \mathsf{ID})$ and $\mathsf{h.sk_{ID}} \leftarrow \mathsf{h.GenSK}(\mathsf{h.PP}, \mathsf{h.sk_{kgc}}, \mathsf{ID})$. Then, it outputs a secret key $\mathsf{sk_{ID}} := (\mathsf{r.sk_{ID}}, \mathsf{h.sk_{ID}})$ for the identity ID and also the KGC's updated secret key $\mathsf{sk'_{kgc}} := (\mathsf{r.sk'_{kgc}}, \mathsf{h.sk_{kgc}})$.

$\mathsf{KeyUp}(\mathsf{PP}, \mathsf{t}, \mathsf{sk_{kgc}}, \mathsf{RL_t}) \to (\mathsf{ku_t}, \mathsf{sk'_{kgc}})$: It takes a public parameter $\mathsf{PP} = (\mathsf{r.PP}, \mathsf{h.PP})$, a time period $\mathsf{t} \in \mathcal{T}$, the KGC's secret key $\mathsf{sk_{kgc}} = (\mathsf{r.sk_{kgc}}, \mathsf{h.sk_{kgc}})$, and a revocation list $\mathsf{RL_t} \subseteq \mathcal{ID}$ as input, and, runs $(\mathsf{r.ku_t}, \mathsf{r.sk'_{kgc}}) \leftarrow \mathsf{r.KeyUp}(\mathsf{r.PP}, \mathsf{t}, \mathsf{r.sk_{kgc}}, \mathsf{RL_t})$. Then, it outputs a key update $\mathsf{ku_t} := \mathsf{r.ku_t}$ and also the KGC's updated secret key $\mathsf{sk'_{kgc}} := (\mathsf{r.sk'_{kgc}}, \mathsf{h.sk_{kgc}})$.

$\mathsf{GenDK}(\mathsf{PP}, \mathsf{sk_{ID}}, \mathsf{ku_t}) \to \mathsf{dk_{ID,t}}$ or $\perp$: It takes a public parameter $\mathsf{PP} = (\mathsf{r.PP}, \mathsf{h.PP})$, a secret key $\mathsf{sk_{ID}} = (\mathsf{r.sk_{ID}}, \mathsf{h.sk_{ID}})$, and a key update $\mathsf{ku_t} = \mathsf{r.ku_t}$ as input, and runs $\mathsf{r.dk_{ID,t}} \leftarrow \mathsf{r.GenDK}(\mathsf{r.PP}, \mathsf{r.sk_{ID}}, \mathsf{r.ku_t})$ and $\mathsf{h.sk_{ID,t}} \leftarrow \mathsf{h.Delegate}(\mathsf{h.PP}, \mathsf{h.sk_{ID}}, \mathsf{t})$. Then, it outputs a decryption key $\mathsf{dk_{ID,t}} := (\mathsf{r.dk_{ID,t}}, \mathsf{h.sk_{ID,t}})$ for time period t, except that if $\mathsf{r.dk_{ID,t}} = \perp$, then it returns the special "invalid" symbol $\perp$ indicating that ID has been revoked.

$\mathsf{Decrypt}(\mathsf{PP}, \mathsf{dk_{ID,t}}, \mathsf{ct}) \to \mathsf{M}$: It takes a public parameter $\mathsf{PP} = (\mathsf{r.PP}, \mathsf{h.PP})$, a decryption key $\mathsf{dk_{ID,t}} = (\mathsf{r.dk_{ID,t}}, \mathsf{h.sk_{ID,t}})$ and a ciphertext $\mathsf{ct} = (\mathsf{r.ct}, \mathsf{h.ct})$ as input, and then runs $\mathsf{r.M} \leftarrow \mathsf{r.Decrypt}(\mathsf{r.PP}, \mathsf{r.dk_{ID,t}}, \mathsf{r.ct})$ and $\mathsf{h.M} \leftarrow \mathsf{h.Decrypt}(\mathsf{h.PP}, \mathsf{h.sk_{ID,t}}, \mathsf{h.ct})$. If $\mathsf{r.M} = \perp$ or $\mathsf{h.M} = \perp$, then it returns $\perp$. Otherwise, it outputs the decryption result $\mathsf{M} := \mathsf{r.M} + \mathsf{h.M}$.

It is immediate to see that the correctness of the constructed RIBE scheme Π follows from that of the building blocks. The security of Π is guaranteed by the following theorem.

Theorem 1. *If the underlying RIBE scheme* $\mathsf{r}.\Pi$ *satisfies weak selective-identity (resp. weak adaptive-identity) security and the underlying 2-level HIBE scheme* $\mathsf{h}.\Pi$ *satisfies selective-identity (resp. adaptive-identity) security, then the resulting RIBE scheme* Π *satisfies selective-identity (resp. adaptive-identity) security.*

Proof of Theorem 1. Since the proof for the selective-identity security and that for adaptive-identity security are essentially the same, we only show the proof for the former.

Let us call a query made by an adversary *valid* if the answer to the query by the challenger is not $\perp$. We consider following two attack strategies of an adversary against the RIBE scheme Π that are mutually exclusive and cover all possibilities:

- Type-I: The adversary issues a valid secret key reveal query on ID^*.
- Type-II: The adversary does not issue a valid secret key reveal query on ID^*.

Whether an adversary has deviated from one strategy, is easy to detect. By Lemma 8, in order to prove the theorem, it is sufficient to show that for each type of adversary (that is promised to follow the attack strategy), its selective-identity advantage is negligible. We show it in the following lemmas.

Lemma 9. *For every PPT Type-I adversary* $\mathcal{A}_1$*, there exists a PPT adversary* $\mathcal{B}_1$ *against the weak-selective security of the underlying RIBE scheme* $\mathsf{r}.\Pi$ *such that* $\mathsf{Adv}^{\mathtt{RIBE\text{-}sel}}_{\Pi,\mathcal{A}_1}(\lambda) = \mathsf{Adv}^{\mathtt{RIBE\text{-}sel\text{-}weak}}_{\mathsf{r}.\Pi,\mathcal{B}_1}(\lambda)$.

Proof of Lemma 9. Let $\mathcal{A}_1$ be any PPT Type-I adversary. First of all, recall that the condition of the secret key reveal queries says that if ID^* has not been revoked before t^* (i.e. $\mathsf{ID}^* \notin \mathsf{RL_{t^*}}$), then a secret key reveal query on ID^* made after t^* cannot be valid. Recall also that the condition of the revoke & key update queries implies that if an adversary has made a valid secret key reveal query on ID^* before t^* and $\mathsf{t_{cu}} \geq \mathsf{t}^*$, then $\mathsf{ID}^* \in \mathsf{RL}_{\mathsf{t}^*}$. Hence, if $\mathsf{t_{cu}} \geq \mathsf{t}^*$, then we must have $\mathsf{ID}^* \in \mathsf{RL}_{\mathsf{t}^*}$. This fact will be used in this proof.

Now, using $\mathcal{A}_1$ as a building block, we construct a PPT adversary $\mathcal{B}_1$ that attacks the weak selective-identity security of the underlying RIBE scheme $\mathsf{r}.\Pi$ with the claimed advantage. The description of $\mathcal{B}_1$ is as follows:

At the beginning, $\mathcal{A}_1$ declares its challenge identity/time period pair $(\mathsf{ID}^*, \mathsf{t}^*)$. $\mathcal{B}_1$ sends the pair $(\mathsf{ID}^*, \mathsf{t}^*)$ as its own challenge identity/time period pair to $\mathcal{B}_1$'s challenger $\mathsf{r}.\mathcal{C}$, and then receives the public parameter $\mathsf{r.PP}$ and the key update $\mathsf{r.ku}_1$ from $\mathsf{r}.\mathcal{C}$. $\mathcal{B}_1$ runs $(\mathsf{h.PP}, \mathsf{h.sk_{kgc}}) \leftarrow \mathsf{h.Setup}(1^\lambda)$, and gives $\mathsf{PP} := (\mathsf{r.PP}, \mathsf{h.PP})$ and $\mathsf{ku}_1 := \mathsf{r.ku}_1$ to $\mathcal{A}_1$. Also, $\mathcal{B}_1$ initializes the counter $\mathsf{t_{cu}} := 1$ (which will always be synchronized by the one maintained by $\mathsf{r}.\mathcal{C}$), and also generates an empty list $\mathtt{SKList}_{\mathcal{B}}$ into which identity/secret key pairs $(\mathsf{ID}, \mathsf{sk_{ID}})$ that are known to $\mathcal{B}_1$, will be stored. From this point on, $\mathcal{A}_1$ starts making queries.

For a secret key generation query $\mathsf{ID} \in \mathcal{ID}$ from $\mathcal{A}_1$, $\mathcal{B}_1$ makes a secret key generation query ID to $\mathsf{r}.\mathcal{C}$. (Note that upon this query, $\mathsf{r}.\mathcal{C}$ executes $(\mathsf{r.sk_{ID}}, \mathsf{r.sk'_{kgc}}) \leftarrow \mathsf{r.GenSK}(\mathsf{r.PP}, \mathsf{r.sk_{kgc}}, \mathsf{ID})$, but returns nothing to $\mathcal{B}_1$.) Right after this, $\mathcal{B}_1$ further makes a secret key reveal query ID to $\mathsf{r}.\mathcal{C}$, and receives $\mathsf{r.sk_{ID}}$ from $\mathsf{r}.\mathcal{C}$. Then, $\mathcal{B}_1$ generates $\mathsf{h.sk_{ID}} \leftarrow \mathsf{h.GenSK}(\mathsf{h.sk_{kgc}}, \mathsf{ID})$. Finally, $\mathcal{B}_1$ sets $\mathsf{sk_{ID}} := (\mathsf{r.sk_{ID}}, \mathsf{h.sk_{ID}})$, and adds $(\mathsf{ID}, \mathsf{sk_{ID}})$ into the list $\mathtt{SKList}_{\mathcal{B}}$ (and returns nothing to $\mathcal{A}_1$).

For a secret key reveal query $\mathsf{ID} \in \mathcal{ID}$ from $\mathcal{A}_1$, $\mathcal{B}_1$ does the same check as the challenger in the selective-identity security game does. Namely, $\mathcal{B}_1$ checks that if $\mathsf{t_{cu}} \geq \mathsf{t}^*$ and $\mathsf{ID}^* \notin \mathsf{RL}_{\mathsf{t}^*}$ then $\mathsf{ID} \neq \mathsf{ID}^*$. If this is *not* satisfied, then $\mathcal{B}_1$ returns $\perp$ to $\mathcal{A}_1$. Otherwise, it is guaranteed that $(\mathsf{ID}, \mathsf{sk_{ID}})$ is contained in the list $\mathtt{SKList}_{\mathcal{B}}$, and thus $\mathcal{B}_1$ returns $\mathsf{sk_{ID}}$ to $\mathcal{A}_1$.

For a revoke & key update query $\mathsf{RL} \subset \mathcal{ID}$ from $\mathcal{A}$, $\mathcal{B}_1$ forwards RL to $\mathsf{r}.\mathcal{C}$, and receives the result $\mathsf{r.ku}_{\mathsf{t_{cu}}}$ (which may be $\perp$) from $\mathsf{r}.\mathcal{C}$. If the answer from $\mathsf{r}.\mathcal{C}$ is $\perp$, then $\mathcal{B}_1$ returns $\perp$ to $\mathcal{A}_1$. Otherwise, $\mathsf{r}.\mathcal{C}$ has incremented the counter $\mathsf{t_{cu}}$, and thus so does $\mathcal{B}_1$ (which ensures that the counter $\mathsf{t_{cu}}$ maintained by $\mathcal{B}_1$ and that maintained by $\mathsf{r}.\mathcal{C}$ are synchronized). Then, $\mathcal{B}_1$ returns $\mathsf{ku}_{\mathsf{t_{cu}}} := \mathsf{r.ku}_{\mathsf{t_{cu}}}$ to $\mathcal{A}$. Here, as mentioned at he beginning of the proof of this lemma, if $\mathsf{t_{cu}} \geq \mathsf{t}^*$, then it is guaranteed that $\mathsf{ID}^* \in \mathsf{RL}_{\mathsf{t}^*}$.

For a decryption key reveal query $(\mathsf{ID}, \mathsf{t}) \in \mathcal{ID} \times \mathcal{T}$ from $\mathcal{A}_1$, $\mathcal{B}_1$ does the checks in the same way as the challenger in the selective-identity security game does. Namely, whether $\mathsf{t} \leq \mathsf{t_{cu}}$, $\mathsf{ID} \notin \mathsf{RL_t}$, and $(\mathsf{ID}, \mathsf{t}) \neq (\mathsf{ID}^*, \mathsf{t}^*)$ hold. If these conditions are *not* satisfied simultaneously, then $\mathcal{B}_1$ returns $\perp$ to $\mathcal{A}_1$. Otherwise, it is guaranteed that $\mathcal{B}_1$ has already obtained $\mathsf{ku_t} = \mathsf{r.ku_t}$ from $\mathsf{r}.\mathcal{C}$, and $\mathcal{B}_1$ owns $\mathsf{sk_{ID}} = (\mathsf{r.sk_{ID}}, \mathsf{h.sk_{ID}})$ in $\mathtt{SKList}_{\mathcal{B}}$ (because $\mathcal{A}_1$ must have made a secret key generation query on ID, in which case $\mathcal{B}_1$ has obtained $\mathsf{sk_{ID}}$ in the response to the query). Using $\mathsf{r.ku_t}$ and $\mathsf{r.sk_{ID}}$, $\mathcal{B}_1$ runs $\mathsf{r.dk_{ID,t}} \leftarrow \mathsf{r.GenDK}(\mathsf{r.PP}, \mathsf{r.sk_{ID}}, \mathsf{r.ku_t})$. $\mathcal{B}_1$ also runs $\mathsf{h.sk_{ID,t}} \leftarrow \mathsf{h.Delegate}(\mathsf{h.PP}, \mathsf{h.sk_{ID}}, \mathsf{t})$. Finally, $\mathcal{B}_1$ returns $\mathsf{dk_{ID,t}} := (\mathsf{r.dk_{ID,t}}, \mathsf{h.sk_{ID,t}})$ to $\mathcal{A}_1$.

For the challenge query $(\mathsf{M}_0, \mathsf{M}_1)$ from $\mathcal{A}_1$, $\mathcal{B}_1$ picks $\mathsf{h.M} \in \mathcal{M}$ uniformly at random, and then sets $\mathsf{r.M}_0 \leftarrow \mathsf{M}_0 - \mathsf{h.M}$ and $\mathsf{r.M}_1 \leftarrow \mathsf{M}_1 - \mathsf{h.M}$. Then, $\mathcal{B}_1$ submits the challenge query $(\mathsf{r.M}_0, \mathsf{r.M}_1)$ to $\mathsf{r}.\mathcal{C}$, and receives $\mathcal{B}_1$'s challenge ciphertext $\mathsf{r.ct}^* \leftarrow \mathsf{r.Encrypt}(\mathsf{r.PP}, \mathsf{ID}^*, \mathsf{t}^*, \mathsf{r.M}_\beta)$ from $\mathsf{r}.\mathcal{C}$, where β is $\mathcal{B}_1$'s challenge bit. $\mathcal{B}_1$ also executes $\mathsf{h.ct}^* \leftarrow \mathsf{h.Encrypt}(\mathsf{h.PP}, (\mathsf{ID}^*, \mathsf{t}^*), \mathsf{h.M})$. Finally, $\mathcal{B}_1$ returns the challenge ciphertext $\mathsf{ct}^* := (\mathsf{r.ct}^*, \mathsf{h.ct}^*)$ to $\mathcal{A}_1$.

Eventually, $\mathcal{A}_1$ terminates with output its guess bit b'. Then, $\mathcal{B}_1$ sets $\beta' \leftarrow b'$, and terminates with output β'.

The above completes the description of $\mathcal{B}_1$. Note that $\mathcal{B}_1$ simulates the selective-identity security game perfectly for the Type-I adversary $\mathcal{A}_1$ so that $\mathcal{B}_1$'s challenge bit β is that of $\mathcal{A}_1$'s (i.e. the plaintext encrypted in ct^* is M_β). Since $\mathcal{B}_1$ uses $\mathcal{A}_1$'s final output b' as its own guess β', the probability that $\mathcal{B}_1$ succeeds in guessing $\mathcal{B}_1$'s challenge bit is the same as the probability that $\mathcal{A}_1$ succeeds in guessing the challenge bit in the selective-identity security game. Hence, we have $\mathsf{Adv}^{\mathtt{RIBE\text{-}sel\text{-}weak}}_{\mathsf{r}.\Pi,\mathcal{B}_1}(\lambda) = \mathsf{Adv}^{\mathtt{RIBE\text{-}sel}}_{\Pi,\mathcal{A}_1}(\lambda)$, as desired. This completes the proof of Lemma 9. □

Lemma 10. *For any Type-II adversary $\mathcal{A}_2$, there exists a PPT adversary $\mathcal{B}_2$ against the selective-identity security of the underlying 2-level HIBE scheme $\mathsf{h}.\Pi$ such that $\mathsf{Adv}^{\mathtt{RIBE\text{-}sel}}_{\Pi,\mathcal{A}_2}(\lambda) = \mathsf{Adv}^{\mathtt{HIBE\text{-}sel}}_{\mathsf{h}.\Pi,\mathcal{B}_2}(\lambda)$.*

Proof of Lemma 10. Let $\mathcal{A}_2$ be any PPT Type-II adversary. Using $\mathcal{A}_2$ as a building block, we construct a PPT adversary $\mathcal{B}_2$ that attacks the selective-identity security of the underlying 2-level HIBE scheme $\mathsf{h}.\Pi$ with the claimed advantage. The description of $\mathcal{B}_2$ is as follows:

At the beginning, $\mathcal{A}_2$ declares its challenge identity/time period pair $(\mathsf{ID}^*, \mathsf{t}^*)$. $\mathcal{B}_2$ sends the pair $(\mathsf{ID}^*, \mathsf{t}^*)$ as its own challenge (2-level hierarchical) identity to $\mathcal{B}_2$'s challenger $\mathsf{h}.\mathcal{C}$, and receives the public parameter $\mathsf{h.PP}$ from $\mathsf{h}.\mathcal{C}$. $\mathcal{B}_2$ initializes the counter $\mathsf{t_{cu}} := 1$, and then runs $(\mathsf{r.PP}, \mathsf{r.sk_{kgc}}) \leftarrow \mathsf{r.Setup}(1^\lambda)$ and $(\mathsf{r.ku}_1, \mathsf{r.sk}'_{\mathsf{kgc}}) \leftarrow \mathsf{r.KeyUp}(\mathsf{r.PP}, 1, \mathsf{r.sk_{kgc}}, \mathsf{RL}_1 = \emptyset)$. Then, $\mathcal{B}_2$ gives $\mathsf{PP} := (\mathsf{r.PP}, \mathsf{h.PP})$ and $\mathsf{ku}_1 := \mathsf{r.ku}_1$ to $\mathcal{A}_2$. From this point on, $\mathcal{A}_2$ starts making queries.

For a secret key generation query $\mathsf{ID} \in \mathcal{ID}$ from $\mathcal{A}_2$, $\mathcal{B}_2$ forwards ID to $\mathsf{h}.\mathcal{C}$ as a level-1 secret key generation query. (Note that by this query, $\mathsf{h}.\mathcal{C}$ executes $\mathsf{h.sk_{ID}} \leftarrow \mathsf{h.GenSK}(\mathsf{h.PP}, \mathsf{h.sk_{kgc}}, \mathsf{ID})$, but returns nothing to $\mathcal{B}_2$.) Also, $\mathcal{B}_2$ generates $(\mathsf{r.sk_{ID}}, \mathsf{r.sk}'_{\mathsf{kgc}}) \leftarrow \mathsf{r.GenSK}(\mathsf{r.sk_{kgc}}, \mathsf{ID})$, and keeps it to itself.

For a secret key reveal query $\mathsf{ID} \in \mathcal{ID}$ from $\mathcal{A}_2$, $\mathcal{B}_2$ does the same check as the challenger in the selective-identity security game does. Namely, $\mathcal{B}_2$ checks that if $\mathsf{t_{cu}} \ge \mathsf{t}^*$ and $\mathsf{ID}^* \notin \mathsf{RL}_{\mathsf{t_{cu}}}$, then $\mathsf{ID} \ne \mathsf{ID}^*$. If this is *not* satisfied, then $\mathcal{B}_2$ returns $\perp$ to $\mathcal{A}_2$. Otherwise, it is guaranteed that $\mathcal{A}_2$'s query is valid. At this point, it is guaranteed that $\mathsf{ID} \ne \mathsf{ID}^*$ because $\mathcal{A}_2$ is of Type-II. $\mathcal{B}_2$ submits a level-1 secret key reveal query ID to $\mathsf{h}.\mathcal{C}$, and receives $\mathsf{h.sk_{ID}}$ from $\mathsf{h}.\mathcal{C}$. $\mathcal{B}_2$ also finds $\mathsf{r.sk_{ID}}$ that $\mathcal{B}_2$ has already generated, and returns $\mathsf{sk_{ID}} := (\mathsf{r.sk_{ID}}, \mathsf{h.sk_{ID}})$ to $\mathcal{A}_2$.

For a revoke & key update query $\mathsf{RL} \subset \mathcal{ID}$ from $\mathcal{A}_2$, $\mathcal{B}_2$ responds to it in exactly the same way as the challenger in the selective-identity security game does, which is possible because $\mathcal{B}_2$ possesses $\mathsf{r.sk_{kgc}}$. (Note that if the query is valid, then the counter $\mathsf{t_{cu}}$ is incremented, and a key update $\mathsf{ku}_{\mathsf{t_{cu}}} := \mathsf{r.ku_t}$ is generated.)

For a decryption key reveal query $(\mathsf{ID}, \mathsf{t}) \in \mathcal{ID} \times \mathcal{T}$ from $\mathcal{A}_2$, $\mathcal{B}_2$ does the checks in the same way as the challenger in the selective-identity security game does. Namely, $\mathcal{B}_2$ checks whether $\mathsf{t} \le \mathsf{t_{cu}}$, $\mathsf{ID} \notin \mathsf{RL_t}$, and $(\mathsf{ID}, \mathsf{t}) \ne (\mathsf{ID}^*, \mathsf{t}^*)$ hold simultaneously. If these conditions are *not* satisfied, then $\mathcal{B}_2$ returns $\perp$ to $\mathcal{A}_2$. Otherwise, it is guaranteed that $\mathcal{B}_2$ has already generated $\mathsf{ku_t} = \mathsf{r.ku_t}$ and $\mathsf{r.sk_{ID}}$. Using $\mathsf{r.ku_t}$ and $\mathsf{r.sk_{ID}}$, $\mathcal{B}_2$ runs $\mathsf{r.dk_{ID,t}} \leftarrow \mathsf{r.GenDK}(\mathsf{r.PP}, \mathsf{r.sk_{ID}}, \mathsf{r.ku_t})$. $\mathcal{B}_2$ also makes a 2-level secret key reveal query $(\mathsf{ID}, \mathsf{t})$, and receives $\mathsf{h.sk_{ID,t}}$ from $\mathsf{h}.\mathcal{C}$. Finally, $\mathcal{B}_2$ returns $\mathsf{dk_{ID,t}} := (\mathsf{r.dk_{ID,t}}, \mathsf{h.sk_{ID,t}})$ to $\mathcal{A}_2$.

For the challenge query $(\mathsf{M}_0, \mathsf{M}_1)$ from $\mathcal{A}_2$, $\mathcal{B}_2$ picks $\mathsf{r.M} \in \mathcal{M}$ uniformly at random, and then sets $\mathsf{h.M}_0 \leftarrow \mathsf{M}_0 - \mathsf{r.M}$ and $\mathsf{h.M}_1 \leftarrow \mathsf{M}_1 - \mathsf{r.M}$. Then, $\mathcal{B}_2$ submits the challenge query $(\mathsf{h.M}_0, \mathsf{h.M}_1)$ to $\mathsf{h}.\mathcal{C}$, and receives $\mathcal{B}_2$'s challenge ciphertext $\mathsf{h.ct}^* \leftarrow \mathsf{h.Encrypt}(\mathsf{h.PP}, (\mathsf{ID}^*, \mathsf{t}^*), \mathsf{h.M}_\beta)$ from $\mathsf{r}.\mathcal{C}$, where β is $\mathcal{B}_2$'s challenge bit. $\mathcal{B}_2$ also executes $\mathsf{r.ct}^* \leftarrow \mathsf{r.Encrypt}(\mathsf{r.PP}, \mathsf{ID}^*, \mathsf{t}^*, \mathsf{r.M})$. Finally, $\mathcal{B}_2$ returns the challenge ciphertext $\mathsf{ct}^* := (\mathsf{r.ct}^*, \mathsf{h.ct}^*)$ to $\mathcal{A}_2$.

Eventually, $\mathcal{A}_2$ terminates with output its guess bit b'. Then, $\mathcal{B}_2$ sets $\beta' \leftarrow b'$, and terminates with output β'.

The above completes the description of $\mathcal{B}_2$. Note that $\mathcal{B}_2$ never falls into the situation in which $\mathcal{B}_2$ has to make a level-1 secret key reveal query on ID^* or a level-2 secret key reveal query on

$(\mathsf{ID}^*, \mathsf{t}^*)$. Note also that $\mathcal{B}_2$ simulates the selective-identity security game perfectly for $\mathcal{A}_2$ so that $\mathcal{B}_2$'s challenge bit β is that of $\mathcal{A}_2$'s (i.e. the plaintext encrypted in ct^* is M_β). Since $\mathcal{B}_2$ uses $\mathcal{A}_2$'s final output b' as its own guess β', the probability that $\mathcal{B}_2$ succeeds in guessing $\mathcal{B}_2$'s challenge bit is the same as the probability that $\mathcal{A}_2$ succeeds in guessing the challenge bit in the selective-identity security game. Hence, we have $\mathsf{Adv}^{\texttt{HIBE-sel}}_{\mathsf{h}.\Pi,\mathcal{B}_2}(\lambda) = \mathsf{Adv}^{\texttt{RIBE-sel}}_{\Pi,\mathcal{A}_2}(\lambda)$, as desired. This completes the proof of Lemma 10. □

Due to Lemmas 8, 9, and 10, we can conclude that the RIBE scheme Π satisfies selective-identity security. This completes the proof of Theorem 1. □

6 RHIBE from Lattices

In this section, we first explain our treatment on binary trees, the CS method, and the parameters used in the scheme. Then, we show our proposed scheme in Section 6.1 and discuss the security in Section 6.2.

On the Treatment of Binary Trees and the CS Method. Every user ID such that $|\mathsf{ID}| \leq L-1$ (including KGC) keeps a binary tree $\mathtt{BT}_{\mathsf{ID}}$ as a part of his secret key $\mathsf{sk}_{\mathsf{ID}}$, which is used to manage his children. The binary trees have at least N leaves and less than $2N$ leaves, where N denotes the maximum number of children ID managed by the parent $\mathsf{pa}(\mathsf{ID})$. We use θ to denote a node in a binary tree, where η especially denotes a leaf node. To utilize the complete subtree (CS) method, each child user ID is assigned to a randomly selected leaf η_{ID} in the binary tree $\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}$ that is managed by his parent $\mathsf{pa}(\mathsf{ID})$. In addition, to achieve the revocation mechanism, each node $\theta \in \mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}$ may be associated with a random vector $\mathbf{u}_{\mathsf{pa}(\mathsf{ID}),\theta} \in \mathbb{Z}_q^n$. Although we do not mention explicitly, if we write $\mathtt{BT}_{\mathsf{ID}}$, we assume it contains the description of the nodes (e.g., as natural numbers) along with the leaf assignments of children users ID and the random vectors $\mathbf{u}_{\mathsf{ID},\theta}$. Let $\mathsf{Path}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \eta_{\mathsf{ID}})$ denote a path from the root to the leaf η_{ID}, where the nodes $\theta \in \mathsf{Path}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \eta_{\mathsf{ID}})$ are associated with a secret key $\mathsf{sk}_{\mathsf{ID}}$ whose size is $O(\log N)$. Let $\mathsf{KUNode}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}})$ denote a set of nodes that are associated with a key update $\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}$ whose size is $O(R\log(N/R))$ where R denotes the number of revoked users. Each user ID extracts a part of the secret key $\mathsf{sk}_{\mathsf{ID}}$ and the key update $\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}$ that are associated with a node $\theta \in \mathsf{Path}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \eta_{\mathsf{ID}}) \cap \mathsf{KUNode}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}})$, then derives his decryption key $\mathsf{dk}_{\mathsf{ID},\mathsf{t}}$. In this paper, we define the CS method as the following four algorithms ($\mathsf{CS.SetUp}$, $\mathsf{CS.Assign}$, $\mathsf{CS.Cover}$, $\mathsf{CS.Match}$):

$\mathsf{CS.SetUp}(N) \to \mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}$: on input the number of users N, it outputs a binary tree $\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}$.

$\mathsf{CS.Assign}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \mathsf{ID}) \to (\eta_{\mathsf{ID}}, \mathtt{BT}_{\mathsf{pa}(\mathsf{ID})})$: on input a binary tree $\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}$ and an identity ID, it randomly assigns a leaf node η_{ID}, which no other ID's are still assigned to, to the identity ID. Then, it outputs the leaf η_{ID} and also the "updated" binary tree $\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}$.

$\mathsf{CS.Cover}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}) \to \mathsf{KUNode}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}})$: on input a binary tree $\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}$ and a revocation list $\mathsf{RL}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}$, it outputs a set of nodes $\mathsf{KUNode}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}})$ where the subtrees with root $\theta \in \mathsf{KUNode}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}})$ cover all leaves η_{ID} in $\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}$ for $\mathsf{ID} \notin \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}$ and do not cover any leaves η_{ID} for $\mathsf{ID} \in \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}$.

$\mathsf{CS.Match}(\mathsf{Path}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \eta_{\mathsf{ID}}), \mathsf{KUNode}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}})) \to \theta$ or $\emptyset$: on input $\mathsf{Path}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \eta_{\mathsf{ID}})$ and $\mathsf{KUNode}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}})$, it outputs an arbitrary node $\theta \in \mathsf{Path}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \eta_{\mathsf{ID}}) \cap \mathsf{KUNode}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}})$ if it exists. Otherwise, it outputs $\emptyset$.

Parameters. Let L denote the maximum depth of the hierarchy and N denote the maximum number of children each parent manages. Furthermore, let n, m, q be positive integers such that q is a prime and $\alpha, \alpha', (\sigma_i)_{i=0}^{L}$ be positive reals denoting the Gaussian parameters. We note that all parameters are implicitly a function of the security parameter λ, in particular we set $n(\lambda) = \lambda$, and for the other parameters, a concrete parameter selection is provided in Section 6.1. Finally, we set the plaintext space as $\mathcal{M} = \{0,1\}$, the element identity space as $\mathcal{ID} = \mathbb{Z}_q^n \setminus \{\mathbf{0}_n\}$, and the hierarchal identity space as $\mathcal{ID}_{\mathtt{h}} := (\mathbb{Z}_q^n \setminus \{\mathbf{0}_n\})^{\leq L}$. We also encode the time period space $\mathcal{T} = \{1, 2, \cdots, \mathsf{t}_{\max}\}$ into a polynomial sized subset of $\mathbb{Z}_q^n$. In the following, for readability, we may simply address each space $\mathcal{ID}, \mathcal{ID}_{\mathtt{h}}, \mathcal{T}$ as $\mathcal{T} = \mathcal{ID} = \mathbb{Z}_q^n \setminus \{\mathbf{0}_n\}, \mathcal{ID}_{\mathtt{h}} = (\mathbb{Z}_q^n \setminus \{\mathbf{0}_n\})^{\leq L}$, unless stated otherwise.

6.1 Construction

We provide our RHIBE scheme below. The intuition of the construction follows the explanation given in Section 2. Due to the contrived nature of our scheme, we encourage readers to go back to Section 2 whenever needed.

$\mathsf{Setup}(1^n, L) \to (\mathsf{PP}, \mathsf{sk}_{\mathsf{kgc}})$: The setup algorithm is run by the KGC. It takes the security parameter 1^n and the maximum depth of the hierarchy L as input, and runs $(\mathbf{A}_i, \mathbf{T}_{\mathbf{A}_i}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ for $i \in [L+1]$. It also samples uniformly random matrices $(\mathbf{B}_j)_{j \in [L+1]} \leftarrow (\mathbb{Z}_q^{n \times m})^{(L+1)}$ and vectors $(\mathbf{u}_k)_{k \in [L]} \leftarrow (\mathbb{Z}_q^n)^L$. Finally, it creates a binary tree by running $\mathsf{BT}_{\mathsf{kgc}} \leftarrow \mathsf{CS.SetUp}(N)$ and outputs

$$\mathsf{PP} := \Big((\mathbf{A}_i)_{i \in [L+1]}, (\mathbf{B}_j)_{j \in [L+1]}, (\mathbf{u}_k)_{k \in [L]} \Big), \quad \mathsf{sk}_{\mathsf{kgc}} := \Big(\mathsf{BT}_{\mathsf{kgc}}, (\mathbf{T}_{\mathbf{A}_i})_{i \in [L+1]} \Big).$$

Recall here that the matrices $\mathbf{B}_j$ define the hash functions $\mathbf{E}(\cdot)$ and $\mathbf{F}(\cdot)$ stated in Section 2.

$\mathsf{Encrypt}(\mathsf{PP}, \mathsf{ID} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell), \mathsf{t}, \mathsf{M}) \to \mathsf{ct}$: On input an identity $\mathsf{ID} \in (\mathbb{Z}_q^n)^\ell$ at depth $\ell \in [L]$ and time period $\mathsf{t} \in \mathbb{Z}_q^n$, it first samples $\ell + 1$ uniformly random vectors $(\mathbf{s}_i)_{i \in [\ell]}, \mathbf{s}_{L+1} \in \mathbb{Z}_q^n$. Then it samples $x \leftarrow D_{\mathbb{Z}, \alpha q}, \mathbf{x}_i \leftarrow D_{\mathbb{Z}^{(i+2)m}, \alpha' q}$ for $i \in [\ell]$ and $\mathbf{x}_{L+1} \leftarrow D_{\mathbb{Z}^{(\ell+2)m}, \alpha' q}$, and sets

$$\begin{cases} c_0 = \mathbf{u}_\ell^\top (\mathbf{s}_1 + \cdots + \mathbf{s}_\ell + \mathbf{s}_{L+1}) + x + \mathsf{M} \left\lfloor \dfrac{q}{2} \right\rfloor \\ \mathbf{c}_i = [\mathbf{A}_i | \mathbf{E}(\mathsf{ID}_{[i]}) | \mathbf{F}(\mathsf{t})]^\top \mathbf{s}_i + \mathbf{x}_i \quad \text{for } \ i \in [\ell] \\ \mathbf{c}_{L+1} = [\mathbf{A}_{L+1} | \mathbf{E}(\mathsf{ID}) | \mathbf{F}(\mathsf{t})]^\top \mathbf{s}_{L+1} + \mathbf{x}_{L+1} \end{cases}$$

Finally, it outputs a ciphertext

$$\mathsf{ct} := (c_0, \mathbf{c}_1, \ldots, \mathbf{c}_\ell, \mathbf{c}_{L+1}) \in \mathbb{Z}_q \times \mathbb{Z}_q^{3m} \times \cdots \times \mathbb{Z}_q^{(\ell+2)m} \times \mathbb{Z}_q^{(\ell+2)m}.$$

$\mathsf{GenSK}(\mathsf{PP}, \mathsf{sk}_{\mathsf{pa(ID)}}, \mathsf{ID}) \to (\mathsf{sk}_{\mathsf{ID}}, \mathsf{sk}'_{\mathsf{pa(ID)}})$: The secret key generation algorithm is run by a parent user $\mathsf{pa(ID)}$ at level $\ell - 1$, where $1 \leq \ell \leq L$, to create a secret key for its child ID.[13] It first runs $(\mathsf{BT}_{\mathsf{pa(ID)}}, \eta_{\mathsf{ID}}) \leftarrow \mathsf{CS.Assign}(\mathsf{BT}_{\mathsf{pa(ID)}}, \mathsf{ID})$. Then, for each node $\theta \in \mathsf{Path}(\mathsf{BT}_{\mathsf{pa(ID)}}, \eta_{\mathsf{ID}})$, it checks whether a vector $\mathbf{u}_{\mathsf{pa(ID)}, \theta} \in \mathbb{Z}_q^n$ has already been assigned. If not, pick a uniformly random vector $\mathbf{u}_{\mathsf{pa(ID)}, \theta} \in \mathbb{Z}_q^n$ and update $\mathsf{sk}_{\mathsf{pa(ID)}}$ by storing $\mathbf{u}_{\mathsf{pa(ID)}, \theta}$ in node $\theta \in \mathsf{BT}_{\mathsf{pa(ID)}}$. Next, it samples vectors $\mathbf{e}_{\mathsf{ID}, \theta}, \mathbf{f}_{\mathsf{ID}, k} \in \mathbb{Z}^{(\ell+1)m}$ for $\theta \in \mathsf{Path}(\mathsf{BT}_{\mathsf{pa(ID)}}, \eta_{\mathsf{ID}}), k \in [\ell+1, L]$, respectively, such that

$$[\mathbf{A}_\ell | \mathbf{E}(\mathsf{ID})] \mathbf{e}_{\mathsf{ID}, \theta} = \mathbf{u}_{\mathsf{pa(ID)}, \theta}, \quad [\mathbf{A}_\ell | \mathbf{E}(\mathsf{ID})] \mathbf{f}_{\mathsf{ID}, k} = \mathbf{u}_k - \mathbf{u}_\ell$$

[13] Recall that a user at level 0 corresponds to the kgc, i.e., for any level-1 user $\mathsf{ID} \in \mathbb{Z}_q^n \setminus \{\mathbf{0}_n\}$, $\mathsf{pa(ID)} = \mathsf{kgc}$.

by running $\mathsf{SampleLeft}(\cdot)$ with trapdoor $\mathbf{T}_{[\mathbf{A}_\ell|\mathbf{E}(\mathsf{pa}(\mathsf{ID}))]}$[14] and Gaussian parameter σ_ℓ. Then, it extends its bases by running the following algorithm for $i \in [\ell+1, L+1]$

$$\mathbf{T}_{[\mathbf{A}_i|\mathbf{E}(\mathsf{ID})]} \leftarrow \mathsf{ExtRndLeft}([\mathbf{A}_i|\mathbf{E}(\mathsf{pa}(\mathsf{ID}))],\ \mathbf{B}_\ell + H(\mathsf{id}_\ell)\mathbf{G},\ \mathbf{T}_{[\mathbf{A}_i|\mathbf{E}(\mathsf{pa}(\mathsf{ID}))]},\ \sigma_{\ell-1}),$$

where $\mathbf{T}_{[\mathbf{A}_i|\mathbf{E}(\mathsf{ID})]} \in \mathbb{Z}^{(\ell+1)m\times(\ell+1)m}$. Here, recall that $\mathbf{E}(\mathsf{ID}) = [\mathbf{E}(\mathsf{pa}(\mathsf{ID}))|\mathbf{B}_\ell + H(\mathsf{id}_\ell)\mathbf{G}]$. Finally, it runs $\mathtt{BT}_{\mathsf{ID}} \leftarrow \mathsf{CS.SetUp}(N)$ and outputs,

$$\mathsf{sk}_{\mathsf{ID}} = \left(\begin{array}{c} \mathtt{BT}_{\mathsf{ID}}, \mathsf{Path}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \eta_{\mathsf{ID}}),\ (\mathbf{e}_{\mathsf{ID},\theta})_{\theta\in\mathsf{Path}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})},\eta_{\mathsf{ID}})}, \\ (\mathbf{f}_{\mathsf{ID},k})_{k\in[\ell+1,L]},\ (\mathbf{T}_{[\mathbf{A}_i|\mathbf{E}(\mathsf{ID})]})_{i\in[\ell+1,L+1]} \end{array} \right)$$

along with its updated secret key $\mathsf{sk}'_{\mathsf{pa}(\mathsf{ID})}$.

$\mathsf{KeyUp}(\mathsf{PP}, \mathsf{t}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{RL}_{\mathsf{ID},\mathsf{t}}, \mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}) \rightarrow (\mathsf{ku}_{\mathsf{ID},\mathsf{t}}, \mathsf{sk}'_{\mathsf{ID}})$: The key update information generation algorithm is run by user ID at level ℓ, where $0 \leq \ell \leq L-1$, to create a key update $\mathsf{ku}_{\mathsf{ID},\mathsf{t}}$ for time period t for its children. It first runs $\mathsf{KUNode}(\mathtt{BT}_{\mathsf{ID}}, \mathsf{RL}_{\mathsf{ID},\mathsf{t}}) \leftarrow \mathsf{CS.Cover}(\mathtt{BT}_{\mathsf{ID}}, \mathsf{RL}_{\mathsf{ID},\mathsf{t}})$, and checks whether $\mathbf{u}_{\mathsf{ID},\theta}$ is defined for each node $\theta \in \mathsf{KUNode}(\mathtt{BT}_{\mathsf{ID}}, \mathsf{RL}_{\mathsf{ID},\mathsf{t}})$. If not, it picks a random $\mathbf{u}_{\mathsf{ID},\theta} \in \mathbb{Z}_q^n$ and updates $\mathsf{sk}_{\mathsf{ID}}$ by storing $\mathbf{u}_{\mathsf{ID},\theta}$ in the node $\theta \in \mathtt{BT}_{\mathsf{ID}}$. Then, for each node θ, it samples $\mathbf{e}_{\mathsf{ID},\mathsf{t},\theta} \in \mathbb{Z}^{(\ell+2)m}$ such that

$$[\mathbf{A}_{\ell+1}|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]\mathbf{e}_{\mathsf{ID},\mathsf{t},\theta} = \mathbf{u}_{\ell+1} - \mathbf{u}_{\mathsf{ID},\theta}$$

by running $\mathsf{SampleLeft}(\cdot)$ with trapdoor $\mathbf{T}_{[\mathbf{A}_{\ell+1}|\mathbf{E}(\mathsf{ID})]}$ and Gaussian parameter $\sigma_{\ell+1}$.
At this point, the algorithm behaves differently depending on $\ell \geq 1$ or $\ell = 0$ (i.e., $\mathsf{ID} = \mathsf{kgc}$). In case $\ell \geq 1$, it computes its own decryption key $\mathsf{dk}_{\mathsf{ID},\mathsf{t}}$, which includes a vector $\mathbf{d}_{\mathsf{ID},\mathsf{t}} \in \mathbb{Z}^{(\ell+2)m}$, using the decryption key generation algorithm $\mathsf{GenDK}(\mathsf{PP}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ku}_{\mathsf{pa}(\mathsf{ID},\mathsf{t})})$ defined below, and computes the following vectors for $k \in [\ell+1, L]$:

$$\mathbf{f}_{\mathsf{ID},\mathsf{t},k} = \mathbf{d}_{\mathsf{ID},\mathsf{t}} + [\mathbf{f}_{\mathsf{ID},k}\|\mathbf{0}_{m\times 1}] \in \mathbb{Z}^{(\ell+2)m}.$$

Here, $[\cdot\|\cdot]$ denotes vertical concatenation of vectors.
Finally, it extracts $(\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},k} \in \mathbb{Z}^{(i+2)m})_{(i,k)\in[\ell-1]\times[\ell+1,L]}$ from its ancestor's key update information $\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}$ and outputs

$$\mathsf{ku}_{\mathsf{ID},\mathsf{t}} = (\mathsf{KUNode}(\mathtt{BT}_{\mathsf{ID}}, \mathsf{RL}_{\mathsf{ID},\mathsf{t}}), (\mathbf{e}_{\mathsf{ID},\mathsf{t},\theta})_{\theta\in\mathsf{KUNode}(\mathtt{BT}_{\mathsf{ID}},\mathsf{RL}_{\mathsf{ID},\mathsf{t}})}, (\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},k})_{(i,k)\in[\ell]\times[\ell+1,L]})$$

and the possibly updated $\mathsf{sk}'_{\mathsf{ID}}$.
In case $\ell = 0$, it skips all the above procedures and simply outputs

$$\mathsf{ku}_{\mathsf{ID},\mathsf{t}} = (\mathsf{KUNode}(\mathtt{BT}_{\mathsf{ID}}, \mathsf{RL}_{\mathsf{ID},\mathsf{t}}), (\mathbf{e}_{\mathsf{ID},\mathsf{t},\theta})_{\theta\in\mathsf{KUNode}(\mathtt{BT}_{\mathsf{ID}},\mathsf{RL}_{\mathsf{ID}})})$$

and the possibly updated $\mathsf{sk}'_{\mathsf{ID}}$.[15]

$\mathsf{GenDK}(\mathsf{PP}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}) \rightarrow \mathsf{dk}_{\mathsf{ID},\mathsf{t}}$ or $\perp$: The decryption key generation algorithm is run by user ID at level ℓ, where $1 \leq \ell \leq L$. It extracts $\mathsf{Path}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \eta_{\mathsf{ID}})$ in $\mathsf{sk}_{\mathsf{ID}}$ and $\mathsf{KUNode}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}})$ in $\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}}$, and runs

[14] There are two exceptions for this algorithm. In the special case $\mathsf{ID} = \mathsf{kgc}$, recall that we set $\mathbf{T}_{[\mathbf{A}_1|\mathbf{E}(\mathsf{kgc})]}$ as $\mathbf{T}_{\mathbf{A}_1}$, which is included in the $\mathsf{sk}_{\mathsf{kgc}}$. In the other special case when $\ell = L$, we no longer sample $\mathbf{f}_{\mathsf{ID},k}$, since this vector is only required for delegating key updates to its children, which users at level L do not have.

[15] The branch in the algorithm is due to the fact that for the special case $\ell = 0$, i.e., $\mathsf{ID} = \mathsf{kgc}$, we have $\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t}} = \perp$ for all $\mathcal{T}$ and there exists no decryption key $\mathsf{dk}_{\mathsf{ID},\mathsf{t}}$.

$\theta/\emptyset \leftarrow \mathsf{CS.Match}(\mathsf{Path}(\mathsf{BT}_{\mathsf{pa(ID)}}, \eta_{\mathsf{ID}}), \mathsf{KUNode}(\mathsf{BT}_{\mathsf{pa(ID)}}, \mathsf{RL}_{\mathsf{pa(ID)},\mathsf{t}}))$. If the output is $\emptyset$, it outputs $\perp$. Otherwise, it extracts $\mathbf{e}_{\mathsf{ID},\theta}, \mathbf{e}_{\mathsf{pa(ID)},\mathsf{t},\theta} \in \mathbb{Z}^{(\ell+1)m}$ in $\mathsf{sk}_{\mathsf{ID}}, \mathsf{ku}_{\mathsf{pa(ID)},\mathsf{t}}$, respectively, and parse it as

$$\mathbf{e}_{\mathsf{ID},\theta} = [\mathbf{e}^{\mathsf{L}}_{\mathsf{ID},\theta} \| \mathbf{e}^{\mathsf{R}}_{\mathsf{ID},\theta}], \quad \mathbf{e}_{\mathsf{pa(ID)},\mathsf{t},\theta} = [\mathbf{e}^{\mathsf{L}}_{\mathsf{pa(ID)},\mathsf{t},\theta} \| \mathbf{e}^{\mathsf{R}}_{\mathsf{pa(ID)},\mathsf{t},\theta}],$$

where $\mathbf{e}^{\mathsf{L}}_{\mathsf{ID},\theta}, \mathbf{e}^{\mathsf{L}}_{\mathsf{pa(ID)},\mathsf{t},\theta} \in \mathbb{Z}^{\ell m}$ and $\mathbf{e}^{\mathsf{R}}_{\mathsf{ID},\theta}, \mathbf{e}^{\mathsf{R}}_{\mathsf{pa(ID)},\mathsf{t},\theta} \in \mathbb{Z}^{m}$. Then, it computes

$$\mathbf{d}_{\mathsf{ID},\mathsf{t}} = [\mathbf{e}^{\mathsf{L}}_{\mathsf{ID},\theta} + \mathbf{e}^{\mathsf{L}}_{\mathsf{pa(ID)},\mathsf{t},\theta} \| \mathbf{e}^{\mathsf{R}}_{\mathsf{ID},\theta} \| \mathbf{e}^{\mathsf{R}}_{\mathsf{pa(ID)},\mathsf{t},\theta}] \in \mathbb{Z}^{(\ell+2)m}.$$

It further samples $\mathbf{g}_{\mathsf{ID},\mathsf{t}} \in \mathbb{Z}^{(\ell+2)m}$ such that

$$[\mathbf{A}_{L+1} | \mathbf{E}(\mathsf{ID}) | \mathbf{F}(\mathsf{t})] \mathbf{g}_{\mathsf{ID},\mathsf{t}} = \mathbf{u}_{\ell}$$

by running $\mathsf{SampleLeft}(\cdot)$ with trapdoor $\mathbf{T}_{[\mathbf{A}_{L+1}|\mathbf{E}(\mathsf{ID})]}$ and Gaussian parameter σ_{ℓ}. Finally, in case $\ell \geq 2$, it extracts $(\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell})_{i \in [\ell-1]}$ from $\mathsf{ku}_{\mathsf{pa(ID)},\mathsf{t}}$ and outputs

$$\mathsf{dk}_{\mathsf{ID},\mathsf{t}} = (\mathbf{d}_{\mathsf{ID},\mathsf{t}}, (\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell})_{i \in [\ell-1]}, \mathbf{g}_{\mathsf{ID},\mathsf{t}}).$$

Otherwise, in case $\ell = 1$, it simply outputs

$$\mathsf{dk}_{\mathsf{ID},\mathsf{t}} = (\mathbf{d}_{\mathsf{ID},\mathsf{t}}, \mathbf{g}_{\mathsf{ID},\mathsf{t}}).$$

$\mathsf{Decrypt}(\mathsf{PP}, \mathsf{dk}_{\mathsf{ID},\mathsf{t}}, \mathsf{ct}) \to \mathsf{M}$: The decryption algorithm is run by user ID at level ℓ, where $1 \leq \ell \leq L$. It first parses the ciphertext ct as $(c_0, \mathbf{c}_1, \cdots, \mathbf{c}_{\ell}, \mathbf{c}_{L+1})$. Then, in case $\ell \geq 2$, it uses its decryption key $\mathsf{dk}_{\mathsf{ID},\mathsf{t}} = (\mathbf{d}_{\mathsf{ID},\mathsf{t}}, (\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell})_{i \in [\ell-1]}, \mathbf{g}_{\mathsf{ID},\mathsf{t}})$ and computes

$$c' = c_0 - \sum_{i=1}^{\ell-1} \mathbf{f}^{\top}_{\mathsf{ID}_{[i]},\mathsf{t},\ell} \mathbf{c}_i - \mathbf{d}^{\top}_{\mathsf{ID},\mathsf{t}} \mathbf{c}_{\ell} - \mathbf{g}^{\top}_{\mathsf{ID},\mathsf{t}} \mathbf{c}_{L+1} \in \mathbb{Z}_q. \tag{4}$$

Otherwise, in case $\ell = 1$, it uses its decryption key $\mathsf{dk}_{\mathsf{ID},\mathsf{t}} = (\mathbf{d}_{\mathsf{ID},\mathsf{t}}, \mathbf{g}_{\mathsf{ID},\mathsf{t}})$ and computes

$$c' = c_0 - \mathbf{d}^{\top}_{\mathsf{ID},\mathsf{t}} \mathbf{c}_1 - \mathbf{g}^{\top}_{\mathsf{ID},\mathsf{t}} \mathbf{c}_{L+1} \in \mathbb{Z}_q.$$

Finally, it compares c' and $\lfloor \frac{q}{2} \rfloor$ treating them as integers in $\mathbb{Z}$, and outputs 1 in case $|c' - \lfloor \frac{q}{2} \rfloor| < \lfloor \frac{q}{4} \rfloor$ and 0 otherwise.

Correctness. Here, we show that our RHIBE scheme is correct with overwhelming probability.

Lemma 11. *Assume $O((\alpha + mL^2\sigma_L\alpha')q) \leq q/5$ holds with with overwhelming probability. Then the above scheme has negligible decryption error.*

Proof. We consider a user ID at level ℓ for $\ell \in [L]$ that decrypts a ciphertext created on time t. To show correctness, we only need to consider the case where ID and all of its ancestors are not revoked. In other words, ID obtains the key update information $\mathsf{ku}_{\mathsf{pa(ID)},\mathsf{t}}$ from his parent. Below, we only show the case for $\ell \geq 2$, since the case for $\ell = 1$ is a special case of $\ell \geq 2$, where the vectors $\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell}$ are not required for decryption. Now, since ID is not revoked (by his parent), there exists at least one node θ such that $\theta \in \mathsf{Path}(\mathsf{BT}_{\mathsf{pa(ID)}}, \eta_{\mathsf{ID}}) \cap \mathsf{KUNode}(\mathsf{BT}_{\mathsf{pa(ID)}}, \mathsf{RL}_{\mathsf{pa(ID)},\mathsf{t}})$. Furthermore, the key update information $\mathsf{ku}_{\mathsf{pa(ID)},\mathsf{t}}$ includes $(\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell})_{i \in [\ell-1]}$, i.e., "partial" information of the all his ancestor's decryption keys.

We explain the decryption procedure of Eq. (4) one step at a time. Recall that the decryption key is created during GenDK, and is of the form $\mathsf{dk}_{\mathsf{ID},\mathsf{t}} = (\mathbf{d}_{\mathsf{ID},\mathsf{t}}, (\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell})_{i\in[\ell-1]}, \mathbf{g}_{\mathsf{ID},\mathsf{t}})$. First, the vector $\mathbf{d}_{\mathsf{ID},\mathsf{t}} \in \mathbb{Z}_q^{(\ell+2)m}$ can be rewritten as $[\mathbf{e}^{\mathsf{L}}_{\mathsf{ID},\theta} + \mathbf{e}^{\mathsf{L}}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t},\theta} \| \mathbf{e}^{\mathsf{R}}_{\mathsf{ID},\theta} \| \mathbf{e}^{\mathsf{R}}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t},\theta}]$, where

$$\begin{aligned} &[\mathbf{A}_\ell|\mathbf{E}(\mathsf{ID})]\mathbf{e}_{\mathsf{ID},\theta} = \mathbf{u}_{\mathsf{pa}(\mathsf{ID}),\theta}, && [\mathbf{A}_\ell|\mathbf{E}(\mathsf{pa}(\mathsf{ID}))|\mathbf{F}(\mathsf{t})]\mathbf{e}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t},\theta} = \mathbf{u}_\ell - \mathbf{u}_{\mathsf{pa}(\mathsf{ID}),\theta}, \\ &\mathbf{e}_{\mathsf{ID},\theta} = [\mathbf{e}^{\mathsf{L}}_{\mathsf{ID},\theta} \| \mathbf{e}^{\mathsf{R}}_{\mathsf{ID},\theta}], && \mathbf{e}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t},\theta} = [\mathbf{e}^{\mathsf{L}}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t},\theta} \| \mathbf{e}^{\mathsf{R}}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t},\theta}]. \end{aligned}$$

Therefore, we have $[\mathbf{A}_\ell|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]\mathbf{d}_{\mathsf{ID},\mathsf{t}} = \mathbf{u}_\ell$. Next, for each $i \in [\ell-1]$, the vector $\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell} \in \mathbb{Z}_q^{(i+2)m}$ can be rewritten as $\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell} = \mathbf{d}_{\mathsf{ID}_{[i]},\mathsf{t}} + [\mathbf{f}_{\mathsf{ID}_{[i]},\ell} \| \mathbf{0}_{m\times 1}]$, where we have

$$[\mathbf{A}_i|\mathbf{E}(\mathsf{ID}_{[i]})|\mathbf{F}(\mathsf{t})]\mathbf{d}_{\mathsf{ID}_{[i]},\mathsf{t}} = \mathbf{u}_i, \quad [\mathbf{A}_i|\mathbf{E}(\mathsf{ID}_{[i]})]\mathbf{f}_{\mathsf{ID}_{[i]},\ell} = \mathbf{u}_\ell - \mathbf{u}_i.$$

Here, the first equation follows from the exact same argument we made above for the vector $\mathbf{d}_{\mathsf{ID},\mathsf{t}}$. Therefore, combining the two, we have $[\mathbf{A}_i|\mathbf{E}(\mathsf{ID}_{[i]})|\mathbf{F}(\mathsf{t})]\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell} = \mathbf{u}_\ell$. Finally, the vector $\mathbf{g}_{\mathsf{ID},\mathsf{t}} \in \mathbb{Z}_q^{(\ell+2)m}$ satisfies $[\mathbf{A}_{L+1}|\mathbf{E}(\mathsf{ID})|\mathbf{F}(\mathsf{t})]\mathbf{g}_{\mathsf{ID},\mathsf{t}} = \mathbf{u}_\ell$. Combining everything together, we have the following for $i \in [\ell-1]$:

$$\mathbf{f}^\top_{\mathsf{ID}_{[i]},\mathsf{t},\ell}\mathbf{c}_i = \mathbf{u}_\ell^\top \mathbf{s}_i + \mathbf{f}^\top_{\mathsf{ID}_{[i]},\mathsf{t},\ell}\mathbf{x}_i, \quad \mathbf{d}^\top_{\mathsf{ID},\mathsf{t}}\mathbf{c}_\ell = \mathbf{u}_\ell^\top \mathbf{s}_\ell + \mathbf{d}^\top_{\mathsf{ID},\mathsf{t}}\mathbf{x}_\ell, \quad \mathbf{g}^\top_{\mathsf{ID},\mathsf{t}}\mathbf{c}_{L+1} = \mathbf{u}_\ell^\top \mathbf{s}_{L+1} + \mathbf{g}^\top_{\mathsf{ID},\mathsf{t}}\mathbf{x}_{L+1}.$$

Therefore,

$$\begin{aligned} c' &= \mathbf{u}_\ell^\top(\mathbf{s}_1 + \cdots + \mathbf{s}_\ell + \mathbf{s}_{L+1}) + x + \mathsf{M}\left\lfloor\frac{q}{2}\right\rfloor - \sum_{i=1}^{\ell-1}\mathbf{f}^\top_{\mathsf{ID}_{[i]},\mathsf{t},\ell}\mathbf{c}_i - \mathbf{d}^\top_{\mathsf{ID},\mathsf{t}}\mathbf{c}_\ell - \mathbf{g}^\top_{\mathsf{ID},\mathsf{t}}\mathbf{c}_{L+1} \\ &= \mathsf{M}\left\lfloor\frac{q}{2}\right\rfloor + \underbrace{x - \sum_{i=1}^{\ell-1}\mathbf{f}^\top_{\mathsf{ID}_{[i]},\mathsf{t},\ell}\mathbf{x}_i - \mathbf{d}^\top_{\mathsf{ID},\mathsf{t}}\mathbf{x}_\ell - \mathbf{g}^\top_{\mathsf{ID},\mathsf{t}}\mathbf{x}_{L+1}}_{:=\mathbf{z}\ (\text{“noise”})}. \end{aligned}$$

Here the noise can be bounded as follows with overwhelming probability due to Lemma 1:

$$\begin{aligned} \|\mathbf{z}\|_2 &\le |x| + \sum_{i=1}^{\ell-1}\|\mathbf{f}_{\mathsf{ID}_{[i]},\mathsf{t},\ell}\|_2 \cdot \|\mathbf{x}_i\|_2 + \|\mathbf{d}_{\mathsf{ID},\mathsf{t}}\|_2 \cdot \|\mathbf{x}_\ell\|_2 + \|\mathbf{g}_{\mathsf{ID},\mathsf{t}}\|_2 \cdot \|\mathbf{x}_{L+1}\|_2 \\ &\le \alpha q + \big(\sum_{i=1}^{\ell-1} 3\sigma_i(i+2) + 3\sigma_\ell(\ell+2)\big) \cdot m\alpha' q \\ &\le (\alpha + 3m(\ell+1)(\ell+2)\sigma_\ell\alpha')q \end{aligned}$$

Then, since $\ell \le L$ and $\sigma_i \le \sigma_L$ for all $i \in [L]$, the error is upper bounded by $O((\alpha + mL^2\sigma_L\alpha')q)$ with all but negligible probability. By assumption this is smaller than $q/5$ with overwhelming probability. Hence, the error probability for the $\mathsf{Decrypt}$ algorithm is negligible. □

Parameter Selection. Here, we provide an example parameter selection of our scheme. First recall the following restrictions we have on the parameters:

- the error term is less than $q/5$ with high probability (i.e., $O((\alpha + mL^2\sigma_L\alpha')q) < 5/q$. See Lemma 11),
- algorithm $\mathsf{TrapGen}$ works as specified (i.e., $m \ge 2n\lceil\log q\rceil$. See Lemma 3),

- algorithm SampleLeft and ExtRndLeft work as specified in the main construction for each level $0 \leq \ell \leq L$ (i.e., $\sigma_0 \geq \|\mathbf{T}_{\mathbf{A}_i}\|_{\mathrm{GS}} \cdot \omega(\sqrt{\log m})$ for $i \in [L+1]$ and $\sigma_{i+1} \geq \sigma_i \sqrt{m} \cdot \omega(\sqrt{\log m})$ for $i \in [L-1]$. See Lemma 3, 4),
- algorithm ExtRndRight works as specified in the security proof (i.e., $\sigma_0 \geq \|\mathbf{R}_i\|_2 \cdot \|\mathbf{T}_{\mathbf{G}}\| \cdot \omega(\sqrt{\log n})$. See Lemma 4, 6),
- algorithm ReRand works as specified in the security proof (i.e., $\alpha'/2\alpha > \|\mathbf{R}^*\|_2$ where $\mathbf{R}^* \leftarrow \{-1, 1\}^{m \times (L+1)m}$, $\alpha q > \omega(\sqrt{(L+1)m})$. See Lemma 7),
- the hardness assumption of LWE applies, i.e., $q > 2\sqrt{n}/\alpha$.

To satisfy the above requirements, one way to set the parameters is as follows, where L denotes the maximum depth of the hierarchy.

$$
\begin{aligned}
&m = O(n \log q), && q = m^{\frac{L+6}{2}} L^{\frac{5}{2}} \omega((\log n)^{\frac{L+1}{2}}), \\
&\sigma_i = m^{\frac{i+1}{2}} \omega((\log n)^{\frac{i+1}{2}}) && \alpha = m^{-\frac{L+4}{2}} L^{-\frac{5}{2}} \omega((\log n)^{\frac{L+1}{2}})^{-1}, \qquad \alpha' = (mL)^{\frac{1}{2}} \alpha,
\end{aligned}
$$

where $i \in [0, L]$ and we round up q to the nearest larger prime.

Remarks. Note that for simplicity we defined correctness of RHIBE to hold with probability one in Section 4. Therefore, to be consistent with our definition, we can use standard techniques to modify our lattice-based construction to have no decryption error by restricting a bound on the secret/noise vectors. In particular, in the rare case when we sample a long vector, we simply resample until we obtain a short vector. By setting the bound appropriately, we can always sample such a short vector in expected polynomial time; we would only require to sample a constant number of times, with all but negligible probability,

6.2 Security

Theorem 2. *The above RHIBE scheme* Π *is selective-identity secure assuming the hardness of the* $\mathsf{LWE}_{n,m+1,q,\chi}$ *problem, where* $\chi = D_{\mathbb{Z}^{m+1},\alpha q}$.

Proof. Let $\mathcal{A}$ be a PPT adversary that attacks the selective-identity security of the RHIBE scheme Π with advantage $\mathsf{Adv}^{\mathtt{RHIBE\text{-}sel}}_{\Pi,L,\mathcal{A}}(\lambda) = \epsilon$. In addition, let $(\mathsf{ID}^* = (\mathsf{id}^*_1, \ldots, \mathsf{id}^*_{\ell^*}), \mathsf{t}^*)$ be the challenge identity/time period pair that $\mathcal{A}$ sends to the challenger at the beginning of the game. Now, observe that the strategy taken by $\mathcal{A}$ can be divided into the following two types that are mutually exclusive, where the first type can be further divided into ℓ types of strategies that are mutually exclusive:

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

Since all the above strategies fulfill the conditions stated in Lemma 8, we can assume without loss of generality that $\mathcal{A}$ is an adversary that always follows one of the above strategies (which has advantage ϵ). Below we provide two types of security proofs: one for when $\mathcal{A}$ uses the Type-I-i^* ($1 \leq i^* \leq \ell^*$) strategy and another for when $\mathcal{A}$ uses the Type-II strategy. In both proofs, we show security of the scheme through a sequence of games, where we define E_i to be the event that $\mathcal{A}$ guesses correctly the bit chosen by the challenger in Game_i. In particular, regardless of the strategy taken by $\mathcal{A}$, both proofs share a common game sequence $\mathsf{Game}_{\mathsf{real}}$ and Game_0 as defined below:

$\mathsf{Game}_{\mathsf{real}}$: This is the real security game between the adversary $\mathcal{A}$ and a challenger, where $\mathcal{A}$ sends the challenge tuple $(\mathsf{ID}^* = (\mathsf{id}_1^*, \ldots, \mathsf{id}_{\ell^*}^*), \mathsf{t}^*)$ to the challenger at the beginning of the game. By definition, we have

$$\mathsf{Adv}_{\Pi,L,\mathcal{A}}^{\texttt{RHIBE-sel}}(\lambda) = 2 \cdot \left| \Pr[\mathsf{E}_{\mathsf{real}}] - \frac{1}{2} \right| \quad \Leftrightarrow \quad \Pr[\mathsf{E}_{\mathsf{real}}] = \frac{1}{2}(1 \pm \epsilon).$$

Game_0: In this game, we make a conceptual change on how the challenger deals with the trapdoors in $\mathsf{GenSK}, \mathsf{KeyUp}$ and GenDK, so that we only need to keep in mind during the following game sequence whether the challenger is in possession of the "base" trapdoors $(\mathbf{T}_{\mathbf{A}_i})_{i\in[L+1]}$ provided in the master secret key $\mathsf{sk}_{\mathsf{kgc}}$. In particular, in this game whenever the challenger requires to use a trapdoor to sample a short vector, say run algorithm $\mathsf{SampleLeft}$ with trapdoor $\mathbf{T}_{[\mathbf{A}_\ell|\mathbf{E}(\mathsf{pa}(\mathsf{ID}))]}$ during GenSK, he creates the required trapdoor from the base trapdoor $\mathbf{T}_{\mathbf{A}_i}$ provided in $\mathsf{sk}_{\mathsf{kgc}}$ by running algorithm $\mathsf{ExtRndLeft}$. Furthermore, whenever the challenger is required to extend a trapdoor basis, say extend $\mathbf{T}_{[\mathbf{A}_i|\mathbf{E}(\mathsf{pa}(\mathsf{ID}))]}$ to $\mathbf{T}_{[\mathbf{A}_i|\mathbf{E}(\mathsf{ID})]}$ during GenSK, the challenger extends it from the base trapdoor $\mathbf{T}_{\mathbf{A}_i}$ provided in $\mathsf{sk}_{\mathsf{kgc}}$, e.g., extend $\mathbf{T}_{\mathbf{A}_i}$ to $\mathbf{T}_{[\mathbf{A}_i|\mathbf{E}(\mathsf{ID})]}$. In both cases, the Gaussian parameters are set accordingly so that the quality of the extended trapdoors are consistent with the actual trapdoor. Then, due to Lemma 3 and 4, since the sampled vectors and the extended trapdoors are statistically independent from the trapdoors provided as input, we have $|\Pr[\mathsf{E}_{\mathsf{real}}] - \Pr[\mathsf{E}_0]| = \mathsf{negl}(\lambda)$.

In the following, we prove that we have $\Pr[\mathsf{E}_0] = \frac{1}{2} \pm \mathsf{negl}(\lambda)$, regardless of the strategy taken by the adversary $\mathcal{A}$. From the above argument, this implies that $\epsilon = \mathsf{negl}(\lambda)$, which concludes the proof of our theorem. Below, we first provide the proof against an adversary $\mathcal{A}$ that uses the Type-I-i^* strategy.

Lemma 12. *The advantage of an adversary $\mathcal{A}$ using the Type-I-i^* strategy in Game_0 is negligible assuming the hardness of the $\mathsf{LWE}_{n,m+1,q,\chi}$ problem, where $\chi = D_{\mathbb{Z}^{m+1},\alpha q}$.*

Proof. Our goal of this proof is to modify the challenger so that he is able to simulate the game with only the trapdoors $\{\mathbf{T}_{\mathbf{A}_i}\}_{i\in[L+1]\backslash\{i^*\}}$. At a high level, since the challenger will not require $\mathbf{T}_{\mathbf{A}_{i^*}}$, this will allow us to embed the matrix $\mathbf{A}_{i^*}$ given as the LWE problem in the public parameter PP. To this end, we informally illustrate in Table 1 for reference the situations for which the actual challenger in Game_0 requires to use the trapdoor $\mathbf{T}_{\mathbf{A}_{i^*}}$, either implicitly or explicitly, to respond to $\mathcal{A}$'s queries. Note that we do not include a row corresponding to the secret key reveal query, since the challenger simply returns the secret key created during the secret key generation query. Furthermore, we emphasize that we do not explicitly consider the key updates $\mathsf{ku}_{\mathsf{ID},\mathsf{t}}$ created during the secret key generation query since this will be captured by item (iii) in our proof below without loss of generality. (See $\mathsf{Game}_{\text{I-}i^*\text{-}3}$ for further details.) Here, the unnumbered items concerning users $\mathsf{ID} \in (\mathcal{ID})^{i^*}$ in the above table are constructed deterministically from items (i), (ii) and (iii): to answer revoke & key update queries, the challenger creates $(\mathbf{f}_{\mathsf{ID},\mathsf{t},k})_{k\in[i^*+1,L]}$ from combining $(\mathbf{f}_{\mathsf{ID},k})_{k\in[i^*+1,L]}$ and $\mathbf{d}_{\mathsf{ID},\mathsf{t}}$, and to answer decryption key reveal queries, the challenger creates $\mathbf{d}_{\mathsf{ID},\mathsf{t}}$ from combining $\mathbf{e}_{\mathsf{ID},\theta}$ and $\mathbf{e}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t},\theta}$. Note that $\mathbf{e}_{\mathsf{pa}(\mathsf{ID}),\mathsf{t},\theta}$ corresponds to item (iii), since $\mathsf{pa}(\mathsf{ID}) \in (\mathcal{ID})^{i^*-1}$. Therefore, in the following we only focus on how to simulate items (i), (ii) and (iii), ultimately without requiring $\mathbf{T}_{\mathbf{A}_{i^*}}$. We now proceed with the following sequence of games.

$\mathsf{Game}_{\text{I-}i^*\text{-}1}$: In this game, we change the way $(\mathbf{B}_j)_{j\in[L+1]}$ are chosen. At the beginning of the game, the $\mathsf{Game}_{\text{I-}i^*\text{-}1}$ challenger samples $\mathbf{R}_j^* \leftarrow \{-1,1\}^{m\times m}$ for $j \in [L+1]$ and sets $(\mathbf{B}_j)_{j\in[L+1]}$ as

	$\mathsf{ID} \in (\mathcal{ID})^{i^*}$	$\mathsf{ID} \in (\mathcal{ID})^{i^*-1}$	(In case $i^* \geq 3$) $\mathsf{ID} \in (\mathcal{ID})^{\leq i^*-2}$
Secret Key Generation $(\mathsf{sk}_{\mathsf{ID}})$	(i) $(\mathbf{e}_{\mathsf{ID},\theta})_{\theta \in \mathsf{Path}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \eta_{\mathsf{ID}})}$ $(\mathbf{f}_{\mathsf{ID},k})_{k \in [i^*+1, L]}$	(ii) $\mathbf{T}_{[\mathbf{A}_{i^*} \vert \mathbf{E}(\mathsf{ID})]}$	(ii) $\mathbf{T}_{[\mathbf{A}_{i^*} \vert \mathbf{E}(\mathsf{ID})]}$
Revoke & Key Update $(\mathsf{ku}_{\mathsf{ID},\mathsf{t}})$	$(\mathbf{f}_{\mathsf{ID},\mathsf{t},k})_{k \in [i^*+1, L]}$	(iii) $(\mathbf{e}_{\mathsf{ID},\mathsf{t},\theta})_{\theta \in \mathsf{KUNode}(\mathtt{BT}_{\mathsf{ID}}, \mathsf{RL}_{\mathsf{ID},\mathsf{t}})}$	–
Decryption Key Reveal $(\mathsf{dk}_{\mathsf{ID},\mathsf{t}})$	$\mathbf{d}_{\mathsf{ID},\mathsf{t}}$	–	–

Table 1: Items for which the challenger requires $\mathbf{T}_{\mathbf{A}_{i^*}}$ to construct.

follows:

$$\mathbf{B}_j = \begin{cases} \mathbf{A}_{i^*}\mathbf{R}_j^* - H(\mathsf{id}_j^*)\mathbf{G}, & \text{for } j \in [i^*], \\ \mathbf{A}_{i^*}\mathbf{R}_j^* & \text{for } j \in [i^*+1, L], \\ \mathbf{A}_{i^*}\mathbf{R}_j^* - H(\mathsf{t}^*)\mathbf{G}, & \text{for } j = L+1. \end{cases}$$

The challenger keeps the matrices $(\mathbf{R}_j^*)_{j \in [L+1]}$ as a part of $\mathsf{sk}_{\mathsf{kgc}}$. By Lemma 5, the statistical distance between the public parameters PP in Game_0 and $\mathsf{Game}_{\text{I-}i^*\text{-1}}$ is negligible. Therefore, we have $|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_{\text{I-}i^*\text{-1}}]| = \mathsf{negl}(\lambda)$.

$\mathsf{Game}_{\text{I-}i^*\text{-2}}$: In this game, we make two modifications: when we generate the binary tree $\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}$ and how we assign $\mathsf{ID}^*_{[i^*]}$ to the binary tree $\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}$. Recall in the previous game, the challenger created $\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}$ when $\mathcal{A}$ submitted a secret key generation query on $\mathsf{pa}(\mathsf{ID}^*_{[i^*]})$, and assigned $\mathsf{ID}^*_{[i^*]}$ to some random leaf $\eta_{\mathsf{ID}^*_{[i^*]}}$ of $\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}$ when $\mathcal{A}$ submitted a secret key generation query on $\mathsf{ID}^*_{[i^*]}$. In this game, the $\mathsf{Game}_{\text{I-}i^*\text{-2}}$ challenger creates an empty binary tree $\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}$ and chooses a random leaf $\eta_{\mathsf{ID}^*_{[i^*]}}$ in $\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}$ for which he plans to assign $\mathsf{ID}^*_{[i^*]}$ before providing $\mathcal{A}$ the public parameter PP. Then, when $\mathcal{A}$ issues a secret key generation query on some $\mathsf{ID} \in \mathsf{pa}(\mathsf{ID}^*_{[i^*]}) \| \mathcal{ID}$, if $\mathsf{ID} = \mathsf{ID}^*_{[i^*]}$ then the challenger proceeds with GenSK as if $(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID})}, \eta_{\mathsf{ID}^*_{[i^*]}}) \leftarrow \mathsf{CS.Assign}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}, \mathsf{ID}^*_{[i^*]})$, and otherwise it assigns ID to some random leaf of $\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}$ that is not $\eta_{\mathsf{ID}^*_{[i^*]}}$. Note that this can be done, since $\mathcal{A}$ sends the challenger the challenge identity ID^* at the outset of the game. Since the time on which $\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}$ is generated is only a conceptual matter and the random assignment of $\mathsf{ID}^*_{[i^*]}$ made by the challenger is statistically hidden from $\mathcal{A}$, the view of the adversary is unchanged. Therefore, we have $|\Pr[\mathsf{E}_{\text{I-}i^*\text{-1}}] - \Pr[\mathsf{E}_{\text{I-}i^*\text{-2}}]| = 0$.

$\mathsf{Game}_{\text{I-}i^*\text{-3}}$: In this game, we change the challenger so he does *not* have to use the trapdoor $\mathbf{T}_{\mathbf{A}_{i^*}}$ when generating the following short vectors for user $\mathsf{ID}^*_{[i^*]}$: $\mathbf{e}_{\mathsf{ID}^*_{[i^*]},\theta}$ for $\theta \in \mathsf{Path}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}, \eta_{\mathsf{ID}^*_{[i^*]}})$ in $\mathsf{sk}_{\mathsf{ID}^*_{[i^*]}}$ (Table 1, Item (i)) and $\mathbf{e}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\mathsf{t}^*,\theta}$ for $\theta \in \mathsf{KUNode}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}, \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\mathsf{t}^*})$ in $\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\mathsf{t}^*}$ (Table 1, Item (iii)). To this end, we modify when and how the vectors $\mathbf{u}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\theta}$ stored in each node $\theta \in \mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}$ in $\mathsf{sk}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}$ are constructed. In the following, let $S_{\mathsf{Path}} = \mathsf{Path}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}, \eta_{\mathsf{ID}^*_{[i^*]}})$ and $S_{\mathsf{KU},\mathsf{t}^*} = \mathsf{KUNode}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}, \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\mathsf{t}^*})$. By definition of the Type-I-i^* strategy, user $\mathsf{ID}^*_{[i^*]}$ must be revoked before time period t^* for the adversary to win. Therefore, due to the property of the CS scheme, we have $S_{\mathsf{Path}} \cap S_{\mathsf{KU},\mathsf{t}^*} = \emptyset$.

We first recall when and how the vectors $\mathbf{u}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\theta}$ stored in the nodes $\theta \in \mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}$ are constructed. In the beginning, the binary tree $\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}$ is initialized empty. The only situation the challenger updates $\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}$ is when $\mathcal{A}$ issues a secret key generation query for some $\mathsf{ID} \in \mathsf{pa}(\mathsf{ID}^*_{[i^*]}) \| \mathcal{ID}$ or a revoke & key update query, and the relevant nodes $\theta \in \mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}$ for answering theses queries have not been stored any vectors yet. For these particular nodes, the $\mathsf{Game}_{\text{I-}i^*\text{-}2}$ challenger samples a random vector $\mathbf{u}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\theta}$ and updates the binary tree $\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}$ by storing the vectors inside node θ. Note that when $\mathcal{A}$ issues a secret key generation query on $\mathsf{ID} \in \mathsf{pa}(\mathsf{ID}^*_{[i^*]})\|\mathcal{ID}$ (resp. a revoke & key update query[16]), the challenger samples short vectors $\mathbf{e}_{\mathsf{ID},\theta}$ for $\theta \in \mathsf{Path}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}, \eta_{\mathsf{ID}})$ in $\mathsf{sk}_{\mathsf{ID}}$ (resp. $\mathbf{e}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\mathsf{t_{cu}},\theta}$ for $\theta \in \mathsf{KUNode}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}, \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\mathsf{t_{cu}}})$ in $\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\mathsf{t_{cu}}}$) such that

$$[\mathbf{A}_{i^*}|\mathbf{E}(\mathsf{ID})]\,\mathbf{e}_{\mathsf{ID},\theta} = \mathbf{u}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\theta} \quad \text{for} \quad \theta \in \mathsf{Path}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}, \eta_{\mathsf{ID}}), \tag{5}$$

$$\begin{aligned}[\mathbf{A}_{i^*}|\mathbf{E}(\mathsf{pa}(\mathsf{ID}^*_{[i^*]}))|\mathbf{F}(\mathsf{t_{cu}})]\mathbf{e}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\mathsf{t_{cu}},\theta} &= \mathbf{u}_{i^*} - \mathbf{u}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\theta} \\ &\text{for} \quad \theta \in \mathsf{KUNode}(\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}, \mathsf{RL}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\mathsf{t_{cu}}}),\end{aligned} \tag{6}$$

where the required trapdoors for these operations are created by the challenger from $\mathbf{T}_{\mathbf{A}_i^*}$ on the fly due to the modification we made in Game_0. Note that to be precise, we must also take into account the fact that we run the KeyUp algorithm during the secret key generation query. However, we omit this for clarity, since it can be seen that the we can make the same argument as above for the key updates generated during the secret key generation query.

In this game, whenever $\mathcal{A}$ issues a secret key generation query for some $\mathsf{ID} \in \mathsf{pa}(\mathsf{ID}^*_{[i^*]})\|\mathcal{ID}$ or a revoke & key update query, the $\mathsf{Game}_{\text{I-}i^*\text{-}3}$ challenger first checks whether the node $\theta \in S_{\mathsf{undef}}$ is in S_{Path} or not, where S_{undef} denotes the set of nodes in $\mathtt{BT}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]})}$ where a vector $\mathbf{u}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\theta}$ has not been not stored yet and for which it must be defined for the challenger to answer $\mathcal{A}$'s query. If $\theta \in S_{\mathsf{undef}} \cap S_{\mathsf{Path}}$, the challenger first samples a vector $\mathbf{e}_{\mathsf{ID}^*_{[i^*]},\theta} \leftarrow D_{\mathbb{Z}^{(i^*+1)m},\sigma_{i^*}}$ and sets $\mathbf{u}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\theta}$ as in Eq. (5). Then it stores $\mathbf{u}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\theta}$ in the node θ and keeps $\mathbf{e}_{\mathsf{ID}^*_{[i^*]},\theta}$ secret. If $\theta \in S_{\mathsf{undef}} \setminus (S_{\mathsf{undef}} \cap S_{\mathsf{Path}})$, the challenger first samples a vector $\mathbf{e}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\mathsf{t}^*,\theta} \leftarrow D_{\mathbb{Z}^{(i^*+1)m},\sigma_{i^*}}$ and sets $\mathbf{u}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\theta}$ as in Eq. (6) by implicitly setting $\mathsf{t_{cu}} = \mathsf{t}^*$. Specifically, it sets

$$\mathbf{u}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\theta} = -[\mathbf{A}_{i^*}|\mathbf{E}(\mathsf{pa}(\mathsf{ID}^*_{[i^*]}))|\mathbf{F}(\mathsf{t}^*)]\mathbf{e}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\mathsf{t}^*,\theta} + \mathbf{u}_{i^*}.$$

Then it stores $\mathbf{u}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\theta}$ in the node θ and keeps $\mathbf{e}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\mathsf{t}^*,\theta}$ secret. Now if the $\mathsf{ID} \in \mathsf{pa}(\mathsf{ID}^*_{[i^*]})\|\mathcal{ID}$ issued by $\mathcal{A}$ as the secret key generation query is not $\mathsf{ID}^*_{[i^*]}$, then the $\mathsf{Game}_{\text{I-}i^*\text{-}3}$ challenger samples the short vectors $(\mathbf{e}_{\mathsf{ID},\theta})_\theta$ as in Eq. (5) using $\mathbf{T}_{\mathbf{A}_{i^*}}$. Otherwise, in case $\mathsf{ID} = \mathsf{ID}^*_{[i^*]}$, the challenger simply returns the vectors $(\mathbf{e}_{\mathsf{ID}^*_{[i^*]},\theta})_{\theta \in S_{\mathsf{Path}}}$ which he has already created *without* using $\mathbf{T}_{\mathbf{A}_{i^*}}$. Furthermore, if the global counter $\mathsf{t_{cu}}$ on which $\mathcal{A}$ queried the revoke & key update query is not t^*, then the $\mathsf{Game}_{\text{I-}i^*\text{-}3}$ challenger samples the short vectors $(\mathbf{e}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\mathsf{t_{cu}},\theta})_\theta$ as in Eq. (6) using $\mathbf{T}_{\mathbf{A}_{i^*}}$. Otherwise, in case $\mathsf{t_{cu}} = \mathsf{t}^*$, the challenger simply returns the vectors $(\mathbf{e}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\mathsf{t}^*,\theta})_{\theta \in S_{\mathsf{KU},\mathsf{t}^*}}$ which he has already created *without* using $\mathbf{T}_{\mathbf{A}_{i^*}}$. Note that this procedure is well-defined since $S_{\mathsf{Path}} \cap S_{\mathsf{KU},\mathsf{t}^*} = \emptyset$. Now, due to Lemma 2, the distribution of the short vectors provided to $\mathcal{A}$ are distributed statistically close to those of the previous game. Therefore, we have $|\Pr[\mathsf{E}_{\text{I-}i^*\text{-}2}] - \Pr[\mathsf{E}_{\text{I-}i^*\text{-}3}]| = \mathsf{negl}(\lambda)$.

$\mathsf{Game}_{\text{I-}i^*\text{-}4}$: In this game, we change the challenger so he does *not* have to use the trapdoor $\mathbf{T}_{\mathbf{A}_{i^*}}$ for user $\mathsf{ID}^*_{[i^*]}$ when generating the short vectors $(\mathbf{f}_{\mathsf{ID}^*_{[i^*]},k})_{k \in [i^*+1,L]}$ in $\mathsf{sk}_{\mathsf{ID}^*_{[i^*]}}$ (Table 1, Item (i)). In particular, with the change we made in the previous game, the challenger no longer requires

[16] Recall that by our security definition, there exists a global counter $\mathsf{t_{cu}}$ initialized to 1, which the adversary $\mathcal{A}$ can increment only by querying the revoke & key update query. Specifically, all items that are associated with the revoke & key update query are by definition associated with the variable $\mathsf{t_{cu}}$.

$\mathbf{T}_{\mathbf{A}_{i^*}}$ when issued a secret key generation query for $\mathsf{ID}^*_{[i^*]}$. To this end, we modify how we create the vectors $(\mathbf{u}_k)_{k\in[i^*,L]\setminus\{\ell^*\}}$ in PP.

Recall that in the previous game, the challenger sampled $(\mathbf{u}_k)_{k\in[L]}$ as uniformly random vectors in $\mathbb{Z}_q^n$ at the beginning of the game. Then, when $\mathcal{A}$ issued a secret key generation query on $\mathsf{ID}^*_{[i^*]}$, the challenger sampled short vectors $(\mathbf{f}_{\mathsf{ID}^*_{[i^*]},k})_{k\in[i^*+1,L]}$ such that

$$[\mathbf{A}_{i^*}|\mathbf{E}(\mathsf{ID}^*_{[i^*]})]\mathbf{f}_{\mathsf{ID}^*_{[i^*]},k} = \mathbf{u}_k - \mathbf{u}_{i^*}, \tag{7}$$

where the required trapdoor for sampling is created by the challenger from $\mathbf{T}_{\mathbf{A}^*_i}$ on the fly.

We first describe how the vectors $(\mathbf{u}_k)_{k\in[L]}$ in PP are created. In this game, the $\mathsf{Game}_{\text{I-}i^*\text{-}4}$ challenger first samples $(\mathbf{u}_k)_{k\in[i^*-1]\cup\{\ell^*\}}$ as uniformly random vectors in $\mathbb{Z}_q^n$ at the beginning of the game, as was done in the previous game. Next, the challenger computes $\mathbf{u}_{i^*}$ by first sampling $\mathbf{f}_{\mathsf{ID}^*_{[i^*]},\ell^*} \leftarrow D_{\mathbb{Z}^{(i^*+1)m},\sigma_{i^*}}$ and setting it to satisfy the following equation:

$$[\mathbf{A}_{i^*}|\mathbf{E}(\mathsf{ID}^*_{[i^*]})]\mathbf{f}_{\mathsf{ID}^*_{[i^*]},\ell^*} = \mathbf{u}_{\ell^*} - \mathbf{u}_{i^*}.$$

Then, it keeps the vector $\mathbf{f}_{\mathsf{ID}^*_{[i^*]},\ell^*}$ secret. Finally, the challenger computes the remaining $(\mathbf{u}_k)_{k\in[i^*+1,L]\setminus\{\ell^*\}}$ by first sampling $\mathbf{f}_{\mathsf{ID}^*_{[i^*]},k} \leftarrow D_{\mathbb{Z}^{(i^*+1)m},\sigma_{i^*}}$ for $k \in [i^*+1, L] \setminus \{\ell^*\}$ and setting the vectors $\mathbf{u}_k$ to satisfy Eq. (7). Then, it keeps the vectors $(\mathbf{f}_{\mathsf{ID}^*_{[i^*]},k})_{k\in[i^*+1,L]\setminus\{\ell^*\}}$ secret. All other terms in PP are constructed as in the previous game. In this game, the $\mathsf{Game}_{\text{I-}i^*\text{-}4}$ challenger answers all queries made by $\mathcal{A}$ as in the previous game, except for when $\mathcal{A}$ queries $\mathsf{ID}^*_{[i^*]}$ as the secret key generation query. For this specific case, the challenger simply returns the vectors $(\mathbf{f}_{\mathsf{ID}^*_{[i^*]},k})_{k\in[i^*+1,L]}$ which he has already created at the beginning of the game *without* using $\mathbf{T}_{\mathbf{A}_{i^*}}$. Due to Lemma 2, the distribution of the vectors provided to $\mathcal{A}$ are distributed statistically close to those of $\mathsf{Game}_{\text{I-}i^*\text{-}3}$. Therefore, we have $|\Pr[\mathsf{E}_{\text{I-}i^*\text{-}3}] - \Pr[\mathsf{E}_{\text{I-}i^*\text{-}4}]| = \mathsf{negl}(\lambda)$.

$\mathsf{Game}_{\text{I-}i^*\text{-}5}$: In this game, we change how $\mathbf{A}_{i^*}$ is sampled. Namely, in this game, we generate $\mathbf{A}_{i^*}$ as a random matrix in $\mathbb{Z}_q^{n\times m}$ instead of generating it with a trapdoor. By Lemma 3, this makes only negligible difference. Accordingly, we modify the challenger, so that he does not require the trapdoor $\mathbf{T}_{\mathbf{A}_{i^*}}$ to answer any of the queries made by $\mathcal{A}$. Recall that in the previous game, the challenger used $\mathbf{T}_{\mathbf{A}_{i^*}}$ to create the following items in Table 1:

(a) Item (i) for $\mathsf{ID} \in (\mathcal{ID})^{i^*} \setminus \{\mathsf{ID}^*_{[i^*]}\}$.

(b) Item (ii) for $\mathsf{ID} \in (\mathcal{ID})^{\leq i^*-1} \setminus \mathsf{prefix}(\mathsf{ID}^*_{[i^*-1]})$.

(c) Item (iii) for $(\mathsf{ID}, \mathsf{t}) \in (\mathcal{ID})^{i^*-1} \times \mathcal{T}$.

Note that we do not require $\mathbf{T}_{\mathbf{A}_{i^*}}$ anymore to create the secret key $\mathsf{sk}_{\mathsf{ID}^*_{[i^*]}}$ in item (a) due to the modification we made in $\mathsf{Game}_{\text{I-}i^*\text{-}3}$ and $\mathsf{Game}_{\text{I-}i^*\text{-}4}$.[17] Furthermore, we can add the restriction $\mathsf{ID} \notin \mathsf{prefix}(\mathsf{ID}^*_{[i^*-1]})$ in item (b) without loss of generality, since an adversary following the Type-I-i^* strategy never asks for a secret key reveal query for $\mathsf{ID} \in \mathsf{prefix}(\mathsf{ID}^*_{[i^*-1]})$ and due to the change we made in Game_0, i.e., we extend the basis from $\mathbf{T}_{\mathbf{A}_{i^*}}$ instead from $\mathsf{pa}(\mathsf{ID})$'s basis. Finally, recall that when creating the key update $\mathsf{ku}_{\mathsf{pa}(\mathsf{ID}^*_{[i^*]}),\mathsf{t}^*}$ in item (c), we do not require $\mathbf{T}_{\mathbf{A}_{i^*}}$ anymore to sample short vectors corresponding to the path of $\mathsf{ID}^*_{[i^*]}$, i.e., $S_{\mathsf{KU},\mathsf{t}^*}$ due to the modification we made in $\mathsf{Game}_{\text{I-}i^*\text{-}3}$.

We now show that the $\mathsf{Game}_{\text{I-}i^*\text{-}5}$ challenger no longer requires $\mathbf{T}_{\mathbf{A}_{i^*}}$ to construct items (a), (b) and (c), which follows simply from the change we made in $\mathsf{Game}_{\text{I-}i^*\text{-}1}$. In the following we

[17] Recall that we do not require $\mathbf{T}_{\mathbf{A}_{i^*}}$ to execute the KeyUp algorithm for the secret key generation query when $\mathsf{t}_{\mathsf{cu}} = \mathsf{t}^*$.

only show the case for item (a), since the other cases can be easily verified in a similar fashion. Now, if $\mathsf{ID} \in (\mathcal{ID})^{i^*} \setminus \{\mathsf{ID}^*_{[i^*]}\}$, then there must exist an index $j \in [i^*]$ such that $\mathsf{id}_j \neq \mathsf{id}^*_j$ where $\mathsf{id}_j, \mathsf{id}^*_j$ is the j-th element identity of $\mathsf{ID}, \mathsf{ID}^*_{[i^*]}$, respectively. Hence, $H(\mathsf{id}_j) \neq H(\mathsf{id}^*_j)$. Then, to create a trapdoor $\mathbf{T}_{[\mathbf{A}_{i^*}|\mathbf{E}(\mathsf{ID})]}$, the challenger first runs $\mathsf{ExtRndRight}(\mathbf{A}_{i^*}, \mathbf{G}, \mathbf{R}^*_j, \mathbf{T}_{\mathbf{G}}, \sigma_0)$ to create $\mathbf{T}_{[\mathbf{A}_{i^*}|\mathbf{A}_{i^*}\mathbf{R}^*_j + (H(\mathsf{id}_j) - H(\mathsf{id}^*_j))\mathbf{G}]}$. If $i^* = 1$, this is the desired trapdoor basis. Otherwise, using this basis, the challenger extends it to a basis $\mathbf{T}_{[\mathbf{A}_{i^*}|\mathbf{E}(\mathsf{ID})]}$ by running $\mathsf{ExtRndLeft}$, where the Gaussian parameter is set as σ_{i^*} so that the quality of the trapdoor is the same as in the previous game. Note that this can be done since we can rearrange the rows of the basis in an arbitrary manner. Finally, the challenger samples the short secret key vectors by running $\mathsf{SampleLeft}(\cdot)$ with trapdoor $\mathbf{T}_{[\mathbf{A}_{i^*}|\mathbf{E}(\mathsf{ID})]}$ and Gaussian parameter σ_{i^*}. This shows that the challenger is able to create the required trapdoor without using $\mathbf{T}_{\mathbf{A}_{i^*}}$. Due to Lemma 3 and 4, since the sampled vectors and the extended trapdoors are statistically independent from the trapdoors provided as input, this makes a negligible difference. Since, we can make a similar argument in the case for items (b) and (c) as well, we have $|\Pr[\mathsf{E}_{\text{I-}i^*\text{-}4}] - \Pr[\mathsf{E}_{\text{I-}i^*\text{-}5}]| = \mathsf{negl}(\lambda)$.

$\mathsf{Game}_{\text{I-}i^*\text{-}6}$: In this game, we change the way the challenge ciphertext is created. In this game, when the $\mathsf{Game}_{\text{I-}i^*\text{-}6}$ challenger is issued a challenge query on $(\mathsf{M}_0, \mathsf{M}_1)$ by $\mathcal{A}$, it first samples $\mathbf{s}_i \leftarrow \mathbb{Z}_q^n$ for $i \in [\ell^*] \cup \{L+1\}$, $x \leftarrow D_{\mathbb{Z},\alpha q}$, $\bar{\mathbf{x}} \leftarrow D_{\mathbb{Z}^m,\alpha q}$, $\mathbf{x}_i \leftarrow D_{\mathbb{Z}^{(i+2)m},\alpha' q}$ for $i \in [\ell^*] \setminus \{i^*\}$ and $\mathbf{x}_{L+1} \leftarrow D_{\mathbb{Z}^{(\ell^*+2)m},\alpha' q}$. Then it computes $v = \mathbf{u}_{\ell^*}^\top \mathbf{s}_{i^*} + x \in \mathbb{Z}_q$, $\mathbf{v} = \mathbf{A}_{i^*}^\top \mathbf{s}_{i^*} + \bar{\mathbf{x}} \in \mathbb{Z}_q^m$ and the following terms:

$$\begin{cases} c_0 = v + \mathbf{u}_{\ell^*}^\top \Big(\displaystyle\sum_{i \in [\ell^*] \cup \{L+1\} \setminus \{i^*\}} \mathbf{s}_i \Big) + \mathsf{M}_b \\ \mathbf{c}_i = [\mathbf{A}_i | \mathbf{E}(\mathsf{ID}^*_{[i]}) | \mathbf{F}(\mathsf{t}^*)]^\top \mathbf{s}_i + \mathbf{x}_i \quad \text{for} \quad i \in [\ell^*] \setminus \{i^*\} \\ \mathbf{c}_{L+1} = [\mathbf{A}_{L+1} | \mathbf{E}(\mathsf{ID}^*) | \mathbf{F}(\mathsf{t}^*)]^\top \mathbf{s}_{L+1} + \mathbf{x}_{L+1} \end{cases} \tag{8}$$

where b is the random bit chosen by the challenger. It then sets $\mathbf{R}^* = [\mathbf{R}^*_{1^*} | \cdots | \mathbf{R}^*_{i^*} | \mathbf{R}^*_{L+1}] \in \mathbb{Z}^{m \times (i^*+1)m}$ and runs

$$\mathsf{ReRand}\Big([\mathbf{I}_m | \mathbf{R}^*], \mathbf{v}, \alpha q, \frac{\alpha'}{2\alpha} \Big) \to \mathbf{c} \in \mathbb{Z}_q^{(i^*+2)m}$$

from Lemma 7, where $\mathbf{I}_m$ is the identity matrix with size m. Finally, it sets $\mathbf{c}_{i^*} = \mathbf{c}$ and outputs the challenge ciphertext as follows:

$$\mathsf{ct} = (c_0, \mathbf{c}_1, \ldots, \mathbf{c}_{\ell^*}, \mathbf{c}_{L+1}) \in \mathbb{Z}_q \times \mathbb{Z}_q^{3m} \times \cdots \times \mathbb{Z}_q^{(\ell^*+2)m} \times \mathbb{Z}_q^{(\ell^*+2)m}. \tag{9}$$

We claim that this change alters the view of $\mathcal{A}$ only negligibly, which follows from the noise rerandomization lemma (Lemma 7). In particular, we set $\mathbf{V} = [\mathbf{I}_m | \mathbf{R}^*]$, $\mathbf{b} = \mathbf{A}_{i^*}^\top \mathbf{s}_{i^*}$ and $\mathbf{x} = \bar{\mathbf{x}}$ in Lemma 7 to conclude that the obtained distribution $\mathbf{c}$ is negligibly close to the following:

$$\begin{aligned} \mathbf{c}^\top &= \mathbf{s}_{i^*}^\top \mathbf{A}_{i^*} [\mathbf{I}_m | \mathbf{R}^*] + \mathbf{x}'^\top \\ &= \mathbf{s}_{i^*}^\top [\mathbf{A}_{i^*} | \mathbf{B}_1 + H(\mathsf{id}^*_1)\mathbf{G} | \cdots | \mathbf{B}_{i^*} + H(\mathsf{id}^*_{i^*})\mathbf{G} | \mathbf{B}_{L+1} + H(\mathsf{t}^*)\mathbf{G}] + \mathbf{x}'^\top \\ &= \mathbf{s}_{i^*}^\top [\mathbf{A}_{i^*} | \mathbf{E}(\mathsf{ID}^*_{[i^*]}) | \mathbf{F}(\mathsf{t}^*)] + \mathbf{x}'^\top \in \mathbb{Z}_q^{(i^*+2)m} \end{aligned}$$

where $\mathbf{x}'$ is distributed statistically close to $D_{\mathbb{Z}^{(i^*+2)m},\alpha' q}$ due to our parameter selection. It can be seen that the challenge ciphertext in Eq. (9) is distributed statistically close to the previous game. Therefore, we have

$$|\Pr[\mathsf{E}_{\text{I-}i^*\text{-}5}] - \Pr[\mathsf{E}_{\text{I-}i^*\text{-}6}]| = \mathsf{negl}(\lambda).$$

$\mathsf{Game}_{\text{I-}i^*\text{-}7}$: In this game, we further change the way the challenge ciphertext is created. In particular, in this game, the $\mathsf{Game}_{\text{I-}i^*\text{-}7}$ challenger first samples $\mathbf{s}_i \leftarrow \mathbb{Z}_q^n$ for $i \in [\ell^*] \cup \{L+1\} \setminus \{i^*\}$, $w \leftarrow \mathbb{Z}_q$, $\mathbf{w} \leftarrow \mathbb{Z}_q^m$, $x \leftarrow D_{\mathbb{Z},\alpha q}$, $\bar{\mathbf{x}} \leftarrow D_{\mathbb{Z}^m,\alpha q}$, $\mathbf{x}_i \leftarrow D_{\mathbb{Z}^{(i+2)m},\alpha' q}$ for $i \in [\ell^*] \setminus \{i^*\}$ and $\mathbf{x}_{L+1} \leftarrow D_{\mathbb{Z}^{(\ell^*+2)m},\alpha' q}$. Then it computes $v = w + x \in \mathbb{Z}_q$, $\mathbf{v} = \mathbf{w} + \bar{\mathbf{x}} \in \mathbb{Z}_q^m$ and sets the remaining terms as in Eq. (8) of the previous game. Furthermore, it sets $\mathbf{R}^*$ and runs the ReRand algorithm as in $\mathsf{Game}_{\text{I-}i^*\text{-}6}$. Finally, it sets the challenge ciphertext as in Eq. (9). We claim that

$$|\Pr[\mathsf{E}_{\text{I-}i^*\text{-}6}] - \Pr[\mathsf{E}_{\text{I-}i^*\text{-}7}]| = \mathsf{negl}(\lambda),$$

assuming the hardness of the $\mathsf{LWE}_{n,m+1,q,\chi}$ problem. To this end, we use $\mathcal{A}$ to construct an LWE adversary $\mathcal{B}$ as follows:

$\mathcal{B}$ is given the problem instance of LWE as $(\mathbf{A}', \mathbf{v}' = \mathbf{w}' + \bar{\mathbf{z}}') \in \mathbb{Z}_q^{n \times (m+1)} \times \mathbb{Z}_q^{m+1}$ where $\bar{\mathbf{z}}' \leftarrow D_{\mathbb{Z}^{m+1},\alpha q}$. The task of $\mathcal{B}$ is to distinguish whether $\mathbf{w}' = \mathbf{A}'^{\top}\mathbf{s}$ for $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ or $\mathbf{w}' \leftarrow \mathbb{Z}_q^{m+1}$. In the following, let the first column of $\mathbf{A}'$ be $\mathbf{u} \in \mathbb{Z}_q^n$ and the remaining columns be $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. Further, let the first coefficient of $\mathbf{v}'$ be v and the remaining coefficients be $\mathbf{v} \in \mathbb{Z}_q^m$. Using these terms, $\mathcal{B}$ sets the public parameter PP. In particular, $\mathcal{B}$ sets $(\mathbf{A}_{i^*}, \mathbf{u}_{\ell^*}) = (\mathbf{A}, \mathbf{u})$ and proceeds the setup as the $\mathsf{Game}_{\text{I-}i^*\text{-}4}$ challenger. Furthermore, whenever $\mathcal{A}$ issues a query, $\mathcal{B}$ proceeds as the $\mathsf{Game}_{\text{I-}i^*\text{-}5}$ challenger, and answers them without the knowledge of $\mathbf{T}_{\mathbf{A}_{i^*}}$. Finally, to generate the challenge ciphertext, it first picks $b \leftarrow \{0,1\}$ and generates the challenge ciphertext as in Eq. (9) using $v, \mathbf{v}$, and returns it to $\mathcal{A}$. Note that all $\mathcal{B}$ needs to do to generate the ciphertext is to run the ReRand algorithm, which it can do without knowledge of the secret randomness $\mathbf{s}, \bar{\mathbf{z}}'$. Let b' be the output of $\mathcal{A}$. $\mathcal{B}$ outputs 1 if $b' = b$ and 0 otherwise. It can be seen that if $\mathbf{A}', \mathbf{v}'$ is a valid LWE sample (i.e., $\mathbf{w}' = \mathbf{A}'^{\top}\mathbf{s}$), the view of the adversary corresponds to $\mathsf{Game}_{\text{I-}i^*\text{-}6}$. Otherwise (i.e., $\mathbf{w}' \leftarrow \mathbb{Z}_q^{m+1}$), it corresponds to $\mathsf{Game}_{\text{I-}i^*\text{-}7}$. We therefore conclude that assuming the hardness of the $\mathsf{LWE}_{n,m+1,q,\chi}$ problem we have $|\Pr[\mathsf{E}_{\text{I-}i^*\text{-}6}] - \Pr[\mathsf{E}_{\text{I-}i^*\text{-}7}]| = \mathsf{negl}(\lambda)$.

Finally, since v is distributed uniformly at random over $\mathbb{Z}_q$ and independently of all other terms, the probability of adversary $\mathcal{A}$ guessing whether $b = 0$ or $b = 1$ is exactly $1/2$. In particular, we have

$$\Pr[\mathsf{E}_{\text{I-}i^*\text{-}7}] = \frac{1}{2}.$$

Combining everything together, we conclude that if the adversary $\mathcal{A}$ uses the Type-I-i^* strategy, then $\Pr[\mathsf{E}_0] = \frac{1}{2} \pm \mathsf{negl}(\lambda)$ assuming the hardness of $\mathsf{LWE}_{n,m+1,q,\chi}$ problem. □

Similarly, we provide the following lemma against an adversary $\mathcal{A}$ that uses the Type-II strategy. The proof proceeds closely to Lemma 12, where we gradually modify the game so that the challenger no longer requires $\mathbf{T}_{\mathbf{A}_{L+1}}$ in the final game.

Lemma 13. *The advantage of an adversary $\mathcal{A}$ using the Type-II strategy in Game_0 is negligible assuming the hardness of the $\mathsf{LWE}_{n,m+1,q,\chi}$ problem, where $\chi = D_{\mathbb{Z}^{m+1},\alpha q}$.*

Proof. The proof outline is essentially the same as Lemma 12 against the adversary using the Type-I strategy. The only difference is that in this proof, we aim at modifying the challenger so that he is able to simulate the game without using the trapdoor $\mathbf{T}_{\mathbf{A}_{L+1}}$. At a high level, since the adversary does not require $\mathbf{T}_{\mathbf{A}_{L+1}}$ anymore, we would be able to embed the matrix $\mathbf{A}_{L+1}$ provided as the LWE problem into the public parameter PP. To this end, we provide for reference the situations for which the challenger in Game_0 requires to use the trapdoor $\mathbf{T}_{\mathbf{A}_{L+1}}$ to respond to $\mathcal{A}$'s queries:

(i) Secret Key Generation Query ($\mathsf{sk}_{\mathsf{ID}}$): $\mathbf{T}_{[\mathbf{A}_{L+1}|\mathbf{E}(\mathsf{ID})]}$ for any $\mathsf{ID} \in (\mathcal{ID})^{\leq L} \setminus \mathsf{prefix}(\mathsf{ID}^*)$

(ii) Decryption Key Reveal Query ($\mathsf{dk}_{\mathsf{ID},\mathsf{t}}$): $\mathbf{g}_{\mathsf{ID},\mathsf{t}}$ for any $(\mathsf{ID}, \mathsf{t}) \in (\mathcal{ID})^{\leq L} \times \mathcal{T} \setminus \{(\mathsf{ID}^*, \mathsf{t}^*)\}$.

Here, the restriction on the users ID for item (i) follows from the Type-II strategy, where the adversary $\mathcal{A}$ does not issue any secret key reveal queries on users $\mathsf{ID} \in \mathsf{prefix}(\mathsf{ID}^*)$. Furthermore, note that the challenger can respond to all other queries made by $\mathcal{A}$ by using the trapdoors $(\mathbf{T}_{\mathbf{A}_i})_{i\in[L]}$. With this in mind, we proceed with the following sequence of games.

$\mathsf{Game}_{\text{II-1}}$: In this game, we change the way $(\mathbf{B}_j)_{j\in[L+1]}$ are chosen. At the beginning of the game, the $\mathsf{Game}_{\text{II-1}}$ challenger samples $\mathbf{R}_j^* \leftarrow \{-1, 1\}^{m\times m}$ for $j \in [L+1]$ and sets $(\mathbf{B}_j)_{j\in[L+1]}$ as follows:

$$\mathbf{B}_j = \begin{cases} \mathbf{A}_{L+1}\mathbf{R}_j^* - H(\mathsf{id}_j^*)\mathbf{G}, & \text{for } j \in [\ell^*], \\ \mathbf{A}_{L+1}\mathbf{R}_j^* & \text{for } j \in [\ell^*+1, L], \\ \mathbf{A}_{L+1}\mathbf{R}_j^* - H(\mathsf{t}^*)\mathbf{G}, & \text{for } j = L+1. \end{cases}$$

The challenger keeps the matrices $(\mathbf{R}_j^*)_{j\in[L+1]}$ as a part of $\mathsf{sk}_{\mathsf{kgc}}$. By Lemma 5, the statistical distance between the public parameter PP in Game_0 and $\mathsf{Game}_{\text{II-1}}$ is negligible. Therefore, we have

$$|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_{\text{II-1}}]| = \mathsf{negl}(\lambda).$$

$\mathsf{Game}_{\text{II-2}}$: In this game, we modify the challenger so he does *not* require the trapdoor $\mathbf{T}_{\mathbf{A}_{L+1}}$ when generating $\mathbf{g}_{\mathsf{ID}^*_{[i]},\mathsf{t}^*}$ for $i \in [\ell^* - 1]$ in $\mathsf{dk}_{\mathsf{ID},\mathsf{t}^*}$ (See Item (ii)), i.e., decryption keys for users $\mathsf{ID} \in \mathsf{prefix}(\mathsf{ID}^*) \setminus \{\mathsf{ID}^*\}$. Note that due to the definition of the security game, the challenger never has to create a decryption key for the user-time pair $(\mathsf{ID}^*, \mathsf{t}^*)$. To this end, we modify how we create the vectors $(\mathbf{u}_k)_{k\in[\ell^*-1]}$ in PP.

Recall that in the previous game, the challenger sampled all vectors $(\mathbf{u}_k)_{k\in[L]}$ as uniformly random vectors in $\mathbb{Z}_q^n$ at the beginning of the game. Then, when $\mathcal{A}$ issued a decryption key reveal query on user-item pair $(\mathsf{ID}^*_{[i]}, \mathsf{t}^*)$ for $i \in [\ell^* - 1]$, the challenger sampled a short vector $\mathbf{g}_{\mathsf{ID}^*_{[i]},\mathsf{t}^*}$ such that

$$[\mathbf{A}_{L+1}|\mathbf{E}(\mathsf{ID}^*_{[i]})|\mathbf{F}(\mathsf{t}^*)]\mathbf{g}_{\mathsf{ID}^*_{[i]},\mathsf{t}^*} = \mathbf{u}_i. \tag{10}$$

where the required trapdoor for sampling the vector was created by the challenger from $\mathbf{T}_{\mathbf{A}_{L+1}}$ on the fly, due to the modification we made in Game_0.

We first describe how the vectors $(\mathbf{u}_k)_{k\in[L]}$ in PP are created in this game. The $\mathsf{Game}_{\text{II-2}}$ challenger first samples $(\mathbf{u}_k)_{k\in[L]\setminus[\ell^*-1]}$ as uniformly random vectors in $\mathbb{Z}_q^n$ at the beginning of the game, as was done in the previous game. Next the challenger samples $\mathbf{g}_{\mathsf{ID}^*_{[i]},\mathsf{t}^*} \leftarrow D_{\mathbb{Z}^{i+2},\sigma_i}$ and sets the remaining vectors $(\mathbf{u}_k)_{k\in[\ell^*-1]}$ to satisfy Eq. (10). Then, it keeps the vectors $(\mathbf{g}_{\mathsf{ID}^*_{[i]}}, \mathsf{t}^*)_{i\in[\ell^*-1]}$ secret. All other terms in PP are constructed as in the previous game. In this game, the $\mathsf{Game}_{\text{II-2}}$ challenger answers all the queries made by $\mathcal{A}$ as in the previous game, except for when $\mathcal{A}$ queries the user-item pair $(\mathsf{ID}^*_{[i]}, \mathsf{t}^*)$ for $i \in [\ell^*-1]$ as the decryption key reveal query. For this specific case, the challenger simply returns the vector $\mathbf{g}_{\mathsf{ID}^*_{[i]},\mathsf{t}^*}$ which he has already created at the beginning of the game *without* using $\mathbf{T}_{\mathbf{A}_{L+1}}$. Due to Lemma 2, the distribution of the short vectors provided to $\mathcal{A}$ is distributed statistically close to those of the previous game. Therefore, we have

$$|\Pr[\mathsf{E}_{\text{II-1}}] - \Pr[\mathsf{E}_{\text{II-2}}]| = \mathsf{negl}(\lambda).$$

$\mathsf{Game}_{\text{II-3}}$: In this game, we change how $\mathbf{A}_{L+1}$ is sampled. Namely, in this game, we generate $\mathbf{A}_{L+1}$ as a random matrix in $\mathbb{Z}_q^{n\times m}$ instead of generating it with a trapdoor. By Lemma 3, this makes only a negligible difference. Accordingly, we modify the challenger, so that he does not require $\mathbf{T}_{\mathbf{A}_{L+1}}$ to answer any of the queries made by $\mathcal{A}$. Recall that in the previous game, the challenger used $\mathbf{T}_{\mathbf{A}_{L+1}}$ to create the following terms for the items we provided for reference before $\mathsf{Game}_{\text{II-1}}$:

(a) Item (i) for $\mathsf{ID} \in (\mathcal{ID})^{\le L} \setminus \mathsf{prefix}(\mathsf{ID}^*)$.

(b) Item (ii) for $(\mathsf{ID},\mathsf{t}) \in (\mathcal{ID})^{\le L} \times \mathcal{T} \setminus \{(\mathsf{ID},\mathsf{t}^*) \mid \mathsf{ID} \in \mathsf{prefix}(\mathsf{ID}^*)\}$.

We now show that the $\mathsf{Game}_{\text{II-3}}$ challenger no longer requires $\mathbf{T}_{\mathbf{A}_{L+1}}$ to construct items (a) and (b). In the following, we only show the case for item (a), since the case for item (b) can be easily verified in a similar manner. Now, consider a user $\mathsf{ID} \in (\mathcal{ID})^{\le L} \setminus \mathsf{prefix}(\mathsf{ID}^*)$, where $\mathsf{ID} = (\mathsf{id}_1, \cdots, \mathsf{id}_\ell)$ for some $\ell \in [L]$. Then, let $j \in [\ell]$ be the smallest index such that $\mathsf{ID}_{[j]} \notin \mathsf{prefix}(\mathsf{ID}^*)$, which always exists since $\mathsf{ID} \notin \mathsf{prefix}(\mathsf{ID}^*)$. Let us first consider the case $j \le \ell^*$ and denote $\mathsf{id}_j, \mathsf{id}_j^*$ as the j-th element identities of ID, ID^*, respectively, where we have $H(\mathsf{id}_j) \ne H(\mathsf{id}_j^*)$. Then, to create a trapdoor $\mathbf{T}_{[\mathbf{A}_{L+1}|\mathbf{E}(\mathsf{ID})]}$, the challenger first runs $\mathsf{ExtRndRight}(\mathbf{A}_{L+1}, \mathbf{G}, \mathbf{R}_j^*, \mathbf{T}_{\mathbf{G}}, \sigma_0)$ to create $\mathbf{T}_{[\mathbf{A}_{L+1}|\mathbf{A}_{L+1}\mathbf{R}_j^*+(H(\mathsf{id}_j)-H(\mathsf{id}_j^*)\mathbf{G})]}$. If $\ell = 1$, this is our desired basis. Otherwise, using this basis, the challenger extends it to a basis $\mathbf{T}_{[\mathbf{A}_{L+1}|\mathbf{E}(\mathsf{ID})]}$ by running $\mathsf{ExtRndLeft}$ with parameter σ_ℓ. Note that this can be done since we can rearrange the rows of the basis in an arbitrary manner. Furthermore, in case $j > \ell^*$ (or in particular $j = \ell^* + 1$ by definition), since $H(\mathsf{id}_j) \ne \mathbf{0}_{n\times n}$, we first run algorithm $\mathsf{ExtRndRight}$ to create $\mathbf{T}_{[\mathbf{A}_{L+1}|\mathbf{A}_{L+1}\mathbf{R}_j^*+H(\mathsf{id}_j)\mathbf{G})]}$ and then extend it to a basis $\mathbf{T}_{[\mathbf{A}_{L+1}|\mathbf{E}(\mathsf{ID})]}$ by running $\mathsf{ExtRndLeft}$, as done above. In both cases, the challenger is able to create the required trapdoor without using $\mathbf{T}_{\mathbf{A}_{i^*}}$. Now, due to Lemma 3 and 4, since the sampled vectors and the extended trapdoors are statistically independent from the trapdoors being used, this modification makes a negligible difference. Since, we can make a similar argument in the case for item (b) as well, we obtain

$$|\Pr[\mathsf{E}_{\text{II-2}}] - \Pr[\mathsf{E}_{\text{II-3}}]| = \mathsf{negl}(\lambda).$$

$\mathsf{Game}_{\text{II-4}}$: In this game, we change the way the challenge ciphertext is created. In this game, when the $\mathsf{Game}_{\text{II-4}}$ challenger is issued a challenge query on $(\mathsf{M}_0, \mathsf{M}_1)$ by $\mathcal{A}$, it first samples $\mathbf{s}_i \leftarrow \mathbb{Z}_q^n$ for $i \in [\ell^*] \cup \{L+1\}$, $x \leftarrow D_{\mathbb{Z},\alpha q}$, $\bar{\mathbf{x}} \leftarrow D_{\mathbb{Z}^m,\alpha q}$ and $\mathbf{x}_i \leftarrow D_{\mathbb{Z}^{(i+2)m},\alpha' q}$ for $i \in [\ell^*]$. Then it computes $v = \mathbf{u}_{\ell^*}^\top \mathbf{s}_{L+1} + x \in \mathbb{Z}_q$, $\mathbf{v} = \mathbf{A}_{L+1}^\top \mathbf{s}_{L+1} + \bar{\mathbf{x}} \in \mathbb{Z}_q^m$ and the following terms:

$$\begin{cases} c_0 = v + \mathbf{u}_{\ell^*}^\top \Big(\displaystyle\sum_{i\in[\ell^*]} \mathbf{s}_i \Big) + \mathsf{M}_b \\ \mathbf{c}_i = [\mathbf{A}_i | \mathbf{E}(\mathsf{ID}_{[i]}^*) | \mathbf{F}(\mathsf{t}^*)]^\top \mathbf{s}_i + \mathbf{x}_i \quad \text{for} \quad i \in [\ell^*] \end{cases} \tag{11}$$

where b is the random bit chosen by the challenger. It then sets $\mathbf{R}^* = [\mathbf{R}_{1^*}^* | \cdots | \mathbf{R}_{\ell^*}^* | \mathbf{R}_{L+1}^*] \in \mathbb{Z}^{m\times(\ell^*+1)m}$ and runs

$$\mathsf{ReRand}\Big([\mathbf{I}_m | \mathbf{R}^*], \mathbf{v}, \alpha q, \frac{\alpha'}{2\alpha} \Big) \to \mathbf{c} \in \mathbb{Z}_q^{(\ell^*+2)m}$$

from Lemma 7, where $\mathbf{I}_m$ is the identity matrix with size m. Finally, it sets $\mathbf{c}_{L+1} = \mathbf{c}$ and outputs the challenge ciphertext as follows:

$$\mathsf{ct} = (c_0, \mathbf{c}_1, \ldots, \mathbf{c}_{\ell^*}, \mathbf{c}_{L+1}) \in \mathbb{Z}_q \times \mathbb{Z}_q^{3m} \times \cdots \times \mathbb{Z}_q^{(\ell^*+2)m} \times \mathbb{Z}_q^{(\ell^*+2)m}. \tag{12}$$

We claim that this change alters the view of $\mathcal{A}$ only negligibly, which follows from the noise rerandomization lemma (Lemma 7). In particular, we set $\mathbf{V} = [\mathbf{I}_m|\mathbf{R}^*]$, $\mathbf{b} = \mathbf{A}_{L+1}^\top \mathbf{s}_{L+1}$ and $\mathbf{x} = \bar{\mathbf{x}}$ in Lemma 7 to conclude that the obtained distribution $\mathbf{c}$ is negligibly close to the following:

$$\begin{aligned}\mathbf{c}^\top &= \mathbf{s}_{L+1}^\top \mathbf{A}_{L+1}[\mathbf{I}_m|\mathbf{R}^*] + \mathbf{x}'^\top \\ &= \mathbf{s}_{L+1}^\top[\mathbf{A}_{L+1}|\mathbf{B}_1 + H(\mathsf{id}_1^*)\mathbf{G}|\cdots|\mathbf{B}_{\ell^*} + H(\mathsf{id}_{\ell^*}^*)\mathbf{G}|\mathbf{B}_{L+1} + H(\mathsf{t}^*)\mathbf{G}] + \mathbf{x}'^\top \\ &= \mathbf{s}_{L+1}^\top[\mathbf{A}_{L+1}|\mathbf{E}(\mathsf{ID}^*)|\mathbf{F}(\mathsf{t}^*)] + \mathbf{x}'^\top \in \mathbb{Z}_q^{(\ell^*+2)m}\end{aligned}$$

where $\mathbf{x}'$ is distributed statistically close to $D_{\mathbb{Z}^{(\ell^*+2)m},\alpha' q}$. It can be seen that the challenge ciphertext in Eq. (12) is distributed statistically close to the previous game. Therefore, we have

$$|\Pr[\mathsf{E}_{\text{II-3}}] - \Pr[\mathsf{E}_{\text{II-4}}]| = \mathsf{negl}(\lambda).$$

$\mathsf{Game}_{\text{II-5}}$: In this game, we further change the way the challenge ciphertext is created. In particular, in this game, the $\mathsf{Game}_{\text{II-5}}$ challenger first samples $\mathbf{s}_i \leftarrow \mathbb{Z}_q^n$ for $i \in [\ell^*]$, $w \leftarrow \mathbb{Z}_q$, $\mathbf{w} \leftarrow \mathbb{Z}_q^m$, $x \leftarrow D_{\mathbb{Z},\alpha q}$, $\bar{\mathbf{x}} \leftarrow D_{\mathbb{Z}^m,\alpha q}$ and $\mathbf{x}_i \leftarrow D_{\mathbb{Z}^{(i+2)m},\alpha' q}$ for $i \in [\ell^*]$. Then it computes $v = w + x \in \mathbb{Z}_q$, $\mathbf{v} = \mathbf{w} + \bar{\mathbf{x}} \in \mathbb{Z}_q^m$ and sets the remaining terms as in Eq. (11) of the previous game. Furthermore, it sets $\mathbf{R}^*$ and runs the ReRand algorithm as in $\mathsf{Game}_{\text{II-4}}$. Finally, it sets the challenge ciphertext as in Eq. (12). We can show that

$$|\Pr[\mathsf{E}_{\text{II-4}}] - \Pr[\mathsf{E}_{\text{II-5}}]| = \mathsf{negl}(\lambda),$$

assuming the hardness of the $\mathsf{LWE}_{n,m+1,q,\chi}$ problem. We omit this proof, since it is essentially the same proof we provided to bound the advantage between $\mathsf{Game}_{\text{I-}i^*\text{-6}}$ and $\mathsf{Game}_{\text{I-}i^*\text{-7}}$ in Lemma 12 against the adversary using the Type-I strategy. In particular, instead of viewing the matrix $\mathbf{A}$ provided by the LWE problem as $\mathbf{A}_{i^*}$ in the public parameter PP, we view $\mathbf{A}$ as $\mathbf{A}_{L+1}$. Furthermore, the LWE challenger is able to simulate the game for $\mathcal{A}$ properly, since we modified the challenger in $\mathsf{Game}_{\text{II-2}}$ and $\mathsf{Game}_{\text{II-3}}$ so that it does not require $\mathbf{T}_{\mathbf{A}_{L+1}}$ anymore to answer any of $\mathcal{A}$'s queries.

Finally, since v is distributed uniformly at random over $\mathbb{Z}_q$ and independently of all other terms, the probability of adversary $\mathcal{A}$ guessing whether $b = 0$ or $b = 1$ is exactly $1/2$. In particular, we have

$$\Pr[\mathsf{E}_{\text{II-5}}] = \frac{1}{2}.$$

Combining everything together, we conclude that if the adversary $\mathcal{A}$ uses the Type-II strategy, then $\Pr[\mathsf{E}_0] = \frac{1}{2} \pm \mathsf{negl}(\lambda)$ assuming the hardness of $\mathsf{LWE}_{n,m,q,\chi}$ problem. □

Therefore, combining the two Lemmas 12 and 13, and the strategy dividing lemma (Lemmas 8), we can conclude that the RHIBE scheme Π satisfies selective-identity security. □

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 6110 of *Lecture Notes in Computer Science*, pages 553–572. Springer, 2010.

[AI09a] Nuttapong Attrapadung and Hideki Imai. Attribute-based encryption supporting direct/indirect revocation modes. In Matthew G. Parker, editor, *Cryptography and Coding, 12th IMA International Conference, Cryptography and Coding 2009*, volume 5921 of *Lecture Notes in Computer Science*, pages 278–300. Springer, 2009.

[AI09b] Nuttapong Attrapadung and Hideki Imai. Conjunctive broadcast and attribute-based encryption. In Hovav Shacham and Brent Waters, editors, *Pairing-Based Cryptography - Pairing 2009, Third International Conference*, volume 5671 of *Lecture Notes in Computer Science*, pages 248–265. Springer, 2009.

[Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *Automata, Languages and Programming, 26th International Colloquium, ICALP'99*, volume 1644 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 1999.

[AP11] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. *Theory Comput. Syst.*, 48(3):535–553, 2011.

[BF03] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[BGK08] Alexandra Boldyreva, Vipul Goyal, and Virendra Kumar. Identity-based encryption with efficient revocation. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008*, pages 417–426. ACM, 2008.

[CCKS18] Donghoon Chang, Amit Kumar Chauhan, Sandeep Kumar, and Somitra Kumar Sanadhya. Revocable identity-based encryption from codes with rank metric. In Nigel P. Smart, editor, *Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference 2018*, volume 10808 of *Lecture Notes in Computer Science*, pages 435–451. Springer, 2018.

[CDLQ16] Hui Cui, Robert H. Deng, Yingjiu Li, and Baodong Qin. Server-aided revocable attribute-based encryption. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *Computer Security - ESORICS 2016 - 21st European Symposium on Research in Computer Security*, volume 9879 of *Lecture Notes in Computer Science*, pages 570–587. Springer, 2016.

[CHKP12] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *J. Cryptology*, 25(4):601–639, 2012.

[CLL$^+$12a] Jie Chen, Hoon Wei Lim, San Ling, Le Su, and Huaxiong Wang. Anonymous and adaptively secure revocable IBE with constant size public parameters. *CoRR*, abs/1210.6441, 2012.

[CLL$^+$12b] Jie Chen, Hoon Wei Lim, San Ling, Huaxiong Wang, and Khoa Nguyen. Revocable identity-based encryption from lattices. In Willy Susilo, Yi Mu, and Jennifer Seberry, editors, *Information Security and Privacy - 17th Australasian Conference, ACISP 2012*, volume 7372 of *Lecture Notes in Computer Science*, pages 390–403. Springer, 2012.

[DG17] Nico Döttling and Sanjam Garg. From selective IBE to full IBE and selective HIBE. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography - 15th International Conference, TCC 2017*, volume 10677 of *Lecture Notes in Computer Science*, pages 372–408. Springer, 2017.

[ESY16] Keita Emura, Jae Hong Seo, and Taek-Young Youn. Semi-generic transformation of revocable hierarchical identity-based encryption and its DBDH instantiation. *IEICE Transactions*, 99-A(1):83–91, 2016.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 197–206. ACM, 2008.

[ISW17] Yuu Ishida, Junji Shikata, and Yohei Watanabe. CCA-secure revocable identity-based encryption schemes with decryption key exposure resistance. *IJACT*, 3(3):288–311, 2017.

[IWS15] Yuu Ishida, Yohei Watanabe, and Junji Shikata. Constructions of CCA-secure revocable identity-based encryption. In Ernest Foo and Douglas Stebila, editors, *Information Security and Privacy - 20th Australasian Conference, ACISP 2015*, volume 9144 of *Lecture Notes in Computer Science*, pages 174–191. Springer, 2015.

[KY16] Shuichi Katsumata and Shota Yamada. Partitioning via non-linear polynomial functions: More compact ibes from ideal lattices and bilinear maps. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security*, volume 10032 of *Lecture Notes in Computer Science*, pages 682–712, 2016.

[Lee16] Kwangsu Lee. Revocable hierarchical identity-based encryption with adaptive security. *IACR Cryptology ePrint Archive*, 2016:749, 2016.

[LLP17] Kwangsu Lee, Dong Hoon Lee, and Jong Hwan Park. Efficient revocable identity-based encryption via subset difference methods. *Des. Codes Cryptography*, 85(1):39–76, 2017.

[LNWZ17] San Ling, Khoa Nguyen, Huaxiong Wang, and Juanyang Zhang. Revocable predicate encryption from lattices. In Tatsuaki Okamoto, Yong Yu, Man Ho Au, and Yannan Li, editors, *Provable Security - 11th International Conference, ProvSec 2017*, volume 10592 of *Lecture Notes in Computer Science*, pages 305–326. Springer, 2017.

[LNWZ18] San Ling, Khoa Nguyen, Huaxiong Wang, and Juanyang Zhang. Server-aided revocable predicate encryption: Formalization and lattice-based instantiation. *CoRR*, abs/1801.07844, 2018.

[LP16] Kwangsu Lee and Seunghwan Park. Revocable hierarchical identity-based encryption with shorter private keys and update keys. *IACR Cryptology ePrint Archive*, 2016:460, 2016.

[LV09] Benoît Libert and Damien Vergnaud. Adaptive-id secure revocable identity-based encryption. In Marc Fischlin, editor, *Topics in Cryptology - CT-RSA 2009, The Cryptographers' Track at the RSA Conference 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2009.

[MLC$^+$15] Xianping Mao, Junzuo Lai, Kefei Chen, Jian Weng, and Qixiang Mei. Efficient revocable identity-based encryption from multilinear maps. *Security and Communication Networks*, 8(18):3511–3522, 2015.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, 2012.

[NMS12] Juan Manuel González Nieto, Mark Manulis, and Dongdong Sun. Fully private revocable predicate encryption. In Willy Susilo, Yi Mu, and Jennifer Seberry, editors, *Information Security and Privacy - 17th Australasian Conference, ACISP 2012*, volume 7372 of *Lecture Notes in Computer Science*, pages 350–363. Springer, 2012.

[NNL01] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer, 2001.

[NWZ16] Khoa Nguyen, Huaxiong Wang, and Juanyang Zhang. Server-aided revocable identity-based encryption from lattices. In Sara Foresti and Giuseppe Persiano, editors, *Cryptology and Network Security - 15th International Conference, CANS 2016*, volume 10052 of *Lecture Notes in Computer Science*, pages 107–123, 2016.

[PLL15] Seunghwan Park, Kwangsu Lee, and Dong Hoon Lee. New constructions of revocable identity-based encryption from multilinear maps. *IEEE Trans. Information Forensics and Security*, 10(8):1564–1577, 2015.

[PLL16] Seunghwan Park, Dong Hoon Lee, and Kwangsu Lee. Revocable hierarchical identity-based encryption from multilinear maps. *CoRR*, abs/1610.07948, 2016.

[QDLL15] Baodong Qin, Robert H. Deng, Yingjiu Li, and Shengli Liu. Server-aided revocable identity-based encryption. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security*, volume 9326 of *Lecture Notes in Computer Science*, pages 286–304. Springer, 2015.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM, 2005.

[RLPL15] Geumsook Ryu, Kwangsu Lee, Seunghwan Park, and Dong Hoon Lee. Unbounded hierarchical identity-based encryption with efficient revocation. In Howon Kim and Dooho Choi, editors, *Information Security Applications - 16th International Workshop, WISA 2015*, volume 9503 of *Lecture Notes in Computer Science*, pages 122–133. Springer, 2015.

[SE13a] Jae Hong Seo and Keita Emura. Efficient delegation of key generation and revocation functionalities in identity-based encryption. In Ed Dawson, editor, *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013*, volume 7779 of *Lecture Notes in Computer Science*, pages 343–358. Springer, 2013.

[SE13b] Jae Hong Seo and Keita Emura. Revocable identity-based encryption revisited: Security model and construction. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 216–234. Springer, 2013.

[SE14a] Jae Hong Seo and Keita Emura. Revocable hierarchical identity-based encryption. *Theor. Comput. Sci.*, 542:44–62, 2014.

[SE14b] Jae Hong Seo and Keita Emura. Revocable identity-based cryptosystem revisited: Security models and constructions. *IEEE Trans. Information Forensics and Security*, 9(7):1193–1205, 2014.

[SE15] Jae Hong Seo and Keita Emura. Revocable hierarchical identity-based encryption: History-free update, security against insiders, and short ciphertexts. In Kaisa Nyberg, editor, *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015*, volume 9048 of *Lecture Notes in Computer Science*, pages 106–123. Springer, 2015.

[SE16] Jae Hong Seo and Keita Emura. Revocable hierarchical identity-based encryption via history-free approach. *Theor. Comput. Sci.*, 615:45–60, 2016.

[TW17] Atsushi Takayasu and Yohei Watanabe. Lattice-based revocable identity-based encryption with bounded decryption key exposure resistance. In Josef Pieprzyk and Suriadi Suriadi, editors, *Information Security and Privacy - 22nd Australasian Conference, ACISP 2017*, volume 10342 of *Lecture Notes in Computer Science*, pages 184–204. Springer, 2017.

[WES17] Yohei Watanabe, Keita Emura, and Jae Hong Seo. New revocable IBE in prime-order groups: Adaptively secure, decryption key exposure resistant, and with short public parameters. In Helena Handschuh, editor, *Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017*, volume 10159 of *Lecture Notes in Computer Science*, pages 432–449. Springer, 2017.

A Definitions

A.1 Revocable Identity-Based Encryption

Here, we formally define *revocable identity-based encryption (RIBE)*. Basically, the definitions we give here are done in the same way as those for RHIBE, except that the depth of the identity hierarchy is fixed to be 1.

Syntax. An RIBE scheme Π consists of the six algorithms ($\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{GenSK}, \mathsf{KeyUp}, \mathsf{GenDK}, \mathsf{Decrypt}$) with the following interface:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{PP}, \mathsf{sk_{kgc}})$: This is the *setup* algorithm that takes the security parameter 1^λ as input, and outputs a public parameter PP and the KGC's secret key $\mathsf{sk_{kgc}}$ (also called a master secret key).

We assume that the plaintext space $\mathcal{M}$, the time period space $\mathcal{T}$, and the identity space $\mathcal{ID}$ are determined only by the security parameter λ, and their descriptions are contained in PP.

$\mathsf{Encrypt}(\mathsf{PP}, \mathsf{ID}, \mathsf{t}, \mathsf{M}) \rightarrow \mathsf{ct}$: This is the *encryption* algorithm that takes a public parameter PP, an identity ID, a time period t, and a plaintext M as input, and outputs a ciphertext ct.

$\mathsf{GenSK}(\mathsf{PP}, \mathsf{sk}_{\mathsf{kgc}}, \mathsf{ID}) \rightarrow (\mathsf{sk}_{\mathsf{ID}}, \mathsf{sk}'_{\mathsf{kgc}})$: This is the *secret key generation* algorithm that takes a public parameter PP, the KGC's secret key $\mathsf{sk}_{\mathsf{kgc}}$, and an identity $\mathsf{ID} \in \mathcal{ID}$ as input, and may update the KGC's secret key $\mathsf{sk}_{\mathsf{kgc}}$. Then, it outputs a secret key $\mathsf{sk}_{\mathsf{ID}}$ for the identity ID and also the KGC's "updated" secret key $\mathsf{sk}'_{\mathsf{kgc}}$.

$\mathsf{KeyUp}(\mathsf{PP}, \mathsf{t}, \mathsf{sk}_{\mathsf{kgc}}, \mathsf{RL}_{\mathsf{t}}) \rightarrow (\mathsf{ku}_{\mathsf{t}}, \mathsf{sk}'_{\mathsf{kgc}})$: This is the *key update information generation* algorithm that takes a public parameter PP, a time period t, the KGC's secret key $\mathsf{sk}_{\mathsf{kgc}}$, and a revocation list $\mathsf{RL}_{\mathsf{t}} \subseteq \mathcal{ID}$ as input, and may update the KGC's secret key $\mathsf{sk}_{\mathsf{kgc}}$. Then, it outputs a key update ku_{t} and also the "updated" KGC's secret key $\mathsf{sk}'_{\mathsf{kgc}}$.

$\mathsf{GenDK}(\mathsf{PP}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ku}_{\mathsf{t}}) \rightarrow \mathsf{dk}_{\mathsf{ID},\mathsf{t}}$ or $\perp$: This is the *decryption key generation* algorithm that takes a public parameter PP, a secret key $\mathsf{sk}_{\mathsf{ID}}$, and a key update ku_{t} as input, and outputs a decryption key $\mathsf{dk}_{\mathsf{ID},\mathsf{t}}$ for the time period t or the special "invalid" symbol $\perp$ indicating that ID has been revoked.

$\mathsf{Decrypt}(\mathsf{PP}, \mathsf{dk}_{\mathsf{ID},\mathsf{t}}, \mathsf{ct}) \rightarrow \mathsf{M}$: This is the *decryption* algorithm that takes a public parameter PP, a decryption key $\mathsf{dk}_{\mathsf{ID},\mathsf{t}}$ and a ciphertext ct as input, and outputs the decryption result M.

Correctness. We require the following to hold for an RIBE scheme. Informally, we require a ciphertext corresponding to a user ID for time period t to be properly decrypted by user ID if the user is not revoked on time t. To fully capture this, we consider all the possible scenarios of creating the secret key for user ID. Namely, for all $\lambda \in \mathbb{N}$, $(\mathsf{PP}, \mathsf{sk}_{\mathsf{kgc}}) \leftarrow \mathsf{Setup}(1^{\lambda})$, $\mathsf{ID} \in \mathcal{ID}$, $\mathsf{t} \in \mathcal{T}$, $\mathsf{M} \in \mathcal{M}$, and $\mathsf{RL}_{\mathsf{t}} \subseteq \mathcal{ID} \setminus \{\mathsf{ID}\}$, we require $\mathsf{M}' = \mathsf{M}$ to hold after executing the following procedures:

(1) $(\mathsf{sk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{kgc}}) \leftarrow \mathsf{GenSK}(\mathsf{PP}, \mathsf{sk}_{\mathsf{kgc}}, \mathsf{ID})$.
(2) $(\mathsf{ku}_{\mathsf{t}}, \mathsf{sk}_{\mathsf{kgc}}) \leftarrow \mathsf{KeyUp}(\mathsf{PP}, \mathsf{t}, \mathsf{sk}_{\mathsf{kgc}}, \mathsf{RL}_{\mathsf{t}})$.
(3) $\mathsf{dk}_{\mathsf{ID},\mathsf{t}} \leftarrow \mathsf{GenDK}(\mathsf{PP}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ku}_{\mathsf{t}})$.
(4) $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{PP}, \mathsf{ID}, \mathsf{t}, \mathsf{M})$.
(5) $\mathsf{M}' \leftarrow \mathsf{Decrypt}(\mathsf{PP}, \mathsf{dk}_{\mathsf{ID},\mathsf{t}}, \mathsf{ct})$.

We note that, the most stringent way to define correctness would be to also capture the fact that the secret key $\mathsf{sk}_{\mathsf{kgc}}$ could be updated after executing GenSK. In particular, the output of KeyUp, which takes as input the KGC's secret key $\mathsf{sk}_{\mathsf{kgc}}$, may differ in general before and after GenSK is run. Therefore, to be more precise, we should also allow an arbitrary (polynomial) number of executions of GenSK in between steps (1) and (2). However, we defined correctness as above for the sake of simplicity and readability. We note that our scheme satisfies the more stringent correctness (which will be obvious from construction).

Security Definitions. Here, we give the security definitions of an RIBE scheme $\Pi = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{GenSK}, \mathsf{KeyUp}, \mathsf{GenDK}, \mathsf{Decrypt})$. Our default security definition captures the so-called *decryption key exposure resistance* (DKER). However, since we consider a generic transformation that converts any RIBE without DKER into the one with DKER, we also introduce security without DKER. (We will simply refer to security without DKER as *weak security.*) Furthermore, for each notion, we consider selective-identity security and adaptive-identity security, which results in four security notions in total.

We first give the formal definition of selective-identity security (with DKER) via a game between an adversary $\mathcal{A}$ and the challenger $\mathcal{C}$. (The remaining security notions are derived by appropriately changing the game.) The game is parameterized by the security parameter λ, and has the global counter t_{cu}, initialized with 1, that denotes the "current time period" with which $\mathcal{C}$'s responses to $\mathcal{A}$'s queries are controlled. The game proceeds as follows:

At the beginning, $\mathcal{A}$ sends the challenge identity/time period pair $(\mathsf{ID}^*, \mathsf{t}^*) \in \mathcal{ID} \times \mathcal{T}$ to $\mathcal{C}$. Next, $\mathcal{C}$ runs $(\mathsf{PP}, \mathsf{sk}_{\mathsf{kgc}}) \leftarrow \mathsf{Setup}(1^\lambda)$, and prepares a list $\mathtt{SKList}$ that initially contains $(\mathsf{kgc}, \mathsf{sk}_{\mathsf{kgc}})$, and into which identity/secret key pairs $(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}})$ generated during the game will be stored. From this point on, whenever a new secret key is generated for an identity $\mathsf{ID} \in \mathcal{ID}$ or the secret key $\mathsf{sk}_{\mathsf{kgc}}$ is updated due to the execution of GenSK or KeyUp, $\mathcal{C}$ will store $(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}})$ or update the corresponding entry $(\mathsf{kgc}, \mathsf{sk}_{\mathsf{kgc}})$ in $\mathtt{SKList}$, and we will not explicitly mention this addition/update. Then, $\mathcal{C}$ executes $(\mathsf{ku}_{\mathsf{kgc},1}, \mathsf{sk}'_{\mathsf{kgc}}) \leftarrow \mathsf{KeyUp}(\mathsf{PP}, \mathsf{t}_{\mathsf{cu}} = 1, \mathsf{sk}_{\mathsf{kgc}}, \mathsf{RL}_1 = \emptyset)$ for generating the initial time period $\mathsf{t}_{\mathsf{cu}} = 1$. After that, $\mathcal{C}$ gives PP and ku_1 to $\mathcal{A}$.

From this point on, $\mathcal{A}$ may adaptively make the following five types of queries to $\mathcal{C}$:

Secret Key Generation Query: Upon a query $\mathsf{ID} \in \mathcal{ID}$ from $\mathcal{A}$, where it is required that $(\mathsf{ID}, *) \notin \mathtt{SKList}$, $\mathcal{C}$ executes $(\mathsf{sk}_{\mathsf{ID}}, \mathsf{sk}'_{\mathsf{kgc}}) \leftarrow \mathsf{GenSK}(\mathsf{PP}, \mathsf{sk}_{\mathsf{kgc}}, \mathsf{ID})$ (and returns nothing to $\mathcal{A}$).

We require that all identities ID appearing in the following queries (except the challenge query) be "activated" in the sense that $\mathsf{sk}_{\mathsf{ID}}$ is generated via this query and hence $(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}}) \in \mathtt{SKList}$.

Secret Key Reveal Query: Upon a query $\mathsf{ID} \in \mathcal{ID}$ from $\mathcal{A}$, $\mathcal{C}$ checks if the following condition is satisfied:

- If $\mathsf{t}_{\mathsf{cu}} \geq \mathsf{t}^*$ and $\mathsf{ID}^* \notin \mathsf{RL}_{\mathsf{t}^*}$, then $\mathsf{ID} \neq \mathsf{ID}^*$.

If this condition is *not* satisfied, then $\mathcal{C}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ finds $\mathsf{sk}_{\mathsf{ID}}$ from $\mathtt{SKList}$, and returns it to $\mathcal{A}$.

Revoke & Key Update Query: Upon a query $\mathsf{RL} \subseteq \mathcal{ID}$ (which denotes the set of identities that are going to be revoked in the next time period) from $\mathcal{A}$, $\mathcal{C}$ checks if the following conditions are satisfied simultaneously:

- $\mathsf{RL}_{\mathsf{t}_{\mathsf{cu}}} \subseteq \mathsf{RL}$.
- If $\mathsf{t}_{\mathsf{cu}} = \mathsf{t}^* - 1$ and $\mathsf{sk}_{\mathsf{ID}^*}$ for the challenge ID^* has been revealed to $\mathcal{A}$ via a secret key reveal query ID^*, then $\mathsf{ID}^* \in \mathsf{RL}$.

If these conditions are *not* satisfied, then $\mathcal{C}$ returns $\perp$ to $\mathcal{A}$.

Otherwise $\mathcal{C}$ increments the current time period by $\mathsf{t}_{\mathsf{cu}} \leftarrow \mathsf{t}_{\mathsf{cu}} + 1$. Then, $\mathcal{C}$ sets $\mathsf{RL}_{\mathsf{t}_{\mathsf{cu}}} \leftarrow \mathsf{RL}$, and runs $(\mathsf{ku}_{\mathsf{t}_{\mathsf{cu}}}, \mathsf{sk}'_{\mathsf{kgc}}) \leftarrow \mathsf{KeyUp}(\mathsf{PP}, \mathsf{t}_{\mathsf{cu}}, \mathsf{sk}_{\mathsf{kgc}}, \mathsf{RL}_{\mathsf{t}_{\mathsf{cu}}})$. Finally, $\mathcal{C}$ returns $\mathsf{ku}_{\mathsf{t}_{\mathsf{cu}}}$ to $\mathcal{A}$.

Decryption Key Reveal Query: Upon a query $(\mathsf{ID}, \mathsf{t}) \in \mathcal{ID} \times \mathcal{T}$ from $\mathcal{A}$, $\mathcal{C}$ checks if the following conditions are simultaneously satisfied:

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

If these conditions are *not* satisfied, then $\mathcal{C}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ finds $\mathsf{sk}_{\mathsf{ID}}$ from $\mathtt{SKList}$, runs $\mathsf{dk}_{\mathsf{ID},\mathsf{t}} \leftarrow \mathsf{GenDK}(\mathsf{PP}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ku}_{\mathsf{t}})$, and returns $\mathsf{dk}_{\mathsf{ID},\mathsf{t}}$ to $\mathcal{A}$.

Challenge Query: $\mathcal{A}$ is allowed to make this query only once. Upon a query $(\mathsf{M}_0, \mathsf{M}_1)$ from $\mathcal{A}$, where it is required that $|\mathsf{M}_0| = |\mathsf{M}_1|$, $\mathcal{C}$ picks the challenge bit $b \in \{0, 1\}$ uniformly at random, runs $\mathsf{ct}^* \leftarrow \mathsf{Encrypt}(\mathsf{PP}, \mathsf{ID}^*, \mathsf{t}^*, \mathsf{M}_b)$, and returns the challenge ciphertext ct^* to $\mathcal{A}$.

At some point, $\mathcal{A}$ outputs $b' \in \{0, 1\}$ as its guess for b and terminates.

The above completes the description of the game. In this game, $\mathcal{A}$'s selective-security advantage $\mathsf{Adv}^{\mathtt{RIBE\text{-}sel}}_{\Pi,\mathcal{A}}(\lambda)$ is defined by $\mathsf{Adv}^{\mathtt{RIBE\text{-}sel}}_{\Pi,\mathcal{A}}(\lambda) := 2 \cdot |\Pr[b' = b] - 1/2|$.

Definition 3. *We say that an RIBE scheme* Π *satisfies* selective-identity *security, if the advantage* $\mathsf{Adv}^{\mathtt{RIBE\text{-}sel}}_{\Pi,\mathcal{A}}(\lambda)$ *is negligible for all PPT adversaries* $\mathcal{A}$.

The more desirable security notion, called *adaptive-identity* security, is defined in the same way as selective-identity security, except that in the security game the adversary $\mathcal{A}$ chooses the pair of the challenge identity and time period $(\mathsf{ID}^*, \mathsf{t}^*)$ not at the beginning of the game, but at the time it makes the challenge query. More formally, the response to the challenge query is defined differently as follows:

Challenge Query: $\mathcal{A}$ is allowed to make this query only once. Upon a query $(\mathsf{ID}^*, \mathsf{t}^*, \mathsf{M}_0, \mathsf{M}_1)$ from $\mathcal{A}$, where it is required that the following conditions are satisfied simultaneously:

- $|\mathsf{M}_0| = |\mathsf{M}_1|$,
- if $\mathsf{t}^* \leq \mathsf{t}_{\mathsf{cu}}$, then $\mathcal{A}$ has not submitted $(\mathsf{ID}^*, \mathsf{t}^*)$ as a decryption key reveal query, and
- if $\mathsf{sk}_{\mathsf{ID}^*}$ has been revealed to $\mathcal{A}$, then it is required that $\mathsf{ID}^* \in \mathsf{RL}_{\mathsf{t}^*}$,

$\mathcal{C}$ picks the challenge bit $b \in \{0,1\}$ uniformly at random, runs $\mathsf{ct}^* \leftarrow \mathsf{Encrypt}(\mathsf{PP}, \mathsf{ID}^*, \mathsf{t}^*, \mathsf{M}_b)$, and returns the challenge ciphertext ct^* to $\mathcal{A}$.

The adaptive-identity security advantage $\mathsf{Adv}^{\mathtt{RIBE\text{-}ad}}_{\Pi,\mathcal{A}}(\lambda)$ of the adversary $\mathcal{A}$ is defined analogously to that for selective-identity security.

Definition 4. *We say that an RIBE scheme* Π *satisfies* adaptive-identity security*, if the advantage* $\mathsf{Adv}^{\mathtt{RIBE\text{-}ad}}_{\Pi,\mathcal{A}}(\lambda)$ *is negligible for all PPT adversaries* $\mathcal{A}$*.*

The weak security notions (i.e. security without DKER) are defined by changing the corresponding games so that an adversary $\mathcal{A}$ is not allowed to make any decryption key reveal query.[18] We denote the weak selective-identity (resp. adaptive-identity) security advantage of the adversary $\mathcal{A}$ by $\mathsf{Adv}^{\mathtt{RIBE\text{-}sel\text{-}weak}}_{\Pi,\mathcal{A}}(\lambda)$ (resp. $\mathsf{Adv}^{\mathtt{RIBE\text{-}ad\text{-}weak}}_{\Pi,\mathcal{A}}(\lambda)$).

Definition 5. *We say that an RIBE scheme* Π *satisfies* weak selective-identity security*, if the advantage* $\mathsf{Adv}^{\mathtt{RIBE\text{-}sel\text{-}weak}}_{\Pi,\mathcal{A}}(\lambda)$ *is negligible for all PPT adversaries* $\mathcal{A}$*.*

We define weak adaptive-identity security *analogously.*

A.2 2-Level Hierarchical Identity Based Encryption

In this work, we use a 2-level HIBE scheme as a building block for our security-enhancing generic transformation for RIBE, and thus we recall it here. Our definition here is customized from a typical definition of HIBE. Specifically, since we only consider 2-level HIBE, we differentiate the key generation by the KGC and by each user, and refer to the key generation algorithm for the latter as the *delegation algorithm*. Also, we consider the encryption and decryption algorithms only for level-2 users. To the best of our knowledge, all the existing HIBE scheme satisfy the modification.

Syntax. A 2-level HIBE scheme Π consists of the five algorithms $(\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{GenSK}, \mathsf{Delegate}, \mathsf{Decrypt})$ with the following interface:

$\mathsf{Setup}(1^\lambda) \rightarrow (\mathsf{PP}, \mathsf{sk}_{\mathsf{kgc}})$: This is the *setup* algorithm that takes the security parameter 1^λ as input, and outputs a public parameter PP and the KGC's secret key $\mathsf{sk}_{\mathsf{kgc}}$ (also called a master secret key).

We assume that the plaintext space $\mathcal{M}$ and the element identity space $\mathcal{ID}$ are determined only by the security parameter λ, and their descriptions are contained in PP.

[18] In other words, if an adversary $\mathcal{A}$ in the weak security game wants to obtain decryption keys $\mathsf{dk}_{\mathsf{ID}^*,\mathsf{t}}$ for any $\mathsf{t} \neq \mathsf{t}^*$, it should make a secret key reveal query on ID^*. Hence, ID^* will be revoked before t^*. On the other hand, an adversary $\mathcal{A}$ in the standard security game with DKER can obtain the decryption keys $\mathsf{dk}_{\mathsf{ID}^*,\mathsf{t}}$ without revoking ID^* by t^*.

$\mathsf{Encrypt}(\mathsf{PP}, \mathsf{ID} = (\mathsf{id}_1, \mathsf{id}_2), \mathsf{M}) \to \mathsf{ct}$: This is the *encryption* algorithm (for a level-2 user) that takes a public parameter PP, a level-2 user's identity $\mathsf{ID} = (\mathsf{id}_1, \mathsf{id}_2) \in (\mathcal{ID})^2$, and a plaintext M as input, and outputs a ciphertext ct.

$\mathsf{GenSK}(\mathsf{PP}, \mathsf{sk}_{\mathsf{kgc}}, \mathsf{id}_1) \to \mathsf{sk}_{\mathsf{id}_1}$: This is the *secret key generation* algorithm that takes a public parameter PP, the KGC's secret key $\mathsf{sk}_{\mathsf{kgc}}$, and a first-level identity $\mathsf{id}_1 \in \mathcal{ID}$ as input, and outputs a secret key $\mathsf{sk}_{\mathsf{id}_1}$.

$\mathsf{Delegate}(\mathsf{PP}, \mathsf{sk}_{\mathsf{id}_1}, \mathsf{id}_2) \to \mathsf{sk}_{\mathsf{id}_1,\mathsf{id}_2}$: This is the *delegation* algorithm that takes a public parameter PP, a secret key $\mathsf{sk}_{\mathsf{id}_1}$ (of a first-level user with $\mathsf{id}_1 \in \mathcal{ID}$), and a second-level (element) identity $\mathsf{id}_2 \in \mathcal{ID}$ as input, and outputs a secret key $\mathsf{sk}_{\mathsf{id}_1,\mathsf{id}_2}$.

$\mathsf{Decrypt}(\mathsf{PP}, \mathsf{sk}_{\mathsf{id}_1,\mathsf{id}_2}, \mathsf{ct}) \to \mathsf{M}$: This is the *decryption* algorithm that takes a public parameter PP, a decryption key $\mathsf{dk}_{\mathsf{id}_1,\mathsf{id}_2}$ (for a level-2 user with identity $\mathsf{ID} = (\mathsf{id}_1, \mathsf{id}_2)$), and a ciphertext ct as input, and outputs the decryption result M.

Correctness. We require the following to hold for a 2-level HIBE scheme. For all $\lambda \in \mathbb{N}$, $(\mathsf{PP}, \mathsf{sk}_{\mathsf{kgc}}) \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{ID} = (\mathsf{id}_1, \mathsf{id}_2) \in (\mathcal{ID})^2$, $\mathsf{sk}_{\mathsf{id}_1} \leftarrow \mathsf{GenSK}(\mathsf{PP}, \mathsf{sk}_{\mathsf{kgc}}, \mathsf{id}_1)$, $\mathsf{sk}_{\mathsf{id}_1,\mathsf{id}_2} \leftarrow \mathsf{Delegate}(\mathsf{PP}, \mathsf{sk}_{\mathsf{id}_1}, \mathsf{id}_2)$, $\mathsf{M} \in \mathcal{M}$, and $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{PP}, \mathsf{ID}, \mathsf{M})$, it holds that $\mathsf{Decrypt}(\mathsf{PP}, \mathsf{sk}_{\mathsf{id}_1,\mathsf{id}_2}, \mathsf{ct}) = \mathsf{M}$.

Security Definition. Here, we give the security definitions of a 2-level HIBE scheme $\Pi = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{GenSK}, \mathsf{Delegate}, \mathsf{Decrypt})$. We first give the definition of selective-identity security, which is defined via the following game between an adversary $\mathcal{A}$ and the challenger $\mathcal{C}$:

At the beginning, $\mathcal{A}$ sends the challenge identity $\mathsf{ID}^* = (\mathsf{id}_1^*, \mathsf{id}_2^*) \in (\mathcal{ID})^2$ to $\mathcal{C}$. Next, $\mathcal{C}$ runs $(\mathsf{PP}, \mathsf{sk}_{\mathsf{kgc}}) \leftarrow \mathsf{Setup}(1^\lambda)$, and prepares a list $\mathtt{SKList}$ that initially contains $(\mathsf{kgc}, \mathsf{sk}_{\mathsf{kgc}})$, and into which identity/secret key pairs $(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}})$ generated during the game will be stored. From this point on, whenever a new secret key is generated for an identity $\mathsf{ID} \in (\mathcal{ID})^{\le 2}$, $\mathcal{C}$ will store $(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}})$ in $\mathtt{SKList}$, and we will not explicitly mention this procedure. After that, $\mathcal{C}$ gives PP to $\mathcal{A}$.

From this point on, $\mathcal{A}$ may adaptively make the following four types of queries to $\mathcal{C}$:

Level-1 Secret Key Generation Query: Upon a query $\mathsf{id}_1 \in \mathcal{ID}$ from $\mathcal{A}$, $\mathcal{C}$ checks if $(\mathsf{id}_1, *) \in \mathtt{SKList}$, and returns $\perp$ to $\mathcal{A}$ if this is the case. Otherwise, $\mathcal{C}$ executes $\mathsf{sk}_{\mathsf{id}_1} \leftarrow \mathsf{GenSK}(\mathsf{PP}, \mathsf{sk}_{\mathsf{kgc}}, \mathsf{id}_1)$ (but returns nothing to $\mathcal{A}$).[19]

Level-1 Secret Key Reveal Query: Upon a query $\mathsf{id}_1 \in \mathcal{ID}$ from $\mathcal{A}$, $\mathcal{C}$ checks if $(\mathsf{id}_1, \mathsf{sk}_{id_1}) \in \mathtt{SKList}$ for some $\mathsf{sk}_{\mathsf{id}_1}$ and $\mathsf{id}_1 \neq \mathsf{id}_1^*$. If this is *not* the case, then $\mathcal{C}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ returns $\mathsf{sk}_{\mathsf{id}_1}$ to $\mathcal{A}$.

Level-2 Secret Key Reveal Query: Upon a query $(\mathsf{id}_1, \mathsf{id}_2) \in (\mathcal{ID})^2$ from $\mathcal{A}$, $\mathcal{C}$ checks if $(\mathsf{id}_1, \mathsf{sk}_{\mathsf{id}_1}) \in \mathtt{SKList}$ for some $\mathsf{sk}_{\mathsf{id}_1}$, $((\mathsf{id}_1, \mathsf{id}_2), \mathsf{sk}_{\mathsf{id}_1,\mathsf{id}_2}) \notin \mathtt{SKList}$, and $(\mathsf{id}_1, \mathsf{id}_2) \neq (\mathsf{id}_1^*, \mathsf{id}_2^*)$. If this is *not* the case, then $\mathcal{C}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ executes $\mathsf{sk}_{\mathsf{id}_1,\mathsf{id}_2} \leftarrow \mathsf{Delegate}(\mathsf{PP}, \mathsf{sk}_{\mathsf{id}_1}, \mathsf{id}_2)$, and returns $\mathsf{sk}_{\mathsf{id}_1,\mathsf{id}_2}$ to $\mathcal{A}$.

Challenge Query: $\mathcal{A}$ is allowed to make this query only once. Upon a query $(\mathsf{M}_0, \mathsf{M}_1)$ from $\mathcal{A}$, where it is required that $|\mathsf{M}_0| = |\mathsf{M}_1|$, $\mathcal{C}$ picks the challenge bit $b \in \{0, 1\}$ uniformly at random, runs $\mathsf{ct}^* \leftarrow \mathsf{Encrypt}(\mathsf{PP}, \mathsf{ID}^* = (\mathsf{id}_1^*, \mathsf{id}_2^*), \mathsf{M}_b)$, and returns the challenge ciphertext ct^* to $\mathcal{A}$.

At some point, $\mathcal{A}$ outputs $b' \in \{0, 1\}$ as its guess for b and terminates.

The above completes the description of the game. In this game, $\mathcal{A}$'s selective-identity security advantage $\mathsf{Adv}^{\mathtt{HIBE\text{-}sel}}_{\Pi,\mathcal{A}}(\lambda)$ is defined by $\mathsf{Adv}^{\mathtt{HIBE\text{-}sel}}_{\Pi,\mathcal{A}}(\lambda) := 2 \cdot |\Pr[b' = b] - 1/2|$.

[19] Note that just making this query does not return $\mathsf{sk}_{\mathsf{id}_1}$ to $\mathcal{A}$. Revealing $\mathsf{sk}_{\mathsf{id}_1}$ to $\mathcal{A}$ is captured by the next query. This treatment is to allow $\mathcal{A}$ to obtain level-2 secret keys of the form $\mathsf{sk}_{\mathsf{id}_1^*,\mathsf{id}_2}$ with $\mathsf{id}_2 \neq \mathsf{id}_2^*$.

Definition 6. *We say that a 2-level HIBE scheme* Π *satisfies* selective-identity security*, if the advantage* $\mathsf{Adv}^{\mathtt{HIBE\text{-}sel}}_{\Pi,\mathcal{A}}(\lambda)$ *is negligible for all PPT adversaries.*

The game for *adaptive-identity* security is defined in the same way as the selective-identity security game, except that the adversary $\mathcal{A}$ chooses the challenge identity $\mathsf{ID}^* = (\mathsf{id}^*_1, \mathsf{id}^*_2)$ not at the beginning of the game, but at the time it makes the challenge query. More formally, the response to the challenge query is defined differently as follows:

Challenge Query: $\mathcal{A}$ is allowed to make this query only once. Upon a query $(\mathsf{ID}^* = (\mathsf{id}^*_1, \mathsf{id}^*_2), \mathsf{M}_0, \mathsf{M}_1)$ from $\mathcal{A}$, where it is required that the following conditions are satisfied simultaneously:

- $|\mathsf{M}_0| = |\mathsf{M}_1|$,
- $((\mathsf{id}^*_1, \mathsf{id}^*_2), *) \notin \mathtt{SKList}$,
- $\mathsf{sk}_{\mathsf{id}^*_1}$ has not been revealed to $\mathcal{A}$.

$\mathcal{C}$ picks the challenge bit $b \in \{0, 1\}$ uniformly at random, runs $\mathsf{ct}^* \leftarrow \mathsf{Encrypt}(\mathsf{PP}, \mathsf{ID}^* = (\mathsf{id}^*_1, \mathsf{id}^*_2), \mathsf{M}_b)$, and returns the challenge ciphertext ct^* to $\mathcal{A}$.

The adaptive-identity security advantage $\mathsf{Adv}^{\mathtt{HIBE\text{-}ad}}_{\Pi,\mathcal{A}}(\lambda)$ of the adversary $\mathcal{A}$ is defined analogously to that for selective-identity security.

Definition 7. *We say that a 2-level HIBE scheme* Π *satisfies* adaptive-identity security*, if the advantage* $\mathsf{Adv}^{\mathtt{HIBE\text{-}ad}}_{\Pi,\mathcal{A}}(\lambda)$ *is negligible for all PPT adversaries* $\mathcal{A}$*.*

Contents