New Constructions of Revocable Identity-Based Encryption from Multilinear Maps

Seunghwan Park* Kwangsu Lee† Dong Hoon Lee‡

Abstract

A revocation mechanism in cryptosystems for a large number of users is absolutely necessary for maintaining the security of whole systems. A revocable identity-based encryption (RIBE) provides an efficient revocation method in IBE in which a trusted authority periodically broadcasts an update key for non-revoked users and a user can decrypt a ciphertext if his private key is not revoked in the update key. Boldyreva, Goyal, and Kumar (CCS 2008) defined RIBE and proposed an RIBE scheme that uses a tree-based revocation encryption scheme to revoke users' private keys. However, this approach has an inherent limitation in that the number of private key elements and update key elements cannot be constant. In this paper, to overcome this limitation, we devise a new technique for RIBE and propose RIBE schemes with a constant number of private key elements. We achieve the following results:

- We first devise a new technique for RIBE that combines a hierarchical IBE (HIBE) scheme and a public-key broadcast encryption (PKBE) scheme by using multilinear maps. In contrast to the previous technique for RIBE, our technique uses a PKBE scheme in bilinear maps for revocation to achieve short private keys and update keys.
- Following our new technique for RIBE, we propose an RIBE scheme in three-leveled multilinear maps that combines the HIBE scheme of Boneh and Boyen (EUROCRYPT 2004) and the PKBE scheme of Boneh, Gentry, and Waters (CRYPTO 2005). The private key and update key of our scheme possess a constant number of group elements. We introduce a new complexity assumption in multilinear maps and prove the security of our scheme in the selective revocation list model.
- Next, we propose another RIBE scheme with reduced public parameters by combining the HIBE scheme of Boneh and Boyen and the PKBE scheme of Boneh, Waters, and Zhandry (CRYPTO 2014), which uses multilinear maps. Compared with our first RIBE scheme, our second RIBE scheme requires high-leveled multilinear maps since the underlying PKBE scheme is based on high-leveled multilinear maps.

Keywords: Identity-based encryption, Key revocation, Broadcast encryption, Multilinear maps.

*Korea University, Seoul, Korea. Email: sgusa@korea.ac.kr.
†Korea University, Seoul, Korea. Email: guspin@korea.ac.kr.
‡Korea University, Seoul, Korea. Email: donghlee@korea.ac.kr.

Contents

1 Introduction

Providing an efficient revocation mechanism in cryptosystems for a large number of users is very important since it can prevent a user from accessing sensitive data in cryptosystems by revoking the private key of a user when the private key is revealed or expired. In public-key encryption (PKE), which employs the public-key infrastructure (PKI), there are many studies that deal with the certificate revocation problem [1, 16, 30, 32]. In identity-based encryption (IBE) [7, 37], a natural approach for this revocation problem is that a trusted authority periodically renews a user's private key for his identity at a current time period and then a sender creates a ciphertext for both a receiver identity and a current time period. However, this approach has some problems: the trusted authority should always be online to renew the user's private keys, users should always renew their private keys regardless of whether their private keys are revoked, and a secure channel should be established between the trusted authority and a user to transmit a renewed private key.

An IBE scheme that provides an efficient revocation mechanism (RIBE) was proposed by Boldyreva, Goyal, and Kumar [3]. In RIBE, each user receives a (long-term) private key SK_{ID} for his identity ID from a trusted authority, and the trusted authority periodically broadcasts an update key $UK_{T,R}$ at a current time T by including a revoked identity set R. If a user has a private key SK_{ID} that is not revoked by the revoked identity set R of the update key $UK_{T,R}$, then he can derive his (short-term) decryption key $DK_{ID,T}$ from his private key SK_{ID} and the update key $UK_{T,R}$. This decryption key can be used to decrypt a ciphertext $CT_{ID,T}$ for a receiver identity ID and a time period T. The main advantage of this approach is that the trusted authority can be offline because the authority only need to broadcast the update key periodically. To build an RIBE scheme, Boldyreva et al. [3] used the tree-based revocation encryption scheme of Naor, Naor, and Lotspiech [31] for revocation and the ABE scheme of Sahai and Waters [34] for encryption of an identity and a time period. Other RIBE schemes also follow this design approach that uses the tree-based revocation encryption scheme for revocation [29, 35, 36]. This design approach, however, has an inherent limitation in that the number of private key elements and update key elements cannot be constant since a private key is associated with path nodes in a tree and an update key is associated with covering nodes in the tree [31]. Therefore, in this paper, we ask the following questions about RIBE: "Can we build an RIBE scheme with a constant number of private key elements and update key elements? Can we devise a new technique for efficient RIBE that is different from the previous approach?"

1.1 Our Results

In this work, we give affirmative answers to both of the above questions. That is, we first devise a new technique for RIBE that is quite different from the previous technique, and we propose two RIBE schemes with a constant number of private key elements. The following is our results:

New Techniques for Revocable IBE. Previous RIBE schemes [3, 29, 36] use IBE (or ABE) schemes for the main encryption functionality and the tree-based revocation encryption of Naor, Naor, and Lotspiech [31] for the revocation functionality. As mentioned, the inherent limitation of the tree-based revocation encryption scheme is that the number of private key elements and update key elements cannot be constant. To achieve an RIBE scheme with a constant number of private key elements and update key elements, we observe that PKBE schemes [8, 19] in bilinear groups can be directly used for delivering a partial key of IBE to non-revoked users because these broadcast schemes have short private keys and short ciphertexts. That is, the private key $SK_{ID,T}$ of a two-level HIBE scheme with an identity ID and a time period T is divided into two partial keys SK'_{ID} and SK'_{T}. A user's actual key consists of SK'_{ID} and the private key of PKBE, and a trusted authority broadcasts an update key $UK_{T,R}$ that is the encryption of SK'_T, which excludes revoked users R. If the user is not revoked, then he can derive $SK_{ID,T}$ of HIBE combining SK'_{ID} in his actual key and SK'_T in $UK_{T,R}$. However, this simple RIBE scheme is vulnerable under a simple attack–that is, if an adversary corrupts a user ID at time T', then he can obtain a partial key SK'_{ID} and a PKBE key for ID. The adversary then can decrypt a previous ciphertext $CT_{ID,T}$ such that $T < T'$ by obtaining a partial key SK'_T from $UK_{T,R}$ since the PKBE key that was obtained at time T' can still be applied to decrypt $UK_{T,R}$ at time T. To overcome the simple attack, we set the private key SK of RIBE by binding the private key of HIBE and the private key of PKBE, and set the update key UK of RIBE by binding the private key of HIBE and the ciphertext of PKBE. However, this RIBE scheme possesses another problem–a decryption key derived from a private key and an update key by performing a pairing operation cannot be used to decrypt a ciphertext since the decryption key is the result of the pairing operation in bilinear groups. To solve this new problem, we use multilinear maps that were recently proposed by Garg, Gentry, and Halevi [15]. The detailed techniques are discussed below in this section.

RIBE with Shorter Private Keys and Update Keys. We first propose an RIBE scheme with a constant number of private key elements and update key elements by applying our new technique for RIBE on the three-leveled multilinear maps. For a concrete RIBE construction, we use the PKBE scheme of Boneh, Gentry, and Waters (BGW-PKBE) [8] for revocation and the HIBE scheme of Boneh and Boyen (BB-HIBE) [5] for encryption of an identity ID and a time T. The public parameters, the private key, the update key, and the ciphertext of our RIBE scheme just consist of $O(N+\lambda)$, $O(1)$, $O(1)$, and $O(1)$ group elements, respectively. As far as we know, our RIBE scheme is the first one that achieves a constant number of private key elements and update key elements. To prove the security of our RIBE scheme, we introduce a new complexity assumption called the Multilinear Diffie-Hellman Exponent (MDHE) assumption that is a natural multilinear version of the Bilinear Diffie-Hellman Exponent (BDHE) assumption of Boneh et al. [8]. We prove the security of our scheme in the selective revocation list model, where an adversary should initially submit a challenge identity, a challenge time, and the revoked set of identities at the challenge time.

RIBE with Reduced Pubic Parameters. The number of group elements in the public parameters of our first RIBE scheme is proportional to the maximum number of users. To overcome this problem, we propose another RIBE scheme with reduced public parameters on $O(\log N)$-leveled multilinear maps by employing the PKBE scheme of Boneh, Waters, and Zhandry (BWZ-PKBE) [12]. The interesting feature of the BWZ-PKBE scheme is that the public key just consists of $O(\log N)$ group elements whereas the public key of the BGW-PKBE scheme [8] consists of $O(N)$ group elements. Additionally, the BWZ-PKBE scheme has a similar structure to the BGW-PKBE scheme except that it uses $O(\log N)$-leveled multilinear maps. Because of this structural similarity, we can build an RIBE scheme based on the BWZ-PKBE scheme by following our new technique for RIBE. We prove the security of our second RIBE scheme in the selective revocation list model by using the compressed MDHE (cMDHE) assumption. Although the number of group elements in public parameters is reduced, our second RIBE scheme is not a truly identity-based one since the maximum size of the receiver set is restricted to being polynomial in the BWZ-PKBE scheme. A detailed comparison between our RIBE schemes and other RIBE schemes is given in Table 1. Note that the bit size of private keys and update keys in our RIBE schemes is not constant since the bit size of group elements in leveled multilinear maps is not constant [24].

1.2 Our Technique

To devise an RIBE scheme with a constant number of private key elements and update key elements, we use the BGW-PKBE scheme [8] for revocation instead of using the revocation encryption of Naor, Naor,

Table 1: Comparison of revocable identity-based encryption schemes

Scheme	PP Size	SK Size	UK Size	Model	Maps	Assumption
BF [7]	$O(1)$	$O(1)$	$O(N-r)$	Full	BLM	RO, BDH
BGK [3]	$O(1)$	$O(\log N)$	$O(r\log(N/r))$	Selective	BLM	DBDH
LV [29]	$O(\lambda)$	$O(\log N)$	$O(r\log(N/r))$	Full	BLM	DBDH
SE [36]	$O(\lambda)$	$O(\log N)$	$O(r\log(N/r))$	Full	BLM	DBDH
LLP [27]	$O(1)$	$O(\log^{1.5} N)$	$O(r)$	Full	BLM	Static
Our-1	$O(N+\lambda)$	$O(1)$	$O(1)$	SelectiveRL	MLM	MDHE
Our-2	$O(\log N+\lambda)$	$O(1)$	$O(1)$	SelectiveRL	MLM	cMDHE

Let λ be a security parameter, N be the maximum number of users, and r be the maximum number of revoked users. Sizes for public parameters (PP), private keys (SK), and update keys (UK) count the number of group elements. BLM stands for bilinear maps and MLM stands for multilinear maps. The bit sizes of elements in bilinear maps, elements in three-leveled multilinear maps, and elements in k-leveled multilinear maps are $O(\lambda)$, $O(\lambda\log^2\lambda)$, and $O(k^2\lambda\log^2(k\lambda))$, respectively.

and Lotspiech [31]. The revocation encryption of the NNL framework mainly uses a tree for broadcasting, and it is hard to provide a constant number of RIBE private key elements since the private key of the NNL framework is associated with path nodes in the tree and the update key is associated with subset covering nodes in the tree [31]. The BGW-PKBE scheme, by contrast, can provide a constant number of RIBE private key elements since the PKBE scheme has a constant number of private key elements.

For our RIBE construction, we use the BGW-PKBE scheme [8] for revocation and the two-level HIBE scheme of Boneh and Boyen [5] for encryption of an identity ID and a time period T. As mentioned before, the simple approach is vulnerable under a simple attack. To address this problem, we first set the RIBE private key as $SK_{ID} = \big(g^{\alpha^d\gamma}F(ID)^{r_1}, g^{r_1}\big)$, which is a careful combination of the PKBE private key $SK_{BE,d} = g^{\alpha^d\gamma}$ and the HIBE private key $SK_{HIBE,ID} = (g^a F(ID)^{r_1}, g^{r_1})$, where an index d is associated with the identity ID and $F(\cdot)$ is a function from identities to group elements. That is, we replace the master key part g^a of the HIBE private key component with the PKBE private key component. Next, we set the RIBE update key as $UK_{T,R} = \big((g^{\gamma}\prod_{j\in\mathcal{N}\setminus R} g^{\alpha^{N+1-j}})^{\beta}H(T)^{r_2}, g^{r_2}\big)$, which is a careful combination of the PKBE ciphertext $CT_{BE,R} = \big(g^{\beta}, (g^{\gamma}\prod_{j\in\mathcal{N}\setminus R} g^{N+1-j})^{\beta}\big)$ and the HIBE private key $SK_{HIBE,T} = \big(g^a H(T)^{r_2}, g^{r_2}\big)$ where R is a revocation set, T is an update time period, and $H(\cdot)$ is a function from times to group elements. That is, we replace the master key part g^a of the HIBE private key component with the PKBE ciphertext component. If a user with a private key SK_{ID} is not revoked in an update key $UK_{T,R}$ at a time T, then he can derive a decryption key $DK_{ID,T} = \big(g^{\alpha^{N+1}\beta}F(ID)^{r_1}H(T)^{r_2}, g^{r_1}, g^{r_2}\big)$ for his identity ID and the time T. This decryption key can be used to decrypt a ciphertext $CT_{ID,T} = \big(e(g^{\alpha^{N+1}}, g^{\beta})^s \cdot M, g^s, F(ID)^s, H(T)^s\big)$.

However, there is a major problem with this idea. That is, a session key that is derived from the ciphertext and the private key of PKBE in bilinear groups is an element in $\mathbb{G}_T$, and this session key cannot be used for pairing in bilinear groups. This means that the RIBE decryption key $DK_{ID,T}$, which is related with the session key of PKBE, cannot be used to decrypt a RIBE ciphertext $CT_{ID,T}$ since the pairing operation can no longer be applicable. To address this problem, we use three-leveled multilinear maps [15]. Note that bilinear maps correspond to two-leveled multilinear maps. In our RIBE scheme, which uses three-leveled multilinear maps, a private key SK_{ID} is in $\mathbb{G}_1$, an update key $UK_{T,R}$ is in $\mathbb{G}_1$, a decryption key $DK_{ID,T}$ is in $\mathbb{G}_2$, and a ciphertext $CT_{ID,T}$ is in $\mathbb{G}_1$. The ciphertext $CT_{ID,T}$ in $\mathbb{G}_1$ and the decryption key $DK_{ID,T}$ in $\mathbb{G}_2$

can be used to derive a session key by using a bilinear map $e_{1,2}(-,-)$, which is additionally provided by three-leveled multilinear maps. Therefore, we can build an RIBE scheme with a constant number of private key elements and update key elements from three-leveled multilinear maps. This technique also applies to the BWZ-PKBE scheme [12].

1.3 Related Work

Identity-Based Encryption and Its Extensions. IBE, introduced by Shamir [37], can solve the key management problem of PKE since it uses an identity string as a public key instead of using a random value. The first IBE scheme was proposed by Boneh and Franklin [7] by using bilinear groups, and many other IBE schemes have been proposed in bilinear maps [5, 39]. IBE also can be realized under different primitives such as quadratic residues or lattices [14, 17]. Another importance of IBE is that it has many surprising extensions such as hierarchical IBE (HIBE), attribute-based encryption (ABE), predicate encryption (PE), and functional encryption (FE). HIBE was introduced by Horwitz and Lynn [22] and it additionally provides private key delegation functionality [5, 6, 18, 40]. ABE was introduced by Sahai and Waters [34] and it can provide access controls on ciphertexts by associating a ciphertext with attributes and a private key with a policy [21, 28]. PE can provide searches on encrypted data by hiding attributes in ciphertexts [11, 23]. Recently, the concept of FE, which includes all the extensions of IBE, was introduced by Boneh, Sahai, and Waters [9], and it was shown that FE schemes for general circuits can be constructed [20].

Revocation in IBE. As mentioned, providing an efficient revocation mechanism that can revoke a user whose private key is revealed is a very important issue in cryptosystems. In PKE, which employs the public-key infrastructure (PKI), the certificate revocation problem was widely studied [1, 16, 30, 32]. In IBE, there are some works that deal with the key revocation problem [2, 3, 7, 29, 36]. We can categorize the revocation methods for IBE in the following two ways. In the first revocation method, a trusted authority periodically broadcasts a revoked user set R, and a sender creates a ciphertext by additionally including a receiver set S that excludes the revoked user set R [2]. That is, this method conceptually combines an IBE scheme with a PKBE scheme. Though this method is simple to construct and does not require a user to update his private key, the sender should check the validity of the revoked list and the sender has the responsibility for the revocation. Ideally, the sender should proceed as in any IBE scheme and encrypt a message without worrying about potential revoked users.

With the second revocation method, a sender creates a ciphertext for a receiver identity ID and a time T, and a receiver periodically updates his private key on a time T from a trusted authority if he is not revoked at the time T. That is, this method can revoke a user by preventing the user from obtaining his key components from the authority. Boneh and Franklin [7] proposed a revocable IBE scheme by representing a user's identity as $ID\|T$ with a user periodically receiving his private key at a time T by communicating with the authority. However, this RIBE scheme is impractical for a large number of users since all users should be connected to the authority to receive their private keys. To improve the efficiency of RIBE, Boldyreva, Goyal, and Kumar [3] proposed a new RIBE scheme, in which a trusted authority periodically broadcasts an update key for a time T and non-revoked users by using the revocation encryption of Naor et al. [31]. After that, many other RIBE schemes were proposed by following this design principle [27, 29, 35, 36]. The key revocation is also an important issue in ABE. Sahai et al. [33] proposed a revocable-storage ABE (RS-ABE) scheme for cloud storage by extending the idea of RIBE schemes, and Lee et al. [25] proposed an improved RS-ABE scheme and a revocable-storage PE scheme.

2 Preliminaries

In this subsection, we first define revocable identity-based encryption (RIBE) and its security model, and then we review multilinear maps and complexity assumptions for our RIBE schemes.

2.1 Revocable Identity-Based Encryption

Revocable identity-based encryption (RIBE) is an extension of identity-based encryption (IBE) in that a user with an identity ID can be revoked later if his credential is expired [3]. In RIBE, each user receives his (long-term) private key that is associated with an identity ID from a key generation center. After that, the key generation center periodically broadcasts an update key for the non-revoked set of users where the update key is associated with a time T and a revoked set R. If a user is not revoked in the update key, then he can derive his (short-term) decryption key for his identity ID and the current time T from the private key and the update key. Using the decryption key for ID and T, the user can decrypt a ciphertext for a receiver identity ID_c and a time T_c if $ID = ID_c$ and $T = T_c$. The following is the syntax of RIBE.

Definition 2.1 (Revocable IBE)**.** *A revocable IBE (RIBE) scheme that is associated with the identity space $\mathcal{I}$, the time space $\mathcal{T}$, and the message space $\mathcal{M}$, consists of seven algorithms **Setup**, **GenKey**, **UpdateKey**, **DeriveKey**, **Encrypt**, **Decrypt**, and **Revoke**, which are defined as follows:*

***Setup**$(1^\lambda, N)$: The setup algorithm takes as input a security parameter 1^λ and the maximum number of users N. It outputs a master key MK, an (empty) revocation list RL, a state ST, and public parameters PP.*

***GenKey**(ID, MK, ST, PP): The private key generation algorithm takes as input an identity $ID \in \mathcal{I}$, the master key MK, the state ST, and public parameters PP. It outputs a private key SK_{ID} for ID and an updated state ST.*

***UpdateKey**(T, RL, MK, ST, PP): The update key generation algorithm takes as input an update time $T \in \mathcal{T}$, the revocation list RL, the master key MK, the state ST, and the public parameters PP. It outputs an update key $UK_{T,R}$ for T and R where R is a revoked identity set at the time T.*

***DeriveKey**$(SK_{ID}, UK_{T,R}, PP)$: The decryption key derivation algorithm takes as input a private key SK_{ID}, an update key $UK_{T,R}$, and the public parameters PP. It outputs a decryption key $DK_{ID,T}$ or $\perp$.*

***Encrypt**(ID, T, M, PP): The encryption algorithm takes as input an identity $ID \in \mathcal{I}$, a time T, a message $M \in \mathcal{M}$, and the public parameters PP. It outputs a ciphertext $CT_{ID,T}$ for ID and T.*

***Decrypt**$(CT_{ID,T}, DK_{ID',T'}, PP)$: The decryption algorithm takes as input a ciphertext $CT_{ID,T}$, a decryption key $DK_{ID',T'}$, and the public parameters PP. It outputs an encrypted message M or $\perp$.*

***Revoke**(ID, T, RL, ST): The revocation algorithm takes as input an identity ID to be revoked and a revocation time T, a revocation list RL, and a state ST. It outputs an updated revocation list RL.*

*The correctness property of RIBE is defined as follows: For all MK, RL, ST, and PP generated by **Setup**$(1^\lambda, N)$, SK_{ID} generated by **GenKey**(ID, MK, ST, PP) for any ID, $UK_{T,R}$ generated by **UpdateKey**(T, RL, MK, ST, PP) for any T and RL, CT_{ID_c,T_c} generated by **Encrypt**(ID_c, T_c, M, PP) for any ID_c, T_c, and M, it is required that*

- *If $(ID \notin R)$, then **DeriveKey**$(SK_{ID}, UK_{T,R}, PP) = DK_{ID,T}$.*

- *If* $(ID \in R)$, *then* ***DeriveKey***$(SK_{ID}, UK_{T,R}, PP) = \perp$ *with all but negligible probability.*
- *If* $(ID_c = ID) \wedge (T_c = T)$, *then* ***Decrypt***$(CT_{ID_c,T_c}, DK_{ID,T}, PP) = M$.
- *If* $(ID_c \neq ID) \vee (T_c \neq T)$, *then* ***Decrypt***$(CT_{ID,T}, DK_{ID,T}, PP) = \perp$ *with all but negligible probability.*

The security property of RIBE was formally defined by Boldyreva, Goyal, and Kumar [3]. Recently Seo and Emura [36] refined the security model of RIBE by considering decryption key exposure attacks. In this paper, we consider the selective revocation list security model of the refined security model. In the selective revocation list security game, an adversary initially submits a challenge identity ID^*, a challenge time T^*, and a revoked identity set R^* at the time T^*, and then he can adaptively request private key, update key, and decryption key queries with restrictions. In the challenge step, the adversary submits two challenge messages M_0^*, M_1^*, and then he receives a challenge ciphertext CT^* that is an encryption of M_b^* where b is a random coin used to create the ciphertext. The adversary may continue to request private key, update key, and decryption key queries. Finally, the adversary outputs a guess for the random coin b. If the queries of the adversary satisfy the non-trivial conditions and the guess is correct, then the adversary wins the game. The following is the formal definition of the selective revocation security.

Definition 2.2 (Selective Revocation List Security)**.** *The selective revocation list security property of RIBE under chosen plaintext attacks is defined in terms of the following experiment between a challenger* $\mathcal{C}$ *and a PPT adversary* $\mathcal{A}$*:*

1. ***Init***: $\mathcal{A}$ *initially submits a challenge identity* $ID^* \in \mathcal{I}$, *a challenge time* $T^* \in \mathcal{T}$, *and a revoked identity set* $R^* \subseteq \mathcal{I}$ *at the time* T^*.
2. ***Setup***: $\mathcal{C}$ *generates a master key MK, a revocation list RL, a state ST, and public parameters PP by running* ***Setup***$(1^\lambda, N)$. *It keeps* MK, RL, ST *to itself and gives PP to* $\mathcal{A}$.
3. ***Phase 1***: $\mathcal{A}$ *adaptively requests a polynomial number of queries. These queries are processed as follows:*
   - *If this is a private key query for an identity ID, then it gives the corresponding private key* SK_{ID} *to* $\mathcal{A}$ *by running* ***GenKey***(ID, MK, ST, PP) *with the restriction: If* $ID = ID^*$, *then the revocation query for* ID^* *and* T *must be queried for some* $T \leq T^*$.
   - *If this is an update key query for a time T, then it gives the corresponding update key* $UK_{T,R}$ *to* $\mathcal{A}$ *by running* ***UpdateKey***(T, RL, MK, ST, PP) *with the restriction: If* $T = T^*$, *then the revoked identity set of RL at the time* T^* *should be equal to* R^*.
   - *If this is a decryption key query for an identity ID and a time T, then it gives the corresponding decryption key* $DK_{ID,T}$ *to* $\mathcal{A}$ *by running* ***DeriveKey***$(SK_{ID}, UK_{T,R}, PP)$ *with the restriction: The decryption key query for* ID^* *and* T^* *cannot be queried.*
   - *If this is a revocation query for an identity ID and a revocation time T, then it updates the revocation list RL by running* ***Revoke***(ID, T, RL, ST) *with the restriction: The revocation query for a time T cannot be queried if the update key query for the time T was already requested.*

   Note that $\mathcal{A}$ *is allowed to request the update key query and the revocation query in non-decreasing order of time, and an update key* $UK_{T,R}$ *implicitly includes a revoked identity set R derived from RL.*
4. ***Challenge***: $\mathcal{A}$ *submits two challenge messages* $M_0^*, M_1^* \in \mathcal{M}$ *with equal length.* $\mathcal{C}$ *flips a random coin* $b \in \{0,1\}$ *and gives the challenge ciphertext* CT^* *to* $\mathcal{A}$ *by running* ***Encrypt***(ID^*, T^*, M_b^*, PP).

5. ***Phase 2***: *$\mathcal{A}$ may continue to request a polynomial number of private keys, update keys, and decryption keys subject to the same restrictions as before.*

6. ***Guess***: *Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$, and wins the game if $b = b'$.*

The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}_{RIBE,\mathcal{A}}^{IND\text{-}sRL\text{-}CPA}(\lambda) = \big|\Pr[b = b'] - \frac{1}{2}\big|$ where the probability is taken over all the randomness of the experiment. A RIBE scheme is secure in the selective revocation list model under chosen plaintext attacks if for all PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the above experiment is negligible in the security parameter λ.

Remark 2.3. *The selective revocation list security model is weaker than the well-known selective security model since the adversary additionally submits the revoked identity set R^* in advance. However, this weaker model was already introduced by Boldyreva et al. [3] to prove the security of their revocable ABE scheme*[1].

2.2 Leveled Multilinear Maps

We define generic leveled multilinear maps that are the leveled version of the cryptographic multilinear maps introduced by Boneh and Silverberg [10]. We follow the definition of Garg, Gentry, and Halevi [15].

Definition 2.4 (Leveled Multilinear Maps)**.** *We assume the existence of a group generator G, which takes as input a security parameter λ and a positive integer k. Let $\vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_k)$ be a sequence of groups of large prime order $p > 2^{\lambda}$. In addition, we let g_i be a canonical generator of $\mathbb{G}_i$ respectively. We assume the existence of a set of bilinear maps $\{e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \to \mathbb{G}_{i+j} | i,j \geq 1; i+j \leq k\}$ that have the following properties:*

- *Bilinearity: The map $e_{i,j}$ satisfies the following relation: $e_{i,j}(g_i^a, g_j^b) = g_{i+j}^{ab} : \forall a,b \in \mathbb{Z}_p$*
- *Non-degeneracy: We have that $e_{i,j}(g_i, g_j) = g_{i+j}$ for each valid i,j.*

We say that $\vec{\mathbb{G}}$ is a multilinear group if the group operations in $\vec{\mathbb{G}}$ as well as all bilinear maps are efficiently computable. We often omit the subscripts of $e_{i,j}$ and just write e.

2.3 Complexity Assumptions

We introduce new complexity assumptions in multilinear maps. The first assumption is the multilinear version of the well-known Bilinear Diffie-Hellman Exponent (BDHE) assumption of Boneh, Gentry, and Waters [8].

Assumption 2.5 (Multilinear Diffie-Hellman Exponent, (k,N)-MDHE)**.** *Let $(p, \vec{\mathbb{G}}, \{e_{i,j} | i,j \geq 1; i+j \leq k\})$ be the description of a k-leveled multilinear group of order p. Let g_i be a generator of $\mathbb{G}_i$. The decisional (k,N)-MDHE assumption is that if the challenge tuple*

$$D = \big(g_1, g_1^{a}, g_1^{a^2}, \ldots, g_1^{a^N}, g_1^{a^{N+2}}, \ldots, g_1^{a^{2N}}, g_1^{c_1}, \ldots, g_1^{c_{k-1}}\big) \text{ and } Z$$

are given, no PPT algorithm $\mathcal{A}$ can distinguish $Z = Z_0 = g_k^{a^{N+1}\prod_{i=1}^{k-1} c_i}$ from a random element $Z = Z_1 \in \mathbb{G}_k$ with more than a negligible advantage. The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}_{\mathcal{A}}^{(k,N)\text{-}MDHE}(\lambda) = \big|\Pr[\mathcal{A}(D,Z_0) = 0] - \Pr[\mathcal{A}(D,Z_1) = 0]\big|$ where the probability is taken over random choices of $a, c_1, \ldots, c_{k-1} \in \mathbb{Z}_p$.

[1]Boldyreva et al. initially claimed that their revocable ABE scheme is secure in the selective model [3], but they later corrected it as their revocable ABE scheme is secure in the selective revocation list model [4].

For the security proof of our first RIBE scheme, we use $(3,N)$-MDHE assumption that is a specific instance of the MDHE assumption since the scheme is built on the three-leveled multilinear maps.

Assumption 2.6 (Three-Leveled Multilinear Diffie-Hellman Exponent, $(3,N)$-MDHE)**.** *Let $(p, \vec{\mathbb{G}}, e_{1,1}, e_{1,2}, e_{2,1})$ be the description of a three-leveled multilinear group of order p. Let g_i be a generator of $\mathbb{G}_i$. The decisional $(3,N)$-MDHE assumption is that if the challenge tuple*

$$D = \left(g_1, g_1^{a}, g_1^{a^2}, \ldots, g_1^{a^N}, g_1^{a^{N+2}}, \ldots, g_1^{a^{2N}}, g_1^{b}, g_1^{c}\right) \text{ and } Z$$

are given, no PPT algorithm $\mathcal{A}$ can distinguish $Z = Z_0 = g_3^{a^{N+1}bc}$ from a random element $Z = Z_1 \in \mathbb{G}_3$ with more than a negligible advantage. The advantage of $\mathcal{A}$ is defined as $\boldsymbol{Adv}_{\mathcal{A}}^{(3,N)\text{-}MDHE}(\lambda) = \big|\Pr[\mathcal{A}(D,Z_0) = 0] - \Pr[\mathcal{A}(D,Z_1) = 0]\big|$ *where the probability is taken over random choices of $a, b, c \in \mathbb{Z}_p$.*

The second assumption in multilinear maps is the compressed version of the BDHE assumption. Boneh, Waters, and Zhandry [12] introduced this compressed assumption to prove the security of their broadcast encryption in multilinear maps[2]. We slightly modify their assumption for our second RIBE scheme by adding additional one element.

Assumption 2.7 (Compressed Multilinear Diffie-Hellman Exponent, (k,n,l)-cMDHE)**.** *Let $(p, \vec{\mathbb{G}}, \{e_{i,j} | i, j \geq 1; i+j \leq k\})$ be the description of a k-leveled multilinear groups of order p where $k = 2n+l-2$. Let g_i be a generator of $\mathbb{G}_i$. The decisional (k,n,l)-cMDHE assumption is that if the challenge tuple*

$$D = \left(g_1, g_1^{a^{2^0}}, g_1^{a^{2^1}}, \ldots, g_1^{a^{2^n}}, g_l^{b}, g_{n-1}^{c}\right) \text{ and } Z$$

are given, no PPT algorithm $\mathcal{A}$ can distinguish $Z = Z_0 = g_{2n+l-2}^{a^{2^n-1}bc}$ from a random element $Z = Z_1 \in \mathbb{G}_{2n+l-2}$ with more than a negligible advantage. The advantage of $\mathcal{A}$ is defined as $\boldsymbol{Adv}_{\mathcal{A}}^{(k,n,l)\text{-}MDHE}(\lambda) = \big|\Pr[\mathcal{A}(D,Z_0) = 0] - \Pr[\mathcal{A}(D,Z_1) = 0]\big|$ *where the probability is taken over random choices of $a, b, c \in \mathbb{Z}_p$.*

We discuss the difficulty of our new assumptions in generic multilinear groups in Appendix D.

3 Revocable IBE with Shorter Keys

In this section, we propose an RIBE scheme with a constant number of private key elements and update key elements from three-leveled multilinear maps and prove its selective revocation list security. Essentially, we use the broadcast encryption of Boneh, Gentry, and Waters [8], which uses bilinear maps.

3.1 Construction

Let $\mathcal{N} = \{1, \ldots, N\}$ where N is the (polynomial) number of users. Let $\mathcal{I} = \{0,1\}^{l_1}$ be the identity space and $\mathcal{T} = \{0,1\}^{l_2}$ be the time space where $l_1 = 2\lambda$ and $l_2 = \lambda$ for a security parameter λ. Our RIBE scheme from three-leveled multilinear maps is described as follows:

RIBE.Setup$(1^\lambda, N)$**:** This algorithm takes as input a security parameter 1^λ and the maximum number N of users. It generates a 3-leveled multilinear group $\vec{\mathbb{G}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3)$ of prime order p. Let g_1, g_2, g_3 be generators of $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ respectively. Let PP_{MLM} be the description of the multilinear group with generators.

[2] In [12], Boneh et al. called their new assumption as the Multilinear Diffie-Hellman Exponent (MDHE) assumption, but it is different to our MDHE assumption.

1. It selects random elements $f_{1,0}, \{f_{1,i,j}\}_{1 \le i \le l_1, j \in \{0,1\}}, h_{1,0}, \{h_{1,i,j}\}_{1 \le i \le l_2, j \in \{0,1\}} \in \mathbb{G}_1$. Let $\vec{f}_k = (f_{k,0}, \{f_{k,i,j}\}_{1 \le i \le l_1, j \in \{0,1\}})$ and $\vec{h}_k = (h_{k,0}, \{h_{k,i,j}\}_{1 \le i \le l_2, j \in \{0,1\}})$ for a level k. Note that we can obtain $\vec{f}_2$ and $\vec{h}_2$ from $\vec{f}_1$ and $\vec{h}_1$ by performing pairing operations. We define $F_k(ID) = f_{k,0} \prod_{i=1}^{l_1} f_{k,i,ID[i]}$ and $H_k(T) = h_{k,0} \prod_{i=1}^{l_2} h_{k,i,T[i]}$ where $ID[i]$ is a bit value at the position i and $T[i]$ is a bit value at the position i.
2. Next, it selects random exponents $\alpha, \beta, \gamma \in \mathbb{Z}_p$. It outputs a master key $MK = (\alpha, \beta, \gamma)$, an empty revocation list RL, an empty state ST, and public parameters as

$$PP = \Big(PP_{MLM},\ \{g_1^{\alpha^j}\}_{1 \le j, j \ne N+1 \le 2N},\ g_1^{\beta},\ \vec{f}_1,\ \vec{h}_1,\ \Omega = g_3^{\alpha^{N+1}\beta} \Big) \in \mathbb{G}_1^{2N+2l_1+2l_2+3} \times \mathbb{G}_3.$$

RIBE.GenKey(ID, MK, ST, PP): This algorithm takes as input an identity $ID \in \mathcal{I}$, the master key MK, the state ST, and public parameters PP.

1. It first assigns an index $d \in \mathcal{N}$ that is not in ST to the identity ID, and updates the state ST by adding a tuple (ID, d) to ST.
2. Next, it selects a random exponent $r_1 \in \mathbb{Z}_p$ and outputs a private key by implicitly including ID and the index d as

$$SK_{ID} = \Big(K_0 = g_1^{\alpha^d \gamma} F_1(ID)^{-r_1},\ K_1 = g_1^{-r_1} \Big) \in \mathbb{G}_1^2.$$

RIBE.UpdateKey(T, RL, MK, ST, PP): This algorithm takes as input a time $T \in \mathcal{T}$, the revocation list RL, the master key MK, the state ST, and public parameters PP.

1. It first defines the revoked set R of user identities at the time T from RL. That is, if there exists (ID', T') such that $(ID', T') \in RL$ for any $T' \le T$, then $ID' \in R$. It defines the revoked index set $RI \subseteq \mathcal{N}$ of the revoked identity set R by using the state ST since ST contains (ID, d). It also defines the non-revoked index set $SI = \mathcal{N} \setminus RI$.
2. Next, it selects a random exponent $r_2 \in \mathbb{Z}_p$ and outputs an update key by implicitly including T, R, and the revoked index set RI as

$$UK_{T,R} = \Big(U_0 = \big(g_1^{\gamma} \prod_{j \in SI} g_1^{\alpha^{N+1-j}}\big)^{\beta} H_1(T)^{r_2},\ U_1 = g_1^{-r_2} \Big) \in \mathbb{G}_1^2.$$

RIBE.DeriveKey($SK_{ID}, UK_{T,R}, PP$): This algorithm takes as input a private key $SK_{ID} = (K_0, K_1)$ for an identity ID, an update key $UK_{T,R} = (U_0, U_1)$ for a time T and a revoked set R of identities, and the public parameters PP. If $ID \in R$, then it outputs $\perp$ since the identity ID is revoked. Otherwise, it proceeds the following steps:

1. Let d be the index of ID and RI be the revoked index set of R. Note that these are implicitly included in SK and UK respectively. It sets a non-revoked index set $SI = \mathcal{N} \setminus RI$ and derives temporal components T_0, T_1 and T_2 as

$$T_0 = e(g_1^{\alpha^d}, U_0) \cdot e\big(g_1^{\beta}, K_0 \prod_{j \in SI, j \ne d} g_1^{\alpha^{N+1-j+d}}\big)^{-1},\ T_1 = e(g_1^{\beta}, K_1),\ T_2 = e(g_1^{\alpha^d}, U_1).$$

2. Next, it chooses random exponents $r_1', r_2' \in \mathbb{Z}_p$ and re-randomizes these components as $D_0 = T_0 \cdot F_2(ID)^{r_1'} H_2(T)^{r_2'}, D_1 = T_1 \cdot g_2^{-r_1'}, D_2 = T_2 \cdot g_2^{-r_2'}$. Note that these components are formed as $D_0 = g_2^{\alpha^{N+1}\beta} F_2(ID)^{r_1''} H_2(T)^{r_2''},\ D_1 = g_2^{-r_1''},\ D_2 = g_2^{-r_2''}$ where $r_1'' = \beta r_1 + r_1'$ and $r_2'' = \alpha^d r_2 + r_2'$.

3. Finally, it outputs a decryption key as $DK_{ID,T} = \big(D_0, D_1, D_2\big) \in \mathbb{G}_2^3$.

RIBE.Encrypt(ID, T, M, PP): This algorithm takes as input an identity ID, a time T, a message M, and the public parameters PP. It first chooses a random exponent $s \in \mathbb{Z}_p$ and outputs a ciphertext by implicitly including ID and T as

$$CT_{ID,T} = \Big(C = \Omega^s \cdot M,\ C_0 = g_1^s,\ C_1 = F_1(ID)^s,\ C_2 = H_1(T)^s\Big) \in \mathbb{G}_3 \times \mathbb{G}_1^3.$$

RIBE.Decrypt($CT_{ID,T}, DK_{ID',T'}, PP$): This algorithm takes as input a ciphertext $CT_{ID,T} = (C, C_0, C_1, C_2)$, a decryption key $DK_{ID',T'} = (D_0, D_1, D_2)$, and the public parameters PP. If $(ID = ID') \wedge (T = T')$, then it outputs the encrypted message M as $M = C \cdot \big(\prod_{i=0}^{2} e_{1,2}(C_i, D_i)\big)^{-1}$. Otherwise, it outputs $\perp$.

RIBE.Revoke(ID, T, RL, ST): This algorithm takes as input an identity ID, a revocation time T, the revocation list RL, and the state ST. If $(ID, -) \notin ST$, then it outputs $\perp$ since the private key of ID was not generated. Otherwise, it adds (ID, T) to RL. It outputs the updated revocation list RL.

3.2 Correctness

Let SK_{ID} be a private key for an identity ID that is associated with an index d, and $UK_{T,R}$ be an update key for a time T and a revoked identity set R. If $ID \notin R$, then the decryption key derivation algorithm first correctly derives temporal components as

$$\begin{aligned}
T_0 &= e(g_1^{\alpha^d}, U_0) \cdot e(g_1^{\beta}, K_0 \prod_{j \in SI, j \neq d} g_1^{\alpha^{N+1-j+d}})^{-1} \\
&= e(g_1^{\alpha^d}, (g_1^{\gamma} \prod_{j \in SI} g_1^{\alpha^{N+1-j}})^{\beta} H_1(T)^{r_2}) \cdot e(g_1^{\beta}, g_1^{\alpha^d \gamma} F_1(ID)^{-r_1} \cdot \prod_{j \in SI, j \neq d} g_1^{\alpha^{N+1-j+d}})^{-1} \\
&= e(g_1^{\beta}, g_1^{\alpha^{N+1}}) \cdot e(g_1^{\beta}, F_1(ID)^{r_1}) \cdot e(g_1^{\alpha^d}, H_1(T)^{r_2}), \\
&= g_2^{\alpha^{N+1}\beta} F_2(ID)^{\beta r_1} H_2(T)^{\alpha^d r_2}, \\
T_1 &= e(g_1^{\beta}, K_1) = e(g_1^{\beta}, g_1^{-r_1}) = g_2^{-\beta r_1},\ T_2 = e(g_1^{\alpha^d}, U_1) = e(g_1^{\alpha^d}, g_1^{-r_2}) = g_2^{-\alpha^d r_2}
\end{aligned}$$

where RI is the revoked index set of R and $SI = \mathcal{N} \setminus RI$. Next, a decryption key is correctly derived from the temporal components by performing re-randomization as

$$\begin{aligned}
D_0 &= T_0 \cdot F_2(ID)^{r_1'} H_2(T)^{r_2'} = g_2^{\alpha^{N+1}\beta} F_2(ID)^{\beta r_1} H_2(T)^{\alpha^d r_2} \cdot F_2(ID)^{r_1'} H_2(T)^{r_2'} \\
&= g_2^{\alpha^{N+1}\beta} F_2(ID)^{\beta r_1 + r_1'} H_2(T)^{\alpha^d r_2 + r_2'} = g_2^{\alpha^{N+1}\beta} F_2(ID)^{r_1''} H_2(T)^{r_2''}, \\
D_1 &= T_1 \cdot g_2^{-r_1'} = g_2^{-\beta r_1 - r_1'} = g_2^{-r_1''},\ D_2 = T_2 \cdot g_2^{-r_2'} = g_2^{-\alpha^d r_2 - r_2'} = g_2^{-r_2''}
\end{aligned}$$

where $r_1'' = \beta r_1 + r_1'$ and $r_2'' = \alpha^d r_2 + r_2'$.

Let $CT_{ID,T}$ be a ciphertext for an identity ID and a time T, and $DK_{ID',T'}$ be a decryption key for an identity ID' and a time T'. If $(ID = ID') \wedge (T = T')$, then the decryption algorithm correctly outputs an encrypted message by the following equation.

$$\prod_{i=0}^{2} e(C_i, D_i) = e(g_1^s, g_2^{\alpha^{N+1}\beta} F_2(ID)^{r_1''} H_2(T)^{r_2''}) \cdot e(F_1(ID)^s, g_2^{-r_1''}) \cdot e(H_1(T)^s, g_2^{-r_2''})$$

$$= e(g_1^s, g_2^{\alpha^{N+1}\beta}) \cdot \frac{e(g_1^s, F_2(ID)^{r_1''}) \cdot e(g_1^s, H_2(T)^{r_2''})}{e(F_1(ID)^s, g_2^{r_1''}) \cdot e(H_1(T)^s, g_2^{r_2''})}$$
$$= e(g_1^s, g_2^{\alpha^{N+1}\beta}) = (g_3^{\alpha^{N+1}\beta})^s = \Omega^s.$$

3.3 Security Analysis

To prove the security of our RIBE scheme, we carefully combine the partitioning methods of the PKBE scheme of Boneh, Gentry, and Waters [8] and the HIBE scheme of Boneh and Boyen [5].

Theorem 3.1. *The above RIBE scheme is secure in the selective revocation list model under chosen plaintext attacks if the $(3,N)$-MDHE assumption holds where N is the maximum number of users in the system. That is, for any PPT adversary $\mathcal{A}$, we have that $\boldsymbol{Adv}_{RIBE,\mathcal{A}}^{IND\text{-}sRL\text{-}CPA}(\lambda) \leq \boldsymbol{Adv}_{\mathcal{B}}^{(3,N)\text{-}MDHE}(\lambda)$.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that attacks the above RIBE scheme with a non-negligible advantage. A simulator $\mathcal{B}$ that solves the MDHE assumption using $\mathcal{A}$ is given: a challenge tuple $D = \big(g_1, g_1^a, g_1^{a^2}, \ldots, g_1^{a^N}, g_1^{a^{N+2}}, \ldots, g_1^{a^{2N}}, g_1^b, g_1^c\big)$ and Z where $Z = Z_0 = g_3^{a^{N+1}bc}$ or $Z = Z_1 \in \mathbb{G}_3$. Then $\mathcal{B}$ that interacts with $\mathcal{A}$ is described as follows:

Init: $\mathcal{A}$ initially submits a challenge identity ID^*, a challenge time T^*, and a revoked identity set R^* at the time T^*. It first sets a state ST and a revocation list RL as empty one. For each $ID \in \{ID^*\} \cup R^*$, it selects an index $d \in \mathcal{N}$ such that $(-,d) \notin ST$ and adds (ID,d) to ST. Let $RI^* \subseteq \mathcal{N}$ be the revoked index set of R^* at the time T^* and SI^* be the non-revoked index set at the time T^* such that $SI^* = \mathcal{N} \setminus RI^*$.

Setup: $\mathcal{B}$ first chooses random exponents $f_0', \{f_{i,j}'\}_{1\leq i\leq l_1, j\in\{0,1\}}, h_0', \{h_{i,j}'\}_{1\leq i\leq l_2, j\in\{0,1\}}, \theta \in \mathbb{Z}_p$. It implicitly sets $\alpha = a, \beta = b, \gamma = \theta - \sum_{j\in SI^*} a^{N+1-j}$ and publishes the public parameters PP as

$$\big\{g_1^{\alpha^i} = g_1^{a^i}\big\}_{1\leq i, i\neq N+1\leq 2N},\ g_1^{\beta} = g_1^b,$$
$$\vec{f}_1 = \Big(f_{1,0} = g_1^{f_0'}\big(\prod_{i=1}^{l_1} f_{1,i,ID^*[i]}\big)^{-1},\ \big\{f_{1,i,j} = \big(g_1^{a^N}\big)^{f_{i,j}'}\big\}_{1\leq i\leq l_1, j\in\{0,1\}}\Big),$$
$$\vec{h}_1 = \Big(h_{1,0} = g_1^{h_0'}\big(\prod_{i=1}^{l_2} h_{1,i,T^*[i]}\big)^{-1},\ \big\{h_{1,i,j} = \big(g_1^b\big)^{h_{i,j}'}\big\}_{1\leq i\leq l_2, j\in\{0,1\}}\Big),$$
$$\Omega = e\big(e(g_1^{\alpha}, g_1^{\alpha^N}), g_1^b\big) = g_3^{\alpha^{N+1}b}.$$

For notational simplicity, we define $\Delta ID = \sum_{i=1}^{l_1}(f_{i,ID[i]}' - f_{i,ID^*[i]}')$ and $\Delta T = \sum_{i=1}^{l_2}(h_{i,T[i]}' - h_{i,T^*[i]}')$. We have $\Delta ID \not\equiv 0 \mod p$ except with negligible probability if $ID \neq ID^*$ since there exists at least one index i such that $f_{i,ID[i]}' \neq f_{i,ID^*[i]}'$ and $\{f_{i,j}'\}$ are randomly chosen. We also have $\Delta T \not\equiv 0 \mod p$ except with negligible probability if $T \neq T^*$.

Phase 1: $\mathcal{A}$ adaptively requests a polynomial number of private key, update key, and decryption key queries. If this is a private key query for an identity ID, then $\mathcal{B}$ proceeds as follows:

- **Case** $ID \in R^*$: In this case, the simulator can use the partitioning method of Boneh et al. [8]. It first retrieves a tuple (ID,d) from ST where the index d is associated with ID. Note that the tuple (ID,d) exists since all identities in R^* were added to ST in the initialization step. Next, it selects a random exponent $r_1 \in \mathbb{Z}_p$ and creates a private key SK_{ID} as

$$K_0 = \big(g_1^{a^d}\big)^{\theta}\big(\prod_{j\in SI^*} g_1^{a^{N+1-j+d}}\big)^{-1} F_1(ID)^{-r_1},\ K_1 = g_1^{-r_1}.$$

- **Case** $ID \notin R^*$: In this case, we have $ID \neq ID^*$ from the restriction of Definition 2.2 and the simulator can use the partitioning method of Boneh and Boyen [5]. It first selects an index $d \in \mathcal{N}$ such that $(-,d) \notin ST$ and adds (ID,d) to ST. Next, it selects a random exponents $r_1' \in \mathbb{Z}_p$ and creates a private key SK_{ID} by implicitly setting $r_1 = -a/\Delta ID + r_1'$ as

$$K_0 = g_1^{a^d \theta} \prod_{j \in SI^* \setminus \{d\}} g_1^{-a^{N+1-j+d}} (g_1^a)^{f_0'/\Delta ID} F_1(ID)^{-r_1'},\ K_1 = (g_1^a)^{-1/\Delta ID} g_1^{r_1'}.$$

If this is an update key query for a time T, then $\mathcal{B}$ defines a revoked identity set R at the time T from RL and proceeds as follows:

- **Case** $T \neq T^*$: In this case, the simulator can use the partitioning method of Boneh and Boyen [5]. It first sets a revoked index set RI of R by using ST. It also sets $SI = \mathcal{N} \setminus RI$. Next, it selects a random exponent $r_2' \in \mathbb{Z}_p$ and creates an update key $UK_{T,R}$ by implicitly setting $r_2 = -(-\sum_{j \in SI^* \setminus SI} a^{N+1-j} + \sum_{j \in SI \setminus SI^*} a^{N+1-j})/\Delta T + r_2'$ as

$$U_0 = (g_1^b)^\theta \Big(\prod_{j \in SI^* \setminus SI} g_1^{-a^{N+1-j}} \prod_{j \in SI \setminus SI^*} g_1^{a^{N+1-j}} \Big)^{-h_0'/\Delta T} H_1(T)^{r_2'},$$

$$U_1 = \Big(\prod_{j \in SI^* \setminus SI} g_1^{-a^{N+1-j}} \prod_{j \in SI \setminus SI^*} g_1^{a^{N+1-j}} \Big)^{-1/\Delta T} g_1^{r_2'}.$$

- **Case** $T = T^*$: In this case, we have $R = R^*$ and the simulator can use the partitioning method of Boneh et al. [8]. For each $ID \in R^*$, it adds (ID, T^*) to RL if $(ID, T') \notin RL$ for any $T' \leq T^*$. Next, it selects a random exponent $r_2 \in \mathbb{Z}_p$ and creates an update key $UK_{T,R}$ as

$$U_0 = (g_1^b)^\theta H_1(T^*)^{r_2},\ U_1 = g_1^{-r_2}.$$

If this is a decryption key query for an identity ID and a time T, then $\mathcal{B}$ proceeds as follows:

- **Case** $ID \neq ID^*$: In this case, the simulator can use the partitioning method of Boneh and Boyen [5]. If $(ID,-) \notin ST$, then it selects an index $d \in \mathcal{N}$ such that $(-,d) \notin ST$ and adds (ID,d) to ST. Next, it selects random exponents $r_1', r_2 \in \mathbb{Z}_p$ and creates a decryption key $DK_{ID,T}$ by implicitly setting $r_1 = (-a/\Delta ID + r_1')b$ as

$$D_0 = e\big((g_1^a)^{-f_0'/\Delta ID} F_1(ID)^{r_1'}, g_1^b\big) \cdot H_2(T)^{r_2},\ D_1 = e\big((g_1^a)^{-1/\Delta ID} g_1^{r_1'}, g_1^b\big),\ D_2 = g_2^{r_2}.$$

- **Case** $ID = ID^*$: In this case, we have $T \neq T^*$ from the restriction of Definition 2.2, and the simulator can use the partitioning method of Boneh and Boyen [5]. It selects random exponents $r_1, r_2' \in \mathbb{Z}_p$ and creates a decryption key $DK_{ID,T}$ by implicitly setting $r_2 = (-a/\Delta T + r_2')a^N$ as

$$D_0 = e\big((g_1^a)^{-h_0'/\Delta T} H_1(T)^{r_2'}, g_1^{a^N}\big) \cdot F_2(ID)^{r_1},\ D_1 = g_2^{r_1},\ D_2 = e\big((g_1^a)^{-1/\Delta T} g_1^{r_2'}, g_1^{a^N}\big).$$

Challenge: $\mathcal{A}$ submits two challenge messages M_0^*, M_1^*. $\mathcal{B}$ chooses a random bit $\delta \in \{0,1\}$ and creates the challenge ciphertext CT^* by implicitly setting $s = c$ as

$$C = Z \cdot M_\delta^*,\ C_0 = g_1^c,\ C_1 = (g_1^c)^{f_0'},\ C_2 = (g_1^c)^{h_0'}.$$

Phase 2: Same as Phase 1.

Guess: Finally, $\mathcal{A}$ outputs a guess $\delta' \in \{0,1\}$. $\mathcal{B}$ outputs 0 if $\delta = \delta'$ or 1 otherwise.

To finish the proof, we first show that the distribution of the simulation is correct from Lemma 3.2. Let η be a random bit for Z_η. From the above simulation, we have $\Pr[\delta = \delta' | \eta = 0] = \frac{1}{2} + \mathbf{Adv}_{RIBE,\mathcal{A}}^{IND\text{-}sRL\text{-}CPA}(\lambda)$ since the distribution of the simulation is correct, and we also have $\Pr[\delta = \delta' | \eta = 1] = \frac{1}{2}$ since δ is completely hidden to $\mathcal{A}$. Therefore we can obtain the following equation

$$\begin{aligned}
&\mathbf{Adv}_{\mathcal{B}}^{(3,N)\text{-}MDHE}(\lambda) \\
&= \big|\Pr[\mathcal{B}(D,Z_0)=0] - \Pr[\mathcal{B}(D,Z_1)=0]\big| \geq \big|\Pr[\delta=\delta'|\eta=0]\big| - \big|\Pr[\delta=\delta'|\eta=1]\big| \\
&= \frac{1}{2} + \mathbf{Adv}_{RIBE,\mathcal{A}}^{IND\text{-}sRL\text{-}CPA}(\lambda) - \frac{1}{2} = \mathbf{Adv}_{RIBE,\mathcal{A}}^{IND\text{-}sRL\text{-}CPA}(\lambda).
\end{aligned}$$

This completes our proof. $\square$

Lemma 3.2. *The distribution of the above simulation is correct if $Z = Z_0$, and the challenge ciphertext is independent of δ in the adversary's view if $Z = Z_1$.*

Proof. The distribution of public parameters is correct since random exponents $f_0', \{f_{i,j}'\}, h_0', \{h_{i,j}'\}, \theta \in \mathbb{Z}_p$ are chosen. We show that the distribution of private keys is correct. In case of $ID \in R^*$, we have that the private key is correctly distributed from the setting $\gamma = \theta - \sum_{j\in SI^*} a^{N+1-j}$ as the following equation

$$K_0 = g_1^{\alpha^d \gamma} F_1(ID)^{-r_1} = g_1^{a^d(\theta - \sum_{j\in SI^*} a^{N+1-j})} F_1(ID)^{-r_1} = g_1^{a^d \theta}\Big(\prod_{j\in SI^*} g_1^{a^{N+1-j+d}}\Big)^{-1} F_1(ID)^{-r_1}.$$

In case of $ID \notin R^*$, we have that the private key is correctly distributed from the setting $\gamma = \theta - \sum_{j\in SI^*} a^{N+1-j}$ and $r_1 = -a/\Delta ID + r_1'$ as the following equation

$$\begin{aligned}
K_0 &= g_1^{\alpha^d \gamma} F_1(ID)^{-r_1} = g_1^{a^d\theta} \prod_{j\in SI^*} g^{-a^{N+1-j+d}} \Big(f_{1,0} \prod_{i=1}^{l} f_{1,i,ID[i]}\Big)^{-r_1} \\
&= g_1^{a^d\theta} \prod_{j\in SI^*\setminus\{d\}} g_1^{-a^{N+1-j+d}} \cdot g_1^{-a^{N+1}} \big(g_1^{f_0'} g_1^{a^N \Delta ID}\big)^{a/\Delta ID - r_1'} \\
&= g_1^{a^d\theta} \prod_{j\in SI^*\setminus\{d\}} g_1^{-a^{N+1-j+d}} (g_1^a)^{f_0'/\Delta ID} F_1(ID)^{-r_1'}, \\
K_1 &= g_1^{r_1} = (g_1^a)^{-1/\Delta ID} g_1^{r_1'}.
\end{aligned}$$

Next, we show that the distribution of update keys is correct. In case of $T \neq T^*$, we have that the update key is correctly distributed from the setting $\gamma = \theta - \sum_{j\in SI^*} a^{N+1-j}$ and $r_2 = -(-\sum_{j\in SI^*\setminus SI} a^{N+1-j} + \sum_{j\in SI\setminus SI^*} a^{N+1-j})/\Delta T + r_2'$ as the following equation

$$\begin{aligned}
U_0 &= \Big(g_1^\gamma \prod_{j\in SI} g_1^{\alpha^{N+1-j}}\Big)^\beta H_1(T)^{r_2} = \Big(g_1^\theta \Big(\prod_{j\in SI^*} g_1^{a^{N+1-j}}\Big)^{-1} \prod_{j\in SI} g_1^{a^{N+1-j}}\Big)^b \Big(h_{1,0} \prod_{i=1}^{t} h_{1,i,T[i]}\Big)^{r_2} \\
&= (g_1^b)^\theta \Big(\prod_{j\in SI^*\setminus SI} g_1^{-a^{N+1-j}} \prod_{j\in SI\setminus SI^*} g_1^{a^{N+1-j}}\Big)^b \big(g_1^{h_0'} g_1^{b\Delta T}\big)^{-(-\sum_{j\in SI^*\setminus SI} a^{N+1-j} + \sum_{j\in SI\setminus SI^*} a^{N+1-j})/\Delta T + r_2'} \\
&= (g_1^b)^\theta \Big(\prod_{j\in SI^*\setminus SI} g_1^{-a^{N+1-j}} \prod_{j\in SI\setminus SI^*} g_1^{a^{N+1-j}}\Big)^{-h_0'/\Delta T} H_1(T)^{r_2'},
\end{aligned}$$

$$U_1 = g_1^{r_2} = \Big(\prod_{j \in SI^* \setminus SI} g_1^{-a^{N+1-j}} \prod_{j \in SI \setminus SI^*} g_1^{a^{N+1-j}} \Big)^{-1/\Delta T} g_1^{r_2'}.$$

In case of $T = T^*$, we have that the update key is correctly distributed from the setting $\gamma = \theta - \sum_{j \in SI^*} a^{N+1-j}$ as the following equation

$$\begin{aligned} U_0 &= \big(g_1^{\gamma} \prod_{j \in SI^*} g_1^{\alpha^{N+1-j}}\big)^{\beta} \cdot H_1(T^*)^{r_2} = \Big(g_1^{\theta} \big(\prod_{j \in SI^*} g_1^{a^{N+1-j}}\big)^{-1} \cdot \prod_{j \in SI^*} g_1^{a^{N+1-j}}\Big)^{b} H_1(T^*)^{r_2} \\ &= (g_1^b)^{\theta} H_1(T^*)^{r_2}. \end{aligned}$$

We show that the distribution of decryption keys is correct. In case of $ID \neq ID^*$, the decryption key is correctly distributed from the setting $\log_{g_2} F_2(ID) = \alpha^N \Delta ID$ and $r_1 = (-\alpha/\Delta ID + r_1')b$ as the following equation

$$\begin{aligned} D_0 &= g_2^{\alpha^{N+1}\beta} F_2(ID)^{r_1} H_2(T)^{r_2} = g_2^{a^{N+1}b} \big(f_{2,0} \prod_{i=1}^{l} f_{2,i,ID[i]}\big)^{(-a/\Delta ID + r_1')b} H_2(T)^{r_2} \\ &= e\big(g_1^{a^{N+1}} \big(g_1^{f_0'} g_1^{a^N \Delta ID}\big)^{-a/\Delta ID + r_1'}, g_1^b\big) \cdot H_2(T)^{r_2} = e\big((g_1^a)^{-f_0'/\Delta ID} F_1(ID)^{r_1'}, g_1^b\big) \cdot H_2(T)^{r_2}, \\ D_1 &= g_2^{r_1} = e(g_1, g_1)^{(-a/\Delta ID)b} = e\big((g_1^a)^{-1/\Delta ID} g_1^{r_1'}, g_1^b\big). \end{aligned}$$

In case of $ID = ID^*$, the decryption key is correctly distributed from the setting $\log_{g_2} H_2(T) = b\Delta T$ and $r_2 = (-a/\Delta T + r_2')a^N$ as the following equation

$$\begin{aligned} D_0 &= g_2^{\alpha^{N+1}\beta} F_2(ID)^{r_1} H_2(T)^{r_2} = g_2^{a^{N+1}b} F_2(ID)^{r_1} (u_{2,2}^T h_{2,2})^{(-a/\Delta T + r_2')a^N} \\ &= e\big(g_1^{ab} \big(g_1^{b\Delta T} g_1^{h_2'}\big)^{-a/\Delta T + r_2'}, g_1^{a^N}\big) \cdot F_2(ID)^{r_1} = e\big((g_1^a)^{-h_0'/\Delta T} H_1(T)^{r_2'}, g_1^{a^N}\big) \cdot F_2(ID)^{r_1}, \\ D_2 &= g_2^{r_2} = e(g_1, g_1)^{(-a/\Delta T + r_2')a^N} = e((g_1^a)^{-1/\Delta T} g_1^{r_2'}, g_1^{a^N}). \end{aligned}$$

Finally, we show that the distribution of the challenge ciphertext is correct. If $Z = Z_0 = g_3^{a^{N+1}bc}$ is given, then the challenge ciphertext is correctly distributed as the following equation

$$\begin{aligned} C &= \Omega^s \cdot M_{\delta}^* = g_3^{a^{N+1}bs} \cdot M_{\delta}^* = Z_0 \cdot M_{\delta}^*, \ C_0 = g_1^s = g_1^c, \\ C_1 &= \big(g_1^{f_0'} \prod_{i=1}^{l} f_{1,i,ID^*[i]} f_{1,i,ID^*[i]}^{-1}\big)^c = (g_1^c)^{f_0'}, \ C_2 = \big(g_1^{h_0'} \prod_{i=1}^{t} h_{1,i,T^*[i]} h_{1,i,T^*[i]}^{-1}\big)^c = (g_1^c)^{h_0'}. \end{aligned}$$

Otherwise, the component C of the challenge ciphertext is independent of δ in the $\mathcal{A}$'s view since Z_1 is a random element in $\mathbb{G}_3$. This completes our proof. $\square$

3.4 Discussions

Graded Encoding Systems. The candidate multilinear maps of Garg, Gentry, and Halevi [15] are different from the leveled multilinear maps in Section 2.2. The main difference is that the encoding of a group element is randomized in the GGH framework whereas the encoding is deterministic in the leveled multilinear maps. This means that it is not trivial to check whether two strings encode the same element. Thus, additional procedures for this checking are essentially required in the GGH framework. In Appendix A, we define the graded encoding system of Garg et al. [15] and translate our RIBE scheme into the graded encoding system.

Asymptotic Analysis. The number of group elements in public parameters, a private key, an update key, and a ciphertext of our RIBE scheme is $O(N+\lambda)$, $O(1)$, $O(1)$, and $O(1)$ respectively, where N is the maximum number of users. Although our RIBE scheme provides efficient asymptotic parameters, except for the public parameters, it is not actually efficient since the underlying multilinear maps are not yet practically efficient. Let $\lambda = 80$ and $k = 3$. The multilinear maps of Garg et al. [15] have the following asymptotic parameters such that the bit size of the public parameters is $O(k^3\lambda^5 \log(k\lambda)) \approx 7.1 * 10^{11}$ and the bit size of group elements is $O(k^2\lambda^3) \approx 4.6 * 10^6$. To improve the efficiency, we may use the multilinear maps of Langlois et al. [24] such that the bit size of the public parameters is $O(k^3\lambda \log^2(k\lambda)) \approx 1.8 * 10^5$ and the bit size of the group elements is $O(k^2\lambda \log^2(k\lambda)) \approx 4.6 * 10^4$. Note that the bit size of the group elements in bilinear groups is 160.

Parameter Trade-offs. In our RIBE scheme, the number of group elements in public parameters is proportional to the maximum number of users N and the security parameter λ. To reduce the size of the public parameters, we can use the parallel construction technique of PKBE [8] by increasing the size of the update keys since some elements in public parameters can be moved into an update key. This RIBE scheme with parameter trade-offs is described in Appendix C.

Chosen-Ciphertext Security. Security against chosen-ciphertext attacks (CCA security) is similar to security against chosen-plaintext attacks (CPA security) except that an adversary can request a ciphertext decryption query. To provide CCA security, we can use the general transformation of Canetti, Halevi, and Katz [13] since the structure of our RIBE scheme is similar to that of the BB-HIBE scheme [5]. That is, we can modify our RIBE scheme to support three-level hierarchies by providing additional elements, and then the modified RIBE scheme is easily converted to a CCA-secure RIBE scheme since this tree-level HIBE scheme with CPA security is converted to a two-level HIBE scheme with CCA security.

Achieving Full Security. Our RIBE scheme is only secure in the selective revocation list model since the underlying BGW-PKBE scheme [8] only provides static security. If we are willing to use complexity leveraging arguments, then it can be adaptively secure by loosing an exponential factor in the security reduction. Alternatively, we may try to use other PKBE schemes that are adaptively secure [19, 26], but it is not yet clear how to combine the schemes and prove their security in multilinear maps.

4 Revocable IBE with Shorter Parameters

In this section, we propose an RIBE scheme with short public parameters and short keys from multilinear maps and prove its selective revocation list security. To achieve shorter size of public parameters, we use the BWZ-PKBE scheme [12] that uses multilinear maps since it has short public parameters and the structure of it is almost similar to that of the BGW-PKBE scheme [8].

4.1 Construction

We set $N = 2^n - 2$ for some integer n. Note that N should be polynomial in the security parameter λ. Let $\mathcal{N} = \{1, \ldots, N\}$. Let $\mathcal{I} = \{0,1\}^{l_1}$ be the identity space and $\mathcal{T} = \{0,1\}^{l_2}$ be the time space where $l_1 = 2\lambda$ and $l_2 = \lambda$. We suppose that an index d that is assigned to an identity ID has a Hamming weight l. Our RIBE scheme from $2n+l-2$-leveled multilinear maps is described as follows:

RIBE.Setup($1^\lambda, N$): This algorithm takes as input a security parameter 1^λ and the maximum number N of users. It generates a $2n+l-2$-leveled multilinear group $\vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_{2n+l-2})$ of prime order p.

Let g_i be generators of $\mathbb{G}_i$ respectively. Let PP_{MLM} be the description of the multilinear group with generators.

1. It selects random elements $f_{n-1,0}, \{f_{n-1,i,j}\}_{1\leq i\leq l_1, j\in\{0,1\}}, h_{n-1,0}, \{h_{n-1,i,j}\}_{1\leq i\leq l_2, j\in\{0,1\}} \in \mathbb{G}_{n-1}$. Let $\vec{f}_k = \big(f_{k,0}, \{f_{k,i,j}\}_{1\leq i\leq l_1, j\in\{0,1\}}\big)$ and $\vec{h}_k = \big(h_{k,0}, \{h_{k,i,j}\}_{1\leq i\leq l_2, j\in\{0,1\}}\big)$ for a level $k \geq n-1$. Note that we can obtain $\vec{f}_k$ and $\vec{h}_k$ from $\vec{f}_{n-1}$ and $\vec{h}_{n-1}$ by performing pairing operations. We define $F_k(ID) = f_{k,0}\prod_{i=1}^{l_1} f_{k,i,ID[i]}$ and $H_k(T) = h_{k,0}\prod_{i=1}^{l_2} h_{k,i,T[i]}$ where $ID[i]$ is a bit value at the position i and $T[i]$ is a bit value at the position i.
2. Next, it selects random exponents $\alpha,\beta,\gamma \in \mathbb{Z}_p$. It outputs a master key $MK = (\alpha,\beta,\gamma)$, an empty revocation list RL, an empty state ST, and public parameters as

$$PP = \Big(PP_{MLM},\ \{g_1^{\alpha^{2^i}}\}_{0\leq i\leq n},\ g_l^{\beta},\ \vec{f}_{n-1},\ \vec{h}_{n-1},\ \Omega = g_{2n+l-2}^{\alpha^{2^n-1}\beta}\Big)$$
$$\in \mathbb{G}_1^{n+1} \times \mathbb{G}_l \times \mathbb{G}_{n-1}^{2l_1+2l_2+2} \times \mathbb{G}_{2n+l-2}.$$

RIBE.GenKey(ID, MK, ST, PP)**:** This algorithm takes as input an identity $ID \in \mathcal{I}$, the master key MK, the state ST, and public parameters PP.

1. It first assigns an index $d \in \{0,1\}^n$ of Hamming weight l that is not in ST to the identity ID and updates the state ST by adding a tuple (ID,d) to ST.
2. It computes $g_{n-1}^{\alpha^d}$ by performing multiplications and pairing operations on the elements that are given in PP.
3. Next, it selects a random exponent $r_1 \in \mathbb{Z}_p$ and outputs a private key by implicitly including ID and the index d as

$$SK_{ID} = \Big(K_0 = g_{n-1}^{\alpha^d\gamma} F_{n-1}(ID)^{-r_1},\ K_1 = g_{n-1}^{-r_1}\Big) \in \mathbb{G}_{n-1}^2.$$

RIBE.UpdateKey(T, RL, MK, ST, PP)**:** This algorithm takes as input a time $T \in \mathcal{T}$, the revocation list RL, the master key MK, the state ST, and public parameters PP.

1. It first defines the revoked set R of user identities at the time T from RL. That is, if there exists (ID',T') such that $(ID',T') \in RL$ for any $T' \leq T$, then $ID' \in R$. It defines the revoked index set $RI \subseteq \mathcal{N}$ of the revoked identity set R by using the state ST since ST contains (ID,d). It also defines the non-revoked index set $SI = \mathcal{N} \setminus RI$. Note that $|SI|$ is polynomial since N is polynomial.
2. It computes $\{g_{n-1}^{\alpha^{2^n-1-j}}\}_{j\in SI}$ by performing multiplications and pairing operations on the elements that are given in PP.
3. Next, it selects a random exponent $r_2 \in \mathbb{Z}_p$ and outputs an update key by implicitly including T, R, and the revoked index set RI as

$$UK_{T,R} = \Big(U_0 = \big(g_{n-1}^{\gamma} \prod_{j\in SI} g_{n-1}^{\alpha^{2^n-1-j}}\big)^{\beta} H_{n-1}(T)^{r_2},\ U_1 = g_{n-1}^{-r_2}\Big) \in \mathbb{G}_{n-1}^2.$$

RIBE.DeriveKey($SK_{ID}, UK_{T,R}, PP$)**:** This algorithm takes as input a private key $SK_{ID} = (K_0, K_1)$ for an identity ID, an update key $UK_{T,R} = (U_0, U_1)$ for a time T and a revoked set R of identities, and the public parameters PP. If $ID \in R$, then it outputs $\perp$ since the identity ID is revoked. Otherwise, it proceeds the following steps:

1. Let d be the index of ID and RI be the revoked index set of R. Note that these are implicitly included in SK and UK respectively. It sets a non-revoked index set $SI = \mathcal{N} \setminus RI$. Note that $|SI|$ is polynomial since N is polynomial.
2. It computes $g_l^{\alpha^d}, \{g_{n-1}^{\alpha^{2^n-1-j+d}}\}_{j \in SI, j \neq d}$ by performing multiplications and pairing operations on the elements that are given in PP. Using these elements, it derives temporal components T_0, T_1 and T_2 as
$$T_0 = e(g_l^{\alpha^d}, U_0) \cdot e\big(g_l^{\beta}, K_0 \prod_{j \in SI, j \neq d} g_{n-1}^{\alpha^{2^n-1-j+d}}\big)^{-1},\ T_1 = e(g_l^{\beta}, K_1),\ T_2 = e(g_l^{\alpha^d}, U_1).$$
3. Next, it chooses random exponents $r'_1, r'_2 \in \mathbb{Z}_p$ and re-randomizes these components as $D_0 = T_0 \cdot F_{n+l-1}(ID)^{r'_1} H_{n+l-1}(T)^{r'_2},\ D_1 = T_1 \cdot g_{n+l-1}^{-r'_1},\ D_2 = T_2 \cdot g_{n+l-1}^{-r'_2}$. Note that these components are formed as $D_0 = g_{n+l-1}^{\alpha^{2^n-1}\beta} F_{n+l-1}(ID)^{r''_1} H_{n+l-1}(T)^{r''_2},\ D_1 = g_{n+l-1}^{-r''_1},\ D_2 = g_{n+l-1}^{-r''_2}$ where $r''_1 = \beta r_1 + r'_1$ and $r''_2 = \alpha^d r_2 + r'_2$.
4. Finally, it outputs a decryption key as $DK_{ID,T} = \big(D_0, D_1, D_2\big) \in \mathbb{G}_{n+l-1}^3$.

RIBE.Encrypt(ID, T, M, PP): This algorithm takes as input an identity ID, a time T, a message M, and the public parameters PP. It chooses a random exponent $s \in \mathbb{Z}_p$ and outputs a ciphertext by implicitly including ID and T as
$$CT_{ID,T} = \Big(C = \Omega^s \cdot M,\ C_0 = g_{n-1}^s,\ C_1 = F_{n-1}(ID)^s,\ C_2 = H_{n-1}(T)^s\Big) \in \mathbb{G}_{2n+l-2} \times \mathbb{G}_{n-1}^3.$$

RIBE.Decrypt($CT_{ID,T}, DK_{ID',T'}, PP$): This algorithm takes as input a ciphertext $CT_{ID,T} = (C, C_0, C_1, C_2)$, a decryption key $DK_{ID',T'} = (D_0, D_1, D_2)$, and the public parameters PP. If $(ID = ID') \wedge (T = T')$, then it outputs the encrypted message M as $M = C \cdot \big(\prod_{i=0}^{2} e(C_i, D_i)\big)^{-1}$. Otherwise, it outputs $\perp$.

RIBE.Revoke(ID, T, RL, ST): This algorithm takes as input an identity ID, a revocation time T, the revocation list RL, and the state ST. If $(ID, -) \notin ST$, then it outputs $\perp$ since the private key of ID was not generated. Otherwise, it adds (ID, T) to RL. It outputs the updated revocation list RL.

4.2 Correctness

We first show that some elements that are needed for the scheme can be easily computed from the elements in PP. We use the following claim of Boneh, Waters, and Zhandry.

Claim 4.1 ([12]). *Using group multiplications and pairing operations on the $g_1^{\alpha^{2^i}}$ for $i \in [0, n]$, it is possible to compute $g_l^{\alpha^j}$ for $j \in [1, 2^n - 2]$ of weight exactly l, $g_{n-1}^{\alpha^{2^n-1-j}}$ for $j \in [1, 2^n - 2]$ of weight exactly l, and $g_{n-1}^{\alpha^{2^n-1-j+u}}$ for $j, u \in [1, 2^n - 2], j \neq u$ of weight exactly l.*

Now we show that the correctness of decryption keys and the decryption algorithm. Let SK_{ID} be a private key for an identity ID that is associated with an index d, and $UK_{T,R}$ be an update key for a time T and a revoked identity set R. If $ID \notin R$, then the decryption key derivation algorithm first correctly derives temporal components as
$$\begin{aligned} T_0 &= e(g_l^{\alpha^d}, U_0) \cdot e(g_l^{\beta}, K_0 \prod_{j \in SI, j \neq d} g_{n-1}^{\alpha^{2^n-1-j+d}})^{-1} \\ &= e\big(g_l^{\alpha^d}, (g_{n-1}^{\gamma} \prod_{j \in SI} g_{n-1}^{\alpha^{2^n-1-j}})^{\beta} H_{n-1}(T)^{r_2}\big) \cdot e\big(g_l^{\beta}, g_{n-1}^{\alpha^d \gamma} F_{n-1}(ID)^{-r_1} \cdot \prod_{j \in SI, j \neq d} g_{n-1}^{\alpha^{2^n-1-j+d}}\big)^{-1} \end{aligned}$$

$$= e(g_l^{\beta}, g_{n-1}^{\alpha^{2^n-1}}) \cdot e(g_l^{\beta}, F_{n-1}(ID)^{r_1}) \cdot e(g_l^{\alpha^d}, H_{n-1}(T)^{r_2}),$$
$$= g_{n+l-1}^{\alpha^{2^n-1}\beta} F_{n+l-1}(ID)^{\beta r_1} H_{n+l-1}(T)^{\alpha^d r_2},$$
$$T_1 = e(g_l^{\beta}, K_1) = e(g_l^{\beta}, g_{n-1}^{-r_1}) = g_{n+l-1}^{-\beta r_1},\ T_2 = e(g_l^{\alpha^d}, U_1) = e(g_l^{\alpha^d}, g_{n-1}^{-r_2}) = g_{n+l-1}^{-\alpha^d r_2}$$

where RI is the revoked index set of R and $SI = \mathcal{N} \setminus RI$. Next, a decryption key is correctly derived from these components by performing re-randomization as

$$\begin{aligned}
D_0 &= T_0 \cdot F_{n+l-1}(ID)^{r_1'} H_{n+l-1}(T)^{r_2'} \\
&= g_{n+l-1}^{\alpha^{2^n-1}\beta} F_{n+l-1}(ID)^{\beta r_1} H_{n+l-1}(T)^{\alpha^d r_2} \cdot F_{n+l-1}(ID)^{r_1'} H_{n+l-1}(T)^{r_2'} \\
&= g_{n+l-1}^{\alpha^{2^n-1}\beta} F_{n+l-1}(ID)^{\beta r_1 + r_1'} H_{n+l-1}(T)^{\alpha^d r_2 + r_2'} = g_{n+l-1}^{\alpha^{2^n-1}\beta} F_{n+l-1}(ID)^{r_1''} H_{n+l-1}(T)^{r_2''}, \\
D_1 &= T_1 \cdot g_{n+l-1}^{-r_1'} = g_{n+l-1}^{-\beta r_1 - r_1'} = g_{n+l-1}^{-r_1''},\ D_2 = T_2 \cdot g_{n+l-1}^{-r_2'} = g_{n+l-1}^{-\alpha^d r_2 - r_2'} = g_2^{-r_2''}
\end{aligned}$$

where $r_1'' = \beta r_1 + r_1'$ and $r_2'' = \alpha^d r_2 + r_2'$.

Let $CT_{ID,T}$ be a ciphertext for an identity ID and a time T, and $DK_{ID',T'}$ be a decryption key for an identity ID' and a time T'. If $(ID = ID') \wedge (T = T')$, then the decryption algorithm correctly outputs an encrypted message by the following equation.

$$\begin{aligned}
\prod_{i=0}^{2} e(C_i, D_i) &= e(g_{n-1}^{s}, g_{n+l-1}^{\alpha^{2^n-1}\beta} F_{n+l-1}(ID)^{r_1''} H_{n+l-1}(T)^{r_2''}) \cdot e(F_{n-1}(ID)^s, g_{n+l-1}^{-r_1''}) \cdot e(H_{n-1}(T)^s, g_{n+l-1}^{-r_2''}) \\
&= e(g_{n-1}^{s}, g_{n+l-1}^{\alpha^{2^n-1}\beta}) \cdot \frac{e(g_{n-1}^{s}, F_{n+l-1}(ID)^{r_1''}) \cdot e(g_{n-1}^{s}, H_{n+l-1}(T)^{r_2''})}{e(F_{n-1}(ID)^s, g_{n+l-1}^{r_1''}) \cdot e(H_{n-1}(T)^s, g_{n+l-1}^{r_2''})} \\
&= e(g_{n-1}^{s}, g_{n+l-1}^{\alpha^{2^n-1}\beta}) = (g_{2n+l-2}^{\alpha^{2^n-1}\beta})^s = \Omega^s.
\end{aligned}$$

4.3 Security Analysis

The proof of security is almost similar to that in Theorem 3.1.

Theorem 4.2. *The above RIBE scheme is secure in the selective revocation list model under chosen plaintext attacks if the (k,n,l)-cMDHE assumption holds where $N = 2^n - 2$ is the maximum number of users and $k = 2n+l-2$. That is, for any PPT adversary $\mathcal{A}$, we have that $\boldsymbol{Adv}_{RIBE,\mathcal{A}}^{IND\text{-}sRL\text{-}CPA}(\lambda) \leq \boldsymbol{Adv}_{\mathcal{B}}^{(k,n,l)\text{-}cMDHE}(\lambda)$.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that attacks the above RIBE scheme with a non-negligible advantage. A simulator $\mathcal{B}$ that solves the cMDHE assumption using $\mathcal{A}$ is given: a challenge tuple $D = \left(g_1, g_1^{a^{2^0}}, g_1^{a^{2^1}}, \ldots, g_1^{a^{2^n}}, g_l^b, g_{n-1}^c\right)$ and Z where $Z = Z_0 = g_{2n+l-2}^{a^{2^n-1}bc}$ or $Z = Z_1 \in \mathbb{G}_{2n+l-2}$. Then $\mathcal{B}$ that interacts with $\mathcal{A}$ is described as follows:

Init: $\mathcal{A}$ initially submits a challenge identity ID^*, a challenge time T^*, and a revoked identity set R^* at the time T^*. It first sets a state ST and a revocation list RL as empty one. For each $ID \in \{ID^*\} \cup R^*$, it selects an index $d \in \mathcal{N}$ with Hamming weight l such that $(-,d) \notin ST$ and adds (ID,d) to ST. Let $RI^* \subseteq \mathcal{N}$ be the revoked index set of R^* at the time T^* and SI^* be the non-revoked index set at the time T^* such that $SI^* = \mathcal{N} \setminus RI^*$.

Setup: $\mathcal{B}$ first chooses random exponents $f_0', \{f_{i,j}'\}_{1 \leq i \leq l_1, j \in \{0,1\}}, h_0', \{h_{i,j}'\}_{1 \leq i \leq l_2, j \in \{0,1\}}, \theta \in \mathbb{Z}_p$. It implicitly sets $\alpha = a, \beta = b, \gamma = \theta - \sum_{j \in SI^*} a^{2^n-1-j}$ and publishes the public parameters PP as

$$\left\{g_1^{\alpha^{2^i}} = g_1^{a^{2^i}}\right\}_{0 \leq i \leq n},\ g_l^{\beta} = g_l^{b},$$

$$\vec{f}_{n-1} = \Big(f_{n-1,0} = g_{n-1}^{f'_0} \big(\prod_{i=1}^{l_1} f_{n-1,i,ID^*[i]} \big)^{-1},\ \big\{ f_{n-1,i,j} = (g_{n-1}^{a^{2^n-2}})^{f'_{i,j}} \big\}_{1 \le i \le l_1, j \in \{0,1\}} \Big),$$

$$\vec{h}_{n-1} = \Big(h_{n-1,0} = g_{n-1}^{h'_0} \big(\prod_{i=1}^{l_2} h_{n-1,i,T^*[i]} \big)^{-1},\ \big\{ h_{n-1,i,j} = (g_{n-1}^{b})^{h'_{i,j}} \big\}_{1 \le i \le l_2, j \in \{0,1\}} \Big),$$

$$\Omega = e\big(e(g_1^{\alpha}, g_{n-1}^{\alpha^{2^n-2}}), g_l^b, g_{n-2} \big) = g_{2n+l-2}^{\alpha^{2^n-1} b}.$$

For notational simplicity, we define $\Delta ID = \sum_{i=1}^{l_1} (f'_{i,ID[i]} - f'_{i,ID^*[i]})$ and $\Delta T = \sum_{i=1}^{l_2} (h'_{i,T[i]} - h'_{i,T^*[i]})$. We have $\Delta ID \not\equiv 0 \mod p$ except with negligible probability if $ID \neq ID^*$ since there exists at least one index i such that $f'_{i,ID[i]} \neq f'_{i,ID^*[i]}$ and $\{f'_{i,j}\}$ are randomly chosen. We also have $\Delta T \not\equiv 0 \mod p$ except with negligible probability if $T \neq T^*$.

Phase 1: $\mathcal{A}$ adaptively requests a polynomial number of private key, update key, and decryption key queries. If this is a private key query for an identity ID, then $\mathcal{B}$ proceeds as follows:

- **Case** $ID \in R^*$: It first retrieves a tuple (ID, d) from ST where the index d is associated with ID. Note that the tuple (ID, d) exists since all identities in R^* were added to ST in the initialization step. Next, it selects a random exponent $r_1 \in \mathbb{Z}_p$ and creates a private key SK_{ID} as

$$K_0 = \big(g_{n-1}^{a^d} \big)^{\theta} \big(\prod_{j \in SI^*} g_{n-1}^{a^{2^n-1-j+d}} \big)^{-1} F_{n-1}(ID)^{-r_1},\ K_1 = g_{n-1}^{-r_1}.$$

- **Case** $ID \notin R^*$: In this case, we have $ID \neq ID^*$ from the restriction of Definition 2.2. It first selects an index $d \in \mathcal{N}$ such that $(-, d) \notin ST$ and adds (ID, d) to ST. Next, it selects a random exponents $r'_1 \in \mathbb{Z}_p$ and creates a private key SK_{ID} by implicitly setting $r_1 = -a/\Delta ID + r'_1$ as

$$K_0 = g_{n-1}^{a^d \theta} \prod_{j \in SI^* \setminus \{d\}} g_{n-1}^{-a^{2^n-1-j+d}} (g_{n-1}^{a})^{f'_0/\Delta ID} F_{n-1}(ID)^{-r'_1},\ K_1 = (g_{n-1}^{a})^{-1/\Delta ID} g_{n-1}^{r'_1}.$$

If this is an update key query for a time T, then $\mathcal{B}$ defines a revoked identity set R at the time T from RL and proceeds as follows:

- **Case** $T \neq T^*$: It first sets a revoked index set RI of R by using ST. It also sets $SI = \mathcal{N} \setminus RI$. Next, it selects a random exponent $r'_2 \in \mathbb{Z}_p$ and creates an update key $UK_{T,R}$ by implicitly setting $r_2 = -(-\sum_{j \in SI^* \setminus SI} a^{2^n-1-j} + \sum_{j \in SI \setminus SI^*} a^{2^n-1-j})/\Delta T + r'_2$ as

$$U_0 = (g_{n-1}^{b})^{\theta} \Big(\prod_{j \in SI^* \setminus SI} g_{n-1}^{-a^{2^n-1-j}} \prod_{j \in SI \setminus SI^*} g_{n-1}^{a^{2^n-1-j}} \Big)^{-h'_0/\Delta T} H_{n-1}(T)^{r'_2},$$

$$U_1 = \Big(\prod_{j \in SI^* \setminus SI} g_{n-1}^{-a^{2^n-1-j}} \prod_{j \in SI \setminus SI^*} g_{n-1}^{a^{2^n-1-j}} \Big)^{-1/\Delta T} g_{n-1}^{r'_2}.$$

- **Case** $T = T^*$: In this case, we have $R = R^*$. For each $ID \in R^*$, it adds (ID, T^*) to RL if $(ID, T') \notin RL$ for any $T' \le T^*$. Next, it selects a random exponent $r_2 \in \mathbb{Z}_p$ and creates an update key $UK_{T,R}$ as

$$U_0 = (g_{n-1}^{b})^{\theta} H_{n-1}(T^*)^{r_2},\ U_1 = g_{n-1}^{-r_2}.$$

If this is a decryption key query for an identity ID and a time T, then $\mathcal{B}$ proceeds as follows:

- **Case** $ID \neq ID^*$: If $(ID,-) \notin ST$, then it selects an index $d \in \mathcal{N}$ such that $(-,d) \notin ST$ and adds (ID,d) to ST. Next, it selects random exponents $r_1', r_2 \in \mathbb{Z}_p$ and creates a decryption key $DK_{ID,T}$ by implicitly setting $r_1 = (-a/\Delta ID + r_1')b$ as

$$D_0 = e\big((g_{n-1}^a)^{-f_0'/\Delta ID} F_{n-1}(ID)^{r_1'}, g_l^b\big) \cdot H_{n+l-1}(T)^{r_2},$$
$$D_1 = e\big((g_{n-1}^a)^{-1/\Delta ID} g_{n-1}^{r_1'}, g_l^b\big),\ D_2 = g_{n+l-1}^{r_2}.$$

- **Case** $ID = ID^*$: In this case, we have $T \neq T^*$ from the restriction of Definition 2.2. It selects random exponents $r_1, r_2' \in \mathbb{Z}_p$ and creates a decryption key $DK_{ID,T}$ by implicitly setting $r_2 = (-a/\Delta T + r_2')a^{2^n-2}$ as

$$D_0 = e\big((g_l^a)^{-h_0'/\Delta T} H_l(T)^{r_2'}, g_{n-1}^{a^{2^n-2}}\big) \cdot F_{n+l-1}(ID)^{r_1},\ D_1 = g_{n+l-1}^{r_1},$$
$$D_2 = e\big((g_l^a)^{-1/\Delta T} g_l^{r_2'}, g_{n-1}^{a^{2^n-2}}\big).$$

  Note that it can computes $H_l(T)$ since g_l^b is given in the assumption.

Challenge: $\mathcal{A}$ submits two challenge messages M_0^*, M_1^*. $\mathcal{B}$ chooses a random bit $\delta \in \{0,1\}$ and creates the challenge ciphertext CT^* by implicitly setting $s = c$ as

$$C = Z \cdot M_\delta^*,\ C_0 = g_{n-1}^c,\ C_1 = (g_{n-1}^c)^{f_0'},\ C_2 = (g_{n-1}^c)^{h_0'}.$$

Phase 2: Same as Phase 1.

Guess: Finally, $\mathcal{A}$ outputs a guess $\delta' \in \{0,1\}$. $\mathcal{B}$ outputs 0 if $\delta = \delta'$ or 1 otherwise.

To finish the proof, we should show that the distribution of the simulation is correct. We omit the analysis of the distribution since the analysis is almost similar to that of Lemma 3.2 except that it uses multilinear maps and the Claim 4.1. This completes our proof. □

4.4 Discussions

Asymptotic Analysis. The number of group elements in public parameters, a private key, an update key, and a ciphertext of our second RIBE scheme is $O(\log N + \lambda)$, $O(1)$, $O(1)$, and $O(1)$ respectively, where N is the maximum number of users. However, our RIBE scheme requires k-leveled multilinear maps where $k \approx 2.5 \log N$ since the BWZ-PKBE scheme requires $1.5 \log N$-leveled multilinear maps [12]. If we use the improved multilinear maps of Langlois et al. [24], the bit size of the group elements in k-leveled multilinear maps is $O(k^2 \lambda \log^2(k\lambda))$. Let $\lambda = 80$ and $N = 2^{20}$. The bit size of the group elements in $2.5 \log N$-leveled multilinear maps is approximately $2.4 * 10^5$. In the BGK-RIBE scheme [3], the bit size of a private key and an update key is approximately $3.2 * 10^2$ and $1.6 * 10^6$ respectively, where the number of revoked users is $r = 2^{10}$. Therefore, our second RIBE scheme equipped with the currently best leveled multilinear maps [24] does not provide better parameters except the bit size of update keys. However, we expect that the parameters of leveled multilinear maps will be improved in the near future.

5 Conclusion

In this paper, we devised a new technique for RIBE that uses multilinear maps to combine an IBE scheme with a PKBE scheme. Following our technique, we first proposed an RIBE scheme with a constant number of private key elements and update key elements by combining the HIBE scheme of Boneh and Boyen [5] and the BGW-PKBE scheme [8], and then we proved its security in the selective revocation list model. Next, we proposed another RIBE scheme that reduces the number of public parameters from $O(N+\lambda)$ to $O(\log N+\lambda)$ group elements by using the BWZ-PKBE scheme [12], which has short public parameters. We expect that our technique will open a new direction to build an efficient RIBE scheme and its extensions.

There are many interesting unsolved problems in RIBE. The first one is to construct an RIBE scheme with short parameters and short keys that is secure in the adaptive security model instead of in the selective revocation list model. The second one is to construct a revocable HIBE (RHIBE) scheme with better parameters. RHIBE provides the private key delegation functionality and the revocation functionality for each user. The RHIBE scheme of Seo and Emura [36] has $O(l^2 \log N)$ number of private key elements and $O(r \log(N/r))$ number of update key elements where l is the depth of hierarchy, N is the maximum number of users, and r is the maximum number of revoked users. The third one is to build an RIBE scheme with a constant number of private key elements and update key elements that can handle the exponential number of users in the system. Recall that our second RIBE scheme cannot handle the exponential number of users since the size of receiver set in the BWZ-PKBE scheme is restricted to being polynomial.

Acknowledgements

The first two authors (Seunghwan Park and Kwangsu Lee) contributed equally to this work.

References

[1] William Aiello, Sachin Lodha, and Rafail Ostrovsky. Fast digital identity revocation (extended abstract). In Hugo Krawczyk, editor, *CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 137–152. Springer, 1998.

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

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

[4] Alexandra Boldyreva, Vipul Goyal, and Virendra Kumar. Identity-based encryption with efficient revocation. Cryptology ePrint Archive, Report 2012/052, 2012. `http://eprint.iacr.org/2012/052`.

[5] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.

[6] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005.

[7] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[8] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2005.

[9] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer, 2011.

[10] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2003.

[11] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Springer, 2007.

[12] Dan Boneh, Brent Waters, and Mark Zhandry. Low overhead broadcast encryption from multilinear maps. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014*, volume 8616 of *Lecture Notes in Computer Science*, pages 206–223. Springer, 2014.

[13] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2004.

[14] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *IMA Int. Conf.*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer, 2001.

[15] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013.

[16] Craig Gentry. Certificate-based encryption and the certificate revocation problem. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 272–293. Springer, 2003.

[17] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *STOC 2008*, pages 197–206. ACM, 2008.

[18] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, 2002.

[19] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 171–188. Springer, 2009.

[20] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 162–179. Springer, 2012.

[21] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 89–98. ACM, 2006.

[22] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer, 2002.

[23] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162. Springer, 2008.

[24] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. Gghlite: More efficient multilinear maps from ideal lattices. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 239–256. Springer, 2014.

[25] Kwangsu Lee, Seung Geol Choi, Dong Hoon Lee, Jong Hwan Park, and Moti Yung. Self-updatable encryption: Time constrained access control with hidden attributes and better efficiency. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013*, volume 8269 of *Lecture Notes in Computer Science*, pages 235–254. Springer, 2013.

[26] Kwangsu Lee and Dong Hoon Lee. Adaptively secure broadcast encryption under standard assumptions with better efficiency. Cryptology ePrint Archive, Report 2013/488, 2013. `http://eprint.iacr.org/2013/488`.

[27] Kwangsu Lee, Dong Hoon Lee, and Jong Hwan Park. Efficient revocable identity-based encryption via subset difference methods. Cryptology ePrint Archive, Report 2014/132, 2014. `http://eprint.iacr.org/2014/132`.

[28] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91. Springer, 2010.

[29] Benoît Libert and Damien Vergnaud. Adaptive-id secure revocable identity-based encryption. In Marc Fischlin, editor, *CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2009.

[30] Silvio Micali. Efficient certificate revocation. Technical Report MIT/LCS/TM-542b, 1996.

[31] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer, 2001.

[32] Moni Naor and Kobbi Nissim. Certificate revocation and certificate update. *IEEE Journal on Selected Areas in Communications*, 18(4):561–570, 2000.

[33] Amit Sahai, Hakan Seyalioglu, and Brent Waters. Dynamic credentials and ciphertext delegation for attribute-based encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 199–217. Springer, 2012.

[34] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2005.

[35] Jae Hong Seo and Keita Emura. Efficient delegation of key generation and revocation functionalities in identity-based encryption. In Ed Dawson, editor, *CT-RSA 2013*, volume 7779 of *Lecture Notes in Computer Science*, pages 343–358. Springer, 2013.

[36] Jae Hong Seo and Keita Emura. Revocable identity-based encryption revisited: Security model and construction. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *Lecture Notes in Computer Science*, pages 216–234. Springer, 2013.

[37] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

[38] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.

[39] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005.

[40] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer, 2009.

A Revocable IBE in Graded Encoding Systems

In this section, we translate our RIBE scheme in Section 3 into the graded encoding system of Garg, Gentry, and Halevi [15].

A.1 Graded Encoding Systems

We recall the formal definition of a k-graded encoding system and the procedures for the manipulation of this encoding in [15].

Definition A.1 (k-Graded Encoding System [15]). *A k-Graded Encoding System for a ring R is a system of sets $\mathcal{S} = \{S_i^{(\alpha)} \subset \{0,1\}^* : i \in [0,k], \alpha \in R\}$, with the following properties:*

1. *For every $i \in [0,k]$, the sets $\{S_i^{(\alpha)} : \alpha \in R\}$ are disjoint .*

2. *There are binary operations $+$ and $-$ (on $\{0,1\}^*$) such that for every $\alpha_1, \alpha_2 \in R$, every $i \in [0,k]$, and every $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_i^{(\alpha_2)}$, it holds that $u_1 + u_2 \in S_i^{(\alpha_1+\alpha_2)}$ and $u_1 - u_2 \in S_i^{(\alpha_1-\alpha_2)}$ where $\alpha_1 + \alpha_2$ and $\alpha_1 - \alpha_2$ are addition and subtraction in R.*

3. *There is an associative binary operation $\times$ (on $\{0,1\}^*$) such that for every $\alpha_1, \alpha_2 \in R$, every i_1, i_2 with $0 \le i_1 + i_2 \le k$, and every $u_1 \in S_{i_1}^{(\alpha_1)}$ and $u_2 \in S_{i_2}^{(\alpha_2)}$, it holds that $u_1 \times u_2 \in S_{i_1+i_2}^{(\alpha_1 \cdot \alpha_2)}$ where $\alpha_1 \cdot \alpha_2$ is multiplication in R.*

The k-graded encoding system for a ring R includes a system for sets $\mathcal{S} = \{S_i^{(\alpha)} \subset \{0,1\}^* : i \in [0,k], \alpha \in R\}$. The set $S_i^{(\alpha)}$ consists of the "level-i encodings of α". Moreover, the system is equipped with efficient procedures.

Definition A.2 (Efficient Procedures for a k-Graded Encoding System [15]). *A k-Graded Encoding System (see above) consists of the following efficient procedures:*

Instance Generation. *The randomized $\textbf{\textit{InstGen}}(1^\lambda, 1^k)$ takes as inputs the parameters λ and k, and outputs $(\textbf{\textit{params}}, p_{zt})$, where **params** is a description of a k-Graded Encoding System as above, and p_{zt} is a zero-test parameter.*

Ring Sampler. *The randomized $\textbf{\textit{samp}}(\textbf{\textit{params}})$ outputs a "level-zero encoding" $a \in S_0^{(\alpha)}$ for a nearly uniform element $\alpha \in_R R$. Note that the encoding a does not need to be uniform in $S_0^{(\alpha)}$.*

Encoding. *The (possibly randomized) $\textbf{\textit{enc}}(\textbf{\textit{params}}, a)$ takes as input a level-zero encoding $a \in S_0^{(\alpha)}$ for some $\alpha \in R$, and outputs the level-one encoding $u \in S_1^{(\alpha)}$ for the same α.*

Re-Randomization. *The randomized $\textbf{\textit{rerand}}(\textbf{\textit{params}}, i, u)$ re-randomizes encodings relative to the same level i, Specifically, given an encoding $u \in S_i^{(\alpha)}$, it outputs another encoding $u' \in S_i^{(\alpha)}$. Moreover for any two $u_1, u_2 \in S_i^{(\alpha)}$, the output distributions of $\textbf{\textit{rerand}}(\textbf{\textit{params}}, i, u_1)$ and $\textbf{\textit{rerand}}(\textbf{\textit{params}}, i, u_2)$ are nearly the same.*

Addition and negation. *Given **params** and two encodings relative to the same level, $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_i^{(\alpha_2)}$, we have $\textbf{\textit{add}}(\textbf{\textit{params}}, u_1, u_2) \in S_i^{(\alpha_1+\alpha_2)}$ and $\textbf{\textit{neg}}(\textbf{\textit{params}}, u_1) \in S_i^{(-\alpha_1)}$. Below we write $u_1 + u_2$ and $-u_1$ as a shorthand for applying these procedures.*

Multiplication. *For $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_j^{(\alpha_2)}$, we have $\textbf{\textit{mul}}(\textbf{\textit{params}}, u_1, u_2) \in S_{i+j}^{(\alpha_1 \cdot \alpha_2)}$. Below we write $u_1 \cdot u_2$ as a shorthand for applying this procedure.*

Zero-test. *The procedure $\textbf{isZero}(\textbf{\textit{params}}, p_{zt}, u)$ outputs 1 if $u \in S_k^{(\alpha)}$ and 0 otherwise.*

Extraction. *The procedure extracts a random function of ring elements from their level-k encoding. Namely $\textbf{\textit{ext}}(\textbf{\textit{params}}, p_{zt}, u)$ outputs $s \in \{0,1\}^\lambda$, such that:*

1. *For any $\alpha \in R$ and two $u_1, u_2 \in S_k^{(\alpha)}$, $\textbf{\textit{ext}}(\textbf{\textit{params}}, p_{zt}, u_1) = \textbf{\textit{ext}}(\textbf{\textit{params}}, p_{zt}, u_2)$.*
2. *The distribution $\{\textbf{\textit{ext}}(\textbf{\textit{params}}, p_{zt}, u) : \alpha \in_R R, u \in S_\kappa^{(\alpha)}\}$ is nearly uniform over $\{0,1\}^\lambda$.*

For notational simplicity, we omit the repeated **params** arguments that are passed to input arguments in all algorithms. For instance, we write $a = \textbf{samp}()$ instead of $a = \textbf{samp}(\textbf{params})$.

A.2 Construction

Let $\mathcal{N} = \{1, \dots, N\}$, $\mathcal{I} = \{0,1\}^{l_1}$, and $\mathcal{T} = \{0,1\}^{l_2}$. Our RIBE scheme in the three-graded encoding system is described as follows:

RIBE.Setup($1^\lambda, N$): This algorithm obtains $(\mathbf{params}, p_{zt})$ by running $\mathbf{InstGen}(1^\lambda, 1^3)$. Note that **params** includes a level 1 encoding of 1, which is denoted as g_1. It chooses random encodings $f'_0, \{f'_{i,j}\}_{1 \le i \le l_1, j \in \{0,1\}}$, $h'_0, \{h'_{i,j}\}_{1 \le i \le l_2, j \in \{0,1\}}$ by freshly calling $\mathbf{samp}()$ and sets

$$f_{1,0} = \mathbf{rerand}(1, \mathbf{enc}(1, f'_0)),\ \{f_{1,i,j} = \mathbf{rerand}(1, \mathbf{enc}(1, f'_{i,j}))\}_{1 \le i \le l_1, j \in \{0,1\}},$$
$$h_{1,0} = \mathbf{rerand}(1, \mathbf{enc}(1, h'_0)),\ \{h_{1,i,j} = \mathbf{rerand}(1, \mathbf{enc}(1, h'_{i,j}))\}_{1 \le i \le l_2, j \in \{0,1\}}.$$

Let $\vec{f}_k = (f_{k,0}, \{f_{k,i,j}\}_{1 \le i \le l_1, j \in \{0,1\}})$ and $\vec{h}_k = (h_{k,0}, \{h_{k,i,j}\}_{1 \le i \le l_2, j \in \{0,1\}})$ for a level k. Note that we can obtain $\vec{f}_2$ and $\vec{h}_2$ from $\vec{f}_1$ and $\vec{h}_1$ by performing pairing operations. Next, it chooses random encodings α, β, γ by freshly calling $\mathbf{samp}()$. It outputs a master key $MK = (\alpha, \beta, \gamma)$, an empty revocation list RL, an empty state ST, and public parameters as

$$PP = \Big((\mathbf{params}, p_{zt}),\ \{A_j = \mathbf{rerand}(1, \mathbf{enc}(1, \alpha^j))\}_{1 \le j, j \ne N+1 \le 2N},$$
$$B = \mathbf{rerand}(1, \mathbf{enc}(1, \beta)),\ \vec{f}_1, \vec{h}_1,\ \Omega = \mathbf{rerand}(3, \mathbf{enc}(3, \alpha^{N+1}\beta))\Big).$$

RIBE.GenKey(ID, MK, ST, PP): This algorithm first assigns an index $d \in \mathcal{N}$ that is not in ST to the identity ID, and updates the state ST by adding a tuple (ID, d) to ST. Next, it chooses a random encoding r_1 by calling $\mathbf{samp}()$ and outputs a private key by implicitly including ID and d as

$$SK_{ID} = \Big(K_0 = \mathbf{rerand}(1, \mathbf{enc}(1, \alpha^d \cdot \gamma) + (f_{1,0} + \sum_{i=1}^{l_1} f_{1,i,ID[i]}) \cdot (-r_1)),\ K_1 = \mathbf{rerand}(1, \mathbf{enc}(1, -r_1))\Big).$$

RIBE.UpdateKey(T, RL, MK, ST, PP): This algorithm defines the revoked set R, the revoked index set RI, and the non-revoked index set SI as the same as in Section 3.1. It chooses a random encoding r_2 by calling $\mathbf{samp}()$ and outputs an update key by implicitly including T, R, and RI as

$$UK_{T,R} = \Big(U_0 = \mathbf{rerand}(1, \mathbf{enc}(1, (\gamma + \sum_{j \in SI} \alpha^{N+1-j}) \cdot \beta) + (h_{1,0} + \sum_{i=1}^{l_2} h_{1,i,T[i]}) \cdot r_2)),$$
$$U_1 = \mathbf{rerand}(1, \mathbf{enc}(1, -r_2))\Big).$$

RIBE.DeriveKey($SK_{ID}, UK_{T,R}, PP$): Let $SK_{ID} = (K_0, K_1)$ and $UK_{T,R} = (U_0, U_1)$. If $ID \in R$, then it outputs $\perp$. Otherwise, it proceeds the following steps: Let d be the index of ID and RI be the revoked index set of R. It sets a non-revoked index set $SI = \mathcal{N} \setminus RI$ and derives components T_0, T_1 and T_2 as

$$T_0 = \mathbf{rerand}(2, (A_d \cdot U_0 - B \cdot (K_0 + \prod_{j \in S, j \ne d} A_{N+1-j+d}))),$$
$$T_1 = \mathbf{rerand}(2, B \cdot K_1),\ T_2 = \mathbf{rerand}(2, A_d \cdot U_1).$$

Next, it selects random encodings r'_1, r'_2 by freshly calling **samp**() and re-randomizes the temporal components as

$$D_0 = \mathbf{rerand}(2, T_0 + (f_{2,0} + \sum_{i=1}^{l_1} f_{2,i,ID[i]}) \cdot r'_1 + (h_{2,0} + \sum_{i=1}^{l_2} h_{2,i,T[i]}) \cdot r'_2),$$
$$D_1 = \mathbf{rerand}(2, T_1 + \mathbf{enc}(2, -r'_1)),\ D_2 = \mathbf{rerand}(2, T_2 + \mathbf{enc}(2, -r'_2)).$$

Finally, it outputs a decryption key by implicitly including ID and T as $DK_{ID,T} = \big(D_0, D_1, D_2\big)$.

RIBE.Encrypt(ID, T, M, PP**):** This algorithm first chooses a random encoding s by calling **samp**(). If $M = 0$, it sets $C = \mathbf{rerand}(3, \Omega \cdot s)$. Otherwise, it sets $C = \mathbf{rerand}(3, \mathbf{enc}(3, \mathbf{samp}()))$. It outputs a ciphertext by implicitly including ID and T as

$$CT_{ID,T} = \Big(C,\ C_0 = \mathbf{rerand}(1, \mathbf{enc}(1, s)),\ C_1 = \mathbf{rerand}(1, (f_{1,0} + \sum_{i=1}^{l_1} f_{1,i,ID[i]}) \cdot s),$$
$$C_2 = \mathbf{rerand}(1, (h_{1,0} + \sum_{i=1}^{l_2} h_{1,i,T[i]}) \cdot s)\Big).$$

RIBE.Decrypt($CT_{ID,T}, DK_{ID',T'}, PP$**):** Let $CT_{ID,T} = (C, C_0, C_1, C_2)$ and $DK_{ID',T'} = (D_0, D_1, D_2)$. If $(ID = ID') \wedge (T = T')$, then it computes $C' = C_1 \cdot D_1 + C_2 \cdot D_2$ and outputs $M = 1$ if $C = C'$ by using **isZero**$(p_{zt}, C - C')$ and $M = 0$ otherwise. Otherwise, it outputs $\perp$.

RIBE.Revoke(ID, T, RL, ST**):** This algorithm is the same as that of Section 3.1.

Remark A.3. *Although we can translate our RIBE scheme in Section 3 into the GGH framework, we cannot directly translate the security proof in Section 3 into the GGH framework since a level-zero encoding is defined for a ring R in the GGH framework instead of $\mathbb{Z}_p$. In Appendix B, we show that an RIBE scheme for small universe can be proven in the GGH framework.*

B Revocable IBE for Small Universe

In this section, we propose an RIBE scheme form small universe and prove its selective revocation list security.

B.1 Construction

Let $\mathcal{N} = \{1, \ldots, N\}$, $\mathcal{I} = \{ID_1, \ldots, ID_{n_1}\}$, and $\mathcal{T} = \{T_1, \ldots, T_{n_2}\}$. Let ρ_1 be a mapping from identity space $\mathcal{I}$ to integers $\{1, \ldots, n_1\}$ and ρ_2 be a mapping from time space $\mathcal{T}$ to integers $\{1, \ldots, n_2\}$. Our RIBE scheme for small universe is described as follows:

RIBE.Setup($1^\lambda, N$**):** Let PP_{MLM} be the description of a 3-leveled multilinear group of prime order p with generators g_i of $\mathbb{G}_i$. It selects random elements $v_1, \{f_{1,i}\}_{1 \le i \le n_1}, \{h_{1,i}\}_{1 \le i \le n_2} \in \mathbb{G}_1$ and sets $v_3 = e_{2,1}(g_2, v_1)$. Next, it selects random exponents $\alpha, \beta, \gamma \in \mathbb{Z}_p$. It outputs a master key $MK = (\alpha, \beta, \gamma)$, an empty revocation list RL, an empty state ST, and public parameters as

$$PP = \Big(PP_{MLM},\ v_1,\ \{g_1^{\alpha^j}, v_1^{\alpha^j}\}_{1 \le j, j \ne N+1 \le 2N},\ g_1^{\beta},\ \vec{f}_1,\ \vec{h}_1,\ \Omega = v_3^{\alpha^{N+1}\beta}\Big).$$

RIBE.GenKey(ID, MK, ST, PP): Let d be an index of ID as the same as in Section 3.1. It selects a random exponent $r_1 \in \mathbb{Z}_p$ and outputs a private key by implicitly including ID and d as

$$SK_{ID} = \Big(K_0 = v_1^{\alpha^d \gamma} f_{1,\rho_1(ID)}^{-r_1},\ K_1 = g_1^{-r_1} \Big).$$

RIBE.UpdateKey(T, RL, MK, ST, PP): This algorithm defines the revoked set R, the revoked index set RI, and the non-revoked index set SI as the same as in Section 3.1. It selects a random exponent $r_2 \in \mathbb{Z}_p$ and outputs an update key by implicitly including T, R, and the revoked index set RI as

$$UK_{T,R} = \Big(U_0 = \big(v_1^{\gamma} \prod_{j \in SI} v_1^{\alpha^{N+1-j}} \big)^{\beta} h_{1,\rho_2(T)}^{r_2},\ U_1 = g_1^{-r_2} \Big).$$

RIBE.DeriveKey($SK_{ID}, UK_{T,R}, PP$): Let $SK_{ID} = (K_0, K_1)$ and $UK_{T,R} = (U_0, U_1)$. If $ID \in R$, then it outputs $\perp$. Otherwise, it proceeds the following steps: Let d be the index of ID and RI be the revoked index set of R. It sets a non-revoked index set $SI = \mathcal{N} \setminus RI$ and derives components T_0, T_1 and T_2 as

$$T_0 = e\big(g_1^{\alpha^d}, U_0\big) \cdot e\big(g_1^{\beta}, K_0 \prod_{j \in SI, j \neq d} v_1^{\alpha^{N+1-j+d}}\big)^{-1},\ T_1 = e(g_1^{\beta}, K_1),\ T_2 = e(g_1^{\alpha^d}, U_1).$$

Next, it chooses random exponents $r_1', r_2' \in \mathbb{Z}_p$ and re-randomizes these components as $D_0 = T_0 \cdot f_{2,\rho_1(ID)}^{r_1'} h_{2,\rho_2(T)}^{r_2'}$, $D_1 = T_1 \cdot g_2^{-r_1'}$, $D_2 = T_2 \cdot g_2^{-r_2'}$. Finally, it outputs a decryption key $DK_{ID,T} = \big(D_0, D_1, D_2\big)$.

RIBE.Encrypt(ID, T, M, PP): This algorithm chooses a random exponent $s \in \mathbb{Z}_p$ and outputs a ciphertext by implicitly including ID and T as

$$CT_{ID,T} = \Big(C = \Omega^s \cdot M,\ C_0 = g_1^s,\ C_1 = f_{1,\rho_1(ID)}^s,\ C_2 = h_{1,\rho_2(T)}^s \Big).$$

RIBE.Decrypt($CT_{ID,T}, DK_{ID',T'}, PP$): It is the same as that of Section 3.1.

RIBE.Revoke(ID, T, RL, ST): It is the same as that of Section 3.1.

B.2 Security Analysis

Theorem B.1. *The above RIBE scheme for small universe is secure in the selective revocation list model under chosen plaintext attacks if the $(3,N)$-MDHE assumption holds where N is the maximum number of users in the system.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that attacks the above RIBE scheme with a non-negligible advantage. A simulator $\mathcal{B}$ that solves the MDHE assumption using $\mathcal{A}$ is given: a challenge tuple $D = (g_1, g_1^a, g_1^{a^2}, \ldots, g_1^{a^N}, g_1^{a^{N+2}}, \ldots, g_1^{a^{2N}}, g_1^b, g_1^c)$ and Z where $Z = Z_0 = g_3^{a^{N+1}bc}$ or $Z = Z_1 \in \mathbb{G}_3$. Then $\mathcal{B}$ that interacts with $\mathcal{A}$ is described as follows:

Init: $\mathcal{A}$ initially submits a challenge identity ID^*, a challenge time T^*, and a revoked identity set R^* at the time T^*. It first sets a state ST and a revocation list RL as empty one. For each $ID \in \{ID^*\} \cup R^*$, it selects an index $d \in \mathcal{N}$ such that $(-, d) \notin ST$ and adds (ID, d) to ST. Let $RI^* \subseteq \mathcal{N}$ be the revoked index set of R^* at the time T^* and SI^* be the non-revoked index set at the time T^* such that $SI^* = \mathcal{N} \setminus RI^*$.

Setup: $\mathcal{B}$ first chooses random exponents $v_1', \{f_i'\}_{1\le i\le n_1}, \{h_i'\}_{1\le i\le n_2}, \theta \in \mathbb{Z}_p$. For notational simplicity, we use $\prod f_j' = \prod_{j=1}^{n_1} f_j'$, $\prod_{j\neq k} f_j' = \prod_{1\le j\le n_1, j\neq k} f_j'$, $\prod h_j' = \prod_{j=1}^{n_2} h_j'$, and $\prod_{j\neq k} h_j' = \prod_{1\le j\le n_2, j\neq k} h_j'$. It implicitly sets $\alpha = a, \beta = b\prod h_j', \gamma = \theta - \sum_{j\in SI^*} a^{N+1-j}$ and publishes the public parameters PP as

$$\begin{aligned}
&v_1 = g_1^{v_1'\prod f_j'},\ \{g_1^{\alpha^i} = g_1^{a^i}, v_1^{\alpha^i} = \big(g_1^{a^i}\big)^{v_1'\prod f_j'}\}_{1\le i, i\neq N+1\le 2N},\ g_1^{\beta} = g_1^{b\prod h_i'},\\
&\{f_{1,i} = \big(g_1^{a^N}\big)^{\prod_{j\neq\rho_1(ID_i)} f_j'},\}_{1\le i\le n_1, i\neq\rho_1(ID^*)},\ f_{1,\rho(ID^*)} = g_1^{\prod_{j\neq\rho_1(ID^*)} f_j'},\\
&\{h_{1,i} = (g_1^b)^{\prod_{j\neq\rho_2(T_i)} h_j'},\}_{1\le i\le n_2, i\neq\rho_2(T^*)},\ h_{1,\rho(T^*)} = g_1^{\prod_{j\neq\rho_2(T^*)} h_j'},\\
&\Omega = e\big(e(g_1^{\alpha}, g_1^{\alpha^N}), g_1^b\big)^{\prod f_j'\prod h_j'} = g_3^{\alpha^{N+1} b\prod f_j'\prod h_j'}.
\end{aligned}$$

Phase 1: $\mathcal{A}$ adaptively requests a polynomial number of private key, update key, and decryption key queries. If this is a private key query for an identity ID, then $\mathcal{B}$ proceeds as follows:

- **Case** $ID \in R^*$: It first retrieves a tuple (ID, d) from ST where the index d is associated with ID. Next, it selects a random exponent $r_1 \in \mathbb{Z}_p$ and creates a private key SK_{ID} as

$$K_0 = \big(v_1^{a^d}\big)^{\theta}\Big(\prod_{j\in SI^*} v_1^{a^{N+1-j+d}}\Big)^{-1} f_{1,\rho_1(ID)}^{-r_1},\ K_1 = g_1^{-r_1}.$$

- **Case** $ID \notin R^*$: In this case, we have $ID \neq ID^*$ from the restriction of Definition 2.2. It first selects an index $d \in \mathcal{N}$ such that $(-, d) \notin ST$ and adds (ID, d) to ST. Next, it selects a random exponents $r_1' \in \mathbb{Z}_p$ and creates a private key SK_{ID} by implicitly setting $r_1 = -af_{\rho_1(ID)}' + r_1'$ as

$$K_0 = v_1^{a^d\theta} \prod_{j\in SI^*\setminus\{d\}} v_1^{-a^{N+1-j+d}} f_{1,\rho_1(ID)}^{r_1'},\ K_1 = (g_1^a)^{f_{\rho_1(ID)}'} g_1^{-r_1'}.$$

If this is an update key query for a time T, then $\mathcal{B}$ defines a revoked identity set R at the time T from RL and proceeds as follows:

- **Case** $T \neq T^*$: It first sets a revoked index set RI of R by using ST. It also sets $SI = \mathcal{N} \setminus RI$. Next, it selects a random exponent $r_2' \in \mathbb{Z}_p$ and creates an update key $UK_{T,R}$ by implicitly setting $r_2 = -(-\sum_{j\in SI^*\setminus SI} a^{N+1-j} + \sum_{j\in SI\setminus SI^*} a^{N+1-j})h_{1,\rho_2(T)}'\prod f_j' + r_2'$ as

$$U_0 = (g_1^b)^{\theta\prod f_j'} h_{1,\rho_1(T)}^{r_2'},\ U_1 = \Big(\prod_{j\in SI^*\setminus SI} g_1^{-a^{N+1-j}} \prod_{j\in SI\setminus SI^*} g_1^{a^{N+1-j}}\Big)^{-h_{1,\rho_2(T)}'\prod f_j'} g_1^{-r_2'}.$$

- **Case** $T = T^*$: In this case, we have $R = R^*$. For each $ID \in R^*$, it adds (ID, T^*) to RL if $(ID, T') \notin RL$ for any $T' \le T^*$. Next, it selects a random exponent $r_2 \in \mathbb{Z}_p$ and creates an update key $UK_{T,R}$ as

$$U_0 = (g_1^b)^{\theta\prod f_j'} h_{1,\rho_1(T^*)}^{r_2},\ U_1 = g_1^{-r_2}.$$

If this is a decryption key query for an identity ID and a time T, then $\mathcal{B}$ proceeds as follows:

- **Case** $ID \neq ID^*$: If $(ID, -) \notin ST$, then it selects an index $d \in \mathcal{N}$ such that $(-, d) \notin ST$ and adds (ID, d) to ST. Next, it selects random exponents $r_1', r_2 \in \mathbb{Z}_p$ and creates a decryption key $DK_{ID,T}$ by implicitly setting $r_1 = (-af'_{\rho_1(ID)} + r_1')b$ as

$$D_0 = e\big((g_1^{a^N \prod_{j \neq \rho_1(ID)} f'_j})^{r_1'}, g_1^{b \prod h'_j}\big) h_{2,\rho_2(T)}^{r_2},\ D_1 = e\big((g_1^a)^{f'_{\rho_1(ID)}} g_1^{-r_1'}, g_1^b\big),\ D_2 = g_2^{-r_2}.$$

- **Case** $ID = ID^*$: In this case, we have $T \neq T^*$ from the restriction of Definition 2.2. It selects random exponents $r_1, r_2' \in \mathbb{Z}_p$ and creates a decryption key $DK_{ID,T}$ by implicitly setting $r_2 = (-ah'_{\rho_1(T)} + r_2')a^N$ as

$$D_0 = e\big((g_1^{b \prod_{j \neq \rho_2(T)} h'_j})^{r_2'}, g_1^{a^N \prod f'_j}\big) \cdot f_{2,\rho_1(ID)}^{r_1},\ D_1 = g_2^{-r_1},\ D_2 = e\big((g_1^a)^{h'_{\rho_2(T)}} g_1^{-r_2'}, g_1^{a^N}\big).$$

Challenge: $\mathcal{A}$ submits two challenge messages M_0^*, M_1^*. $\mathcal{B}$ chooses a random bit $\delta \in \{0,1\}$ and creates the challenge ciphertext CT^* by implicitly setting $s = c$ as

$$C = Z^{\prod f'_j \prod h'_j} \cdot M_\delta^*,\ C_0 = g_1^c,\ C_1 = (g_1^c)^{\prod_{j \neq \rho_1(ID^*)} f'_j},\ C_2 = (g_1^c)^{\prod_{j \neq \rho_2(T^*)} h'_j}.$$

Phase 2: Same as Phase 1.

Guess: Finally, $\mathcal{A}$ outputs a guess $\delta' \in \{0,1\}$. $\mathcal{B}$ outputs 0 if $\delta = \delta'$ or 1 otherwise. $\square$

B.3 Construction in the GGH Framework

Let $\mathcal{N} = \{1, \ldots, N\}$, $\mathcal{I} = \{ID_1, \ldots, ID_{n_1}\}$, and $\mathcal{T} = \{T_1, \ldots, T_{n_2}\}$. Our RIBE scheme for small universe in the three-graded encoding system is described as follows:

RIBE.Setup($1^\lambda, N$): This algorithm obtains $(\mathbf{params}, p_{zt})$ by running $\mathbf{InstGen}(1^\lambda, 1^3)$. Note that **params** includes a level 1 encoding of 1, which is denoted as g_1. It chooses random encodings $v_1', \{f_i'\}_{1 \le i \le n_1}$, $\{h_i'\}_{1 \le i \le n_2}$ by freshly calling $\mathbf{samp}()$ and sets

$$\{f_{1,i} = \mathbf{rerand}(1, \mathbf{enc}(1, f_i')),\ f_{2,i} = \mathbf{rerand}(2, g_1 \cdot f_{1,i})\}_{1 \le i \le n_1},$$
$$\{h_{1,i} = \mathbf{rerand}(1, \mathbf{enc}(1, h_i')),\ h_{2,i} = \mathbf{rerand}(2, g_1 \cdot h_{1,i})\}_{1 \le i \le n_2}.$$

Next, it chooses random encodings α, β, γ by freshly calling $\mathbf{samp}()$. It outputs a master key $MK = (\alpha, \beta, \gamma)$, an empty revocation list RL, an empty state ST, and public parameters as

$$PP = \Big((\mathbf{params}, p_{zt}),\ \{A_j = \mathbf{rerand}(1, \mathbf{enc}(1, v_1 \cdot \alpha^j))\}_{1 \le j, j \neq N+1 \le 2N},$$
$$B = \mathbf{rerand}(1, \mathbf{enc}(1, v_1 \cdot \beta)), v_1, \vec{f}_1, \vec{h}_1, \vec{f}_2, \vec{h}_2,\ \Omega = \mathbf{rerand}(3, \mathbf{enc}(3, v_1 \cdot \alpha^{N+1}\beta))\Big).$$

RIBE.GenKey(ID, MK, ST, PP): Let $d \in \mathcal{N}$ be an index for ID. It chooses a random encoding r_1 by calling $\mathbf{samp}()$ and outputs a private key by implicitly including ID and d as

$$SK_{ID} = \Big(K_0 = \mathbf{rerand}(1, \mathbf{enc}(1, v_1 \cdot \alpha^d \cdot \gamma) + (f_{1,\rho_1(ID)}) \cdot (-r_1)),\ K_1 = \mathbf{rerand}(1, \mathbf{enc}(1, -r_1))\Big).$$

RIBE.UpdateKey(T, RL, MK, ST, PP): It first defines the revoked set R, the revoked index set RI, and the non-revoked index set SI as the same as in Appendix B.1. It chooses a random encoding r_2 by calling **samp**() and outputs an update key by implicitly including T, R, and the revoked index set RI as

$$UK_{T,R} = \Big(U_0 = \mathbf{rerand}(1, \mathbf{enc}(1, (v_1 \cdot \gamma + v_1 \cdot \sum_{j \in SI} \alpha^{N+1-j}) \cdot \beta) + (h_{1,\rho_2(T)}) \cdot r_2)),$$
$$U_1 = \mathbf{rerand}(1, \mathbf{enc}(1, -r_2))\Big).$$

RIBE.DeriveKey($SK_{ID}, UK_{T,R}, PP$): Let $SK_{ID} = (K_0, K_1)$ and $UK_{T,R} = (U_0, U_1)$. If $ID \in R$, then it outputs $\perp$. Otherwise, it proceeds the following steps: Let d be the index of ID and RI be the revoked index set of R. It sets a non-revoked index set $SI = \mathcal{N} \setminus RI$ and derives components T_0, T_1 and T_2 as

$$T_0 = \mathbf{rerand}(2, (A_d \cdot U_0 - B \cdot (K_0 + \prod_{j \in S, j \neq d} A_{N+1-j+d}))),$$
$$T_1 = \mathbf{rerand}(2, B \cdot K_1),\ T_2 = \mathbf{rerand}(2, A_d \cdot U_1).$$

Next, it selects random encodings r'_1, r'_2 by freshly calling **samp**() and re-randomizes the temporal components as

$$D_0 = \mathbf{rerand}(2, T_0 + (f_{2,\rho_1(ID)}) \cdot r'_1 + (h_{2,\rho_2(T)}) \cdot r'_2),$$
$$D_1 = \mathbf{rerand}(2, T_1 + \mathbf{enc}(2, -r'_1)),\ D_2 = \mathbf{rerand}(2, T_2 + \mathbf{enc}(2, -r'_2)).$$

Finally, it outputs a decryption key $DK_{ID,T} = \big(D_0, D_1, D_2\big)$.

RIBE.Encrypt(ID, T, M, PP): It first chooses a random encoding s by calling **samp**(). If $M = 0$, it sets $C = \mathbf{rerand}(3, \Omega \cdot s)$. Otherwise, it sets $C = \mathbf{rerand}(3, \mathbf{enc}(3, \mathbf{samp}()))$. It outputs a ciphertext by implicitly including ID and T as

$$CT_{ID,T} = \Big(C,\ C_0 = \mathbf{rerand}(1, \mathbf{enc}(1, s)),\ C_1 = \mathbf{rerand}(1, (f_{1,\rho_1(ID)}) \cdot s),$$
$$C_2 = \mathbf{rerand}(1, (h_{1,\rho_2(T)}) \cdot s)\Big).$$

RIBE.Decrypt($CT_{ID,T}, DK_{ID',T'}, PP$): This algorithm is the same as that of Appendix B.1.

RIBE.Revoke(ID, T, RL, ST): This algorithm is the same as that of Appendix B.1.

B.4 Security Analysis in the GGH Framework

We translate the MDHE assumption in Section 2.3 into the graded encoding system version in the GGH framework.

Assumption B.2 (GGH analogue of Decisional Multilinear Diffie-Hellman Exponent, GGH (k,l)-MDHE)**.** *A challenger obtains $(\boldsymbol{params}, p_{zt})$ by running $\boldsymbol{InstGen}(1^\lambda, 1^k)$ and chooses random encodings $a, c_1, \ldots, c_{k-1}$ by calling $\boldsymbol{samp}()$. The GGH analogue of decisional (k,l)-MDHE assumption is that if the challenge tuple*

$$D = \big((\boldsymbol{params}, p_{zt}),\ \{\boldsymbol{rerand}(1, \boldsymbol{enc}(1, a^j))\}_{1 \leq j, j \neq l+1 \leq 2l},\ \{\boldsymbol{rerand}(1, \boldsymbol{enc}(1, c_i))\}_{1 \leq i \leq k-1}\big) \text{ and } Z$$

are given, no PPT algorithm $\mathcal{A}$ can distinguish $Z = Z_0 = \boldsymbol{rerand}(k, \boldsymbol{enc}(k, a^{l+1} \prod_{i=1}^{k-1} c_i))$ from $Z = Z_1 = \boldsymbol{rerand}(k, \boldsymbol{enc}(k, \boldsymbol{samp}()))$ with more than a negligible advantage. The advantage of $\mathcal{A}$ is defined as $\boldsymbol{Adv}_{\mathcal{A}}^{GGH\ (k,l)\text{-}MDHE}(\lambda) = \big|\Pr[\mathcal{A}(D, Z_0) = 0] - \Pr[\mathcal{A}(D, Z_1) = 0]\big|$.

Assumption B.3 (GGH analogue of Decisional Three-Leveled Multilinear Diffie-Hellman Exponent, GGH $(3,l)$-MDHE)**.** *A challenger obtains* $(\boldsymbol{params}, p_{zt})$ *by running* $\boldsymbol{InstGen}(1^\lambda, 1^3)$ *and chooses random encodings* a, b, c *by calling* $\boldsymbol{samp}()$*. The GGH analogue of decisional* $(3,l)$*-MDHE assumption is that if the challenge tuple*

$$D = \big((\boldsymbol{params}, p_{zt}),\ \{\boldsymbol{rerand}(1, \boldsymbol{enc}(1, a^j))\}_{1\le j, j\ne l+1\le 2l},\ \boldsymbol{rerand}(1, \boldsymbol{enc}(1, b)),\\ \boldsymbol{rerand}(1, \boldsymbol{enc}(1, c))\big) \text{ and } Z$$

are given, no PPT algorithm $\mathcal{A}$ *can distinguish* $Z = Z_0 = \boldsymbol{rerand}(3, \boldsymbol{enc}(3, a^{l+1}bc))$ *from* $Z = Z_1 = \boldsymbol{rerand}(3, \boldsymbol{enc}(3, \boldsymbol{samp}()))$ *with more than a negligible advantage. The advantage of* $\mathcal{A}$ *is defined as* $\mathbf{Adv}_{\mathcal{A}}^{GGH\ (3,l)\text{-}MDHE}(\lambda) = \big|\Pr[\mathcal{A}(D, Z_0) = 0] - \Pr[\mathcal{A}(D, Z_1) = 0]\big|$.

Theorem B.4. *The above RIBE scheme for small universe in graded encoding systems is secure in the selective revocation list model under chosen plaintext attacks if the GGH analogue of* $(3,N)$*-MDHE assumption holds where* N *is the maximum number of users in the system.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that attacks the above RIBE scheme in graded encoding systems with a non-negligible advantage. A simulator $\mathcal{B}$ that solves the GGH MDHE assumption using $\mathcal{A}$ is given: a challenge tuple $D = ((\mathbf{params}, p_{zt}), \{\mathbf{rerand}(1, \mathbf{enc}(1, a^j))\}_{1\le j, j\ne N+1\le 2N}, \mathbf{rerand}(1, \mathbf{enc}(1, b)), \mathbf{rerand}(1, \mathbf{enc}(1, c)))$ and Z where $Z = Z_0 = \mathbf{rerand}(3, \mathbf{enc}(3, a^{N+1}bc))$ or $Z = Z_1 = \mathbf{rerand}(3, \mathbf{enc}(3, \mathbf{samp}()))$. Then $\mathcal{B}$ that interacts with $\mathcal{A}$ is described as follows:

Init: $\mathcal{A}$ initially submits a challenge identity ID^*, a challenge time T^*, and a revoked identity set R^* at the time T^*. It first sets a state ST and a revocation list RL as empty one. For each $ID \in \{ID^*\} \cup R^*$, it selects an index $d \in \mathcal{N}$ such that $(-, d) \notin ST$ and adds (ID, d) to ST. Let $RI^* \subseteq \mathcal{N}$ be the revoked index set of R^* at the time T^* and SI^* be the non-revoked index set at the time T^* such that $SI^* = \mathcal{N} \setminus RI^*$.

Setup: $\mathcal{B}$ first chooses random encodings $\{f'_i\}_{1\le i\le n_1}, \{h'_i\}_{1\le i\le n_2}, \theta$ by freshly calling $\mathbf{samp}()$. It sets $\Gamma = \mathbf{rerand}(1, \mathbf{enc}(1, (\theta - \sum_{j\in SI^*}(a^{N+1-j}))))$ by implicitly setting $\gamma = \theta - \sum_{j\in SI^*} a^{N+1-j}$ and publishes the public parameters PP by implicitly setting $\alpha = a, \beta = b\prod h'_i$ as

$$(\mathbf{params}, p_{zt}), v_1 = \mathbf{rerand}(1, \mathbf{enc}(1, \prod f'_i)),$$
$$\big\{A_j = \mathbf{rerand}(1, \mathbf{enc}(1, a^j \cdot \prod f'_i))\big\}_{1\le j, j\ne N+1\le 2N},\ B = \mathbf{rerand}(1, \mathbf{enc}(1, b \cdot \prod f'_i \cdot \prod h'_i)),$$
$$\big\{f_{1,i} = \mathbf{rerand}(1, A_N \cdot \prod_{i\ne\rho_1(ID)} f'_i)\big\}_{1\le i\le n_1, i\ne\rho_1(ID^*)},\ f_{1,\rho(ID^*)} = \mathbf{rerand}(1, \prod_{i\ne\rho_1(ID)} f'_i),$$
$$\big\{h_{1,i} = \mathbf{rerand}(1, B \cdot \prod_{i\ne\rho_2(T)} h'_i),\ \big\}_{1\le i\le n_2, i\ne\rho_1(T^*)},\ h_{1,\rho(T^*)} = \mathbf{rerand}(1, \prod_{i\ne\rho_2(T)} h'_i),$$
$$\Omega = \mathbf{rerand}(3, A_1 \cdot A_N \cdot B \cdot \prod f'_i \cdot \prod h'_i).$$

Phase 1: $\mathcal{A}$ adaptively requests a polynomial number of private key, update key, and decryption key queries. If this is a private key query for an identity ID, then $\mathcal{B}$ proceeds as follows:

- **Case** $ID \in R^*$: It first retrieves a tuple (ID, d) from ST where the index d is associated with ID. Next, it selects a random encoding r_1 by calling $\mathbf{samp}()$ and creates a private key SK_{ID} as

$$K_0 = \mathbf{rerand}(1, A_d \cdot \prod f'_i \cdot \theta - \sum_{j\in SI^*} A_{N+1-j+d} \cdot \prod f'_i + f_{1,\rho_1(ID)} \cdot (-r_1),$$
$$K_1 = \mathbf{rerand}(1, \mathbf{enc}(1, -r_1)).$$

- **Case** $ID \notin R^*$: In this case, we have $ID \neq ID^*$ from the restriction of Definition 2.2. It first selects an index $d \in \mathcal{N}$ such that $(-,d) \notin ST$ and adds (ID,d) to ST. Next, it selects a random encoding r_1' by calling **samp**() and creates a private key SK_{ID} by implicitly setting $r_1 = -af'_{1,\rho_1(ID)} + r_1'$ as

$$K_0 = \textbf{rerand}(1, A_d \cdot \prod f_i' \cdot \theta - \sum_{j \in SI^* \setminus \{d\}} A_{N+1-j+d} \cdot \prod f_i' + f_{1,\rho_1(ID)} \cdot r_1',$$
$$K_1 = \textbf{rerand}(1, A_1 \cdot (f'_{1,\rho_1(ID)}) + \textbf{enc}(1, -r_1')).$$

If this is an update key query for a time T, then $\mathcal{B}$ defines a revoked identity set R at the time T from RL and proceeds as follows:

- **Case** $T \neq T^*$: It first sets a revoked index set RI of R by using ST. It also sets $SI = \mathcal{N} \setminus RI$. Next, it selects a random encoding r_2' by calling **samp**() and creates an update key $UK_{T,R}$ by implicitly setting $r_2 = -(-\sum_{j \in SI^* \setminus SI} a^{N+1-j} + \sum_{j \in SI \setminus SI^*} a^{N+1-j})h'_{1,\rho_2(T)} \prod f_i' + r_2'$ as

$$U_0 = \textbf{rerand}(1, B \cdot \theta \cdot \prod f_i' + +h_{1,\rho_2(T)} \cdot r_2'),$$
$$U_1 = \textbf{rerand}(1, (-\sum_{j \in SI^* \setminus SI} A_{N+1-j} + \sum_{j \in SI \setminus SI^*} A_{N+1-j}) \cdot (-h'_{1,\rho_2(T)} \prod f_i') + \textbf{enc}(1, -r_2')).$$

- **Case** $T = T^*$: In this case, we have $R = R^*$. For each $ID \in R^*$, it adds (ID, T^*) to RL if $(ID, T') \notin RL$ for any $T' \leq T^*$. Next, it selects a random encoding r_2 by calling **samp**() and creates an update key $UK_{T,R}$ as

$$U_0 = \textbf{rerand}(1, B \cdot \theta \cdot \prod f_i' + +h_{1,\rho_2(T^*)} \cdot r_2), \; U_1 = \textbf{rerand}(1, \textbf{enc}(1, -r_2)).$$

If this is a decryption key query for an identity ID and a time T, then $\mathcal{B}$ proceeds as follows:

- **Case** $ID \neq ID^*$: If $(ID,-) \notin ST$, then it selects an index $d \in \mathcal{N}$ such that $(-,d) \notin ST$ and adds (ID,d) to ST. Next, it selects random encodings r_1', r_2 by freshly calling **samp**() and creates a decryption key $DK_{ID,T}$ by implicitly setting $r_1 = (-af'_{1,\rho_1(ID)} + r_1')b$ as

$$D_0 = \textbf{rerand}(2, (A_N \cdot \prod_{i \neq \rho_1(ID)} f_i' \cdot r_1') \cdot B \cdot \prod h_i' + h_{2,\rho_2(T)} \cdot r_2),$$
$$D_1 = \textbf{rerand}(2, (A_1 \cdot (f'_{1,\rho_1(ID)}) + \textbf{enc}(1, -r_1')) \cdot B), \; D_2 = \textbf{rerand}(2, \textbf{enc}(2, -r_2)).$$

- **Case** $ID = ID^*$: In this case, we have $T \neq T^*$ from the restriction of Definition 2.2. It selects random encodings r_1, r_2' by freshly calling **samp**() and creates a decryption key $DK_{ID,T}$ by implicitly setting $r_2 = (-ah'_{1,\rho_2(T)} + r_2')a^N$ as

$$D_0 = \textbf{rerand}(2, (B \cdot \prod_{i \neq \rho_2(T)} h_i' \cdot r_2') \cdot A_N \cdot \prod f_i' + f_{2,\rho_1(ID)} \cdot r_1),$$
$$D_1 = \textbf{rerand}(2, \textbf{enc}(2, -r_1)), \; D_2 = \textbf{rerand}(2, (A_1 \cdot (h'_{1,\rho_2(T)}) + \textbf{enc}(1, -r_2')) \cdot A_N).$$

Challenge: $\mathcal{B}$ creates the challenge ciphertext CT^* by implicitly setting $s = c$ as

$$C = \textbf{rerand}(1, Z \cdot \prod f_i' \prod h_i'), \; C_0 = \textbf{rerand}(1, \textbf{enc}(1, c)),$$
$$C_1 = \textbf{rerand}(1, C_0 \cdot \prod_{i \neq \rho_1(ID^*)} f_i'), \; C_2 = \textbf{rerand}(1, C_0 \cdot \prod_{i \neq \rho_2(T^*)} h_i').$$

If $Z = Z_0$ then this is an encryption of 0; Otherwise ($Z = Z_1$) then it is an encryption of 1.
Phase 2: Same as Phase 1.
Guess: Finally, $\mathcal{A}$ outputs a guess $\delta \in \{0,1\}$. $\mathcal{B}$ outputs 0 if $\delta = 0$ or 1 otherwise. $\square$

C Revocable IBE with Trade-Offs

In this section, we propose another RIBE scheme such that the number of public parameters is reduced from $O(N+\lambda)$ to $O(\lambda)$ group elements by increasing the number of update key elements. The basic idea of our general construction is to use the parallel construction technique of PKBE that reduces the size of public parameters and ciphertexts [8]. Additionally, we can reduce the size of public parameters further in our scheme since an authorized authority in RIBE only can broadcast an update key. That is, we can safely move some elements in public parameters that are used for broadcasting into an update key.

C.1 Construction

Let N be the maximum number of users and $m = \lceil \sqrt{N} \rceil$. An index $d \in \{1,\ldots,N\}$ is represented as a position (d_x, d_y) in a $m \times m$ matrix where $d = (d_y - 1)m + d_x$ for some $1 \le d_y \le m$ and $1 \le d_x \le m$. Let SI be a subset of $\{1,\ldots,N\}$, and define $SI'_k = SI \cap \{(k-1)m+1,\ldots,(k-1)m+m\}$ and $SI_k = \{x-(k-1)m | x \in SI'_k\} \subseteq \{1,\ldots,m\}$. A subset SI is divided to subsets $SI_1,\ldots,SI_m$. Let $\mathcal{N} = \{1,\ldots,N\}, \mathcal{I} = \{0,1\}^{l_1}$, and $\mathcal{T} = \{0,1\}^{l_2}$. Our RIBE scheme with shorter public parameters in three-leveled multilinear maps is described as follows:

RIBE.Setup($1^\lambda, N$): This algorithm takes as input a security parameter 1^λ and the maximum number N of users. It generates a 3-leveled multilinear group $\vec{\mathbb{G}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3)$ of prime order p. Let g_1, g_2, g_3 be canonical generators of $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ respectively. Let PP_{MLM} be the description of the multilinear group with generators.

1. It selects random elements $f_{1,0}, \{f_{1,i,j}\}_{1 \le i \le l_1, j \in \{0,1\}}, h_{1,0}, \{h_{1,i,j}\}_{1 \le i \le l_2, j \in \{0,1\}} \in \mathbb{G}_1$. Let $\vec{f}_k = (f_{k,0}, \{f_{k,i,j}\}_{1 \le i \le l_1, j \in \{0,1\}})$ and $\vec{h}_k = (h_{k,0}, \{h_{k,i,j}\}_{1 \le i \le l_2, j \in \{0,1\}})$ for a level k. We define $F_k(ID) = f_{k,0} \prod_{i=1}^{l_1} f_{k,i,ID[i]}$ and $H_k(T) = h_{k,0} \prod_{i=1}^{l_2} h_{k,i,T[i]}$ where $ID[i]$ is a bit value at the position i and $T[i]$ is a bit value at the position i.
2. Next, it selects random exponents $\alpha, \beta, \gamma_1, \ldots, \gamma_m \in \mathbb{Z}_p$. It outputs a master key $MK = \big(\alpha, \beta, \{\gamma_j\}_{1 \le j \le m}, \{g_1^{\alpha^j}\}_{1 \le j, j \ne m+1 \le 2m}, g_1^{\beta}, \{g_1^{\gamma_k}\}_{1 \le k \le m}\big)$, an empty revocation list RL, an empty state ST, and public parameters as

$$PP = \Big(PP_{MLM},\ \vec{f}_1,\ \vec{h}_1,\ \Omega = g_3^{\alpha^{m+1}\beta}\Big).$$

RIBE.GenKey(ID, MK, ST, PP): This algorithm takes as input an identity $ID \in \mathcal{I}$, the master key MK, the state ST, and public parameters PP. It first assigns an index $d \in \mathcal{N}$ that is not in ST to the identity ID, and updates the state ST by adding a tuple (ID, d) to ST. Note that we can represent d as (d_x, d_y). Next, it selects a random exponent $r_1 \in \mathbb{Z}_p$ and outputs a private key by implicitly including ID and the index d as

$$SK_{ID} = \Big(K_0 = g_1^{\alpha^{d_x}\gamma_{d_y}} F_1(ID)^{-r_1},\ K_1 = g_1^{-r_1}\Big).$$

RIBE.UpdateKey(T, RL, MK, ST, PP): This algorithm takes as input a time T, the revocation list RL, the master key MK, the state ST, and public parameters PP.

1. It first defines the revoked set R of user identities at the time T from RL. That is, if there exists (ID', T') such that $(ID', T') \in RL$ for any $T' \leq T$, then $ID' \in R$. Next, it defines the revoked index set $RI \subseteq \mathcal{N}$ of the revoked identity set R by using the state ST since ST contains (ID, d). It also defines the non-revoked index set $SI = \mathcal{N} \setminus RI$ such that $SI = SI_1 \cup \cdots \cup SI_m$.
2. It selects a random exponent $r_{2,1}, \ldots, r_{2,m} \in \mathbb{Z}_p$ and outputs an update key by implicitly including T, R, and the revoked index set RI as

$$UK_{T,R} = \Big(\{g_1^{\alpha^j}\}_{1 \leq j, j \neq m+1 \leq 2m},\ g_1^{\beta},\ \big\{U_{k,0} = \big(g_1^{\gamma_k} \prod_{j \in SI_k} g_1^{\alpha^{m+1-j}}\big)^{\beta} H_1(T)^{r_{2,k}},\ U_{k,1} = g_1^{-r_{2,k}}\big\}_{1 \leq k \leq m} \Big).$$

RIBE.DeriveKey($SK_{ID}, UK_{T,R}, PP$): This algorithm takes as input a private key $SK_{ID} = (K_0, K_1)$ for an identity ID, an update key $UK_{T,R} = (\{g_1^{\alpha^j}\}, g_1^{\beta}, \{U_{k,0}, U_{k,1}\})$ for a time T and a revoked set R of identities, and the public parameters PP. If $ID \in R$, then it outputs $\perp$ since the identity ID is revoked. Otherwise, it proceeds the following steps:

1. Let $d = (d_x, d_y)$ be the index of ID and RI be the revoked index set of R. Note that these are implicitly included in SK and UK respectively. It sets a non-revoked index set $SI = \mathcal{N} \setminus RI$ such that $SI = SI_1 \cup \cdots \cup SI_m$ and derives temporal components T_0, T_1 and T_2 as

$$T_0 = e(g_1^{\alpha^{d_x}}, U_{d_y,0}) \cdot e(g_1^{\beta}, K_0 \prod_{j \in SI_{d_y}, j \neq d_x} g_1^{\alpha^{m+1-j+d_x}})^{-1},\ T_1 = e(g_1^{\beta}, K_1),\ T_2 = e(g_1^{\alpha^{d_x}}, U_{d_y,1}).$$

2. Next, it chooses random exponents $r_1', r_2' \in \mathbb{Z}_p$ and re-randomizes the temporal components as $D_0 = T_0 \cdot F_2(ID)^{r_1'} H_2(T)^{r_2'}, D_1 = T_1 \cdot g_2^{-r_1'}, D_2 = T_2 \cdot g_2^{-r_2'}$. Finally, it outputs a decryption key as $DK_{ID,T} = \big(D_0, D_1, D_2\big)$.

RIBE.Encrypt(ID, T, M, PP): This algorithm is the same as that of Section 3.1.

RIBE.Decrypt($CT_{ID,T}, DK_{ID',T'}, PP$): This algorithm is the same as that of Section 3.1.

RIBE.Revoke(ID, T, RL, ST): This algorithm is the same as that of Section 3.1.

C.2 Security Analysis

Theorem C.1. *The above RIBE scheme is secure in the selective revocation list model under chosen plaintext attacks if the $(3,m)$-MDHE assumption holds where N is the maximum number of users and $m = \sqrt{N}$. That is, for any PPT adversary $\mathcal{A}$, we have that $\boldsymbol{Adv}_{RIBE,\mathcal{A}}^{IND\text{-}sRL\text{-}CPA} \leq \boldsymbol{Adv}_{\mathcal{B}}^{(3,m)\text{-}MDHE}$.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that attacks the above RIBE scheme with a non-negligible advantage. A simulator $\mathcal{B}$ that solves the MDHE assumption using $\mathcal{A}$ is given: a challenge tuple $D = (g_1, g_1^a, g_1^{a^2}, \ldots, g_1^{a^m}, g_1^{a^{m+2}}, \ldots, g_1^{a^{2m}}, g_1^b, g_1^c)$ and Z where $Z = Z_0 = g_3^{a^{m+1}bc}$ or $Z = Z_1 \in \mathbb{G}_3$. Then $\mathcal{B}$ that interacts with $\mathcal{A}$ is described as follows:

Init: $\mathcal{A}$ initially submits a challenge identity ID^*, a challenge time T^*, and a revoked identity set R^* at the time T^*. It first sets a state ST and a revocation list RL as empty one. For each $ID \in \{ID^*\} \cup R^*$, it selects an index $d \in \mathcal{N}$ such that $(-, d) \notin ST$ and adds (ID, d) to ST. Let $RI^* \subseteq \mathcal{N}$ be the revoked index set of R^* at the time T^* and SI^* be the non-revoked index set at the time T^* such that $SI^* = \mathcal{N} \setminus RI^*$. Note that SI^* is divided to subsets $SI_1^*, \ldots, SI_m^*$.

Setup: $\mathcal{B}$ first chooses random exponents $\theta_1,\ldots,\theta_m \in \mathbb{Z}_p$ and sets master key elements by implicitly setting $\alpha = a, \beta = b$, $\{\gamma_k = \theta_k - \sum_{j\in SI_k^*} a^{m+1-j}\}$ as

$$\{g_1^{\alpha^j} = g_1^{a^j}\}_{1\le j, j\ne m+1\le 2m},\ g_1^{\beta} = g_1^{b},\ \{g_1^{\gamma_k} = g_1^{\theta_k}\prod_{j\in SI_k^*}(g^{a^{m+1-j}})^{-1}\}_{1\le k\le m}.$$

Next, it selects random exponents $f_0', \{f_{i,j}'\}_{1\le i\le l_1, j\in\{0,1\}}, h_0', \{h_{i,j}'\}_{1\le i\le l_2, j\in\{0,1\}} \in \mathbb{Z}_p$ and publishes the public parameters PP as

$$\vec{f}_1 = \big(f_{1,0} = g_1^{f_0'}\big(\prod_{i=1}^{l_1} f_{1,i,ID^*[i]}\big)^{-1},\ \{f_{1,i,j} = (g_1^{a^N})^{f_{i,j}'}\}_{1\le i\le l_1, j\in\{0,1\}}\big),$$
$$\vec{h}_1 = \big(h_{1,0} = g_1^{h_0'}\big(\prod_{i=1}^{l_2} h_{1,i,T^*[i]}\big)^{-1},\ \{h_{1,i,j} = (g_1^{b})^{h_{i,j}'}\}_{1\le i\le l_2, j\in\{0,1\}}\big),$$
$$\Omega = e\big(e(g_1^a, g_1^{a^m}), g_1^b\big) = g_3^{a^{m+1}b}.$$

For notational simplicity, we define $\Delta ID = \sum_{i=1}^{l_1}(f_{i,ID[i]}' - f_{i,ID^*[i]}')$ and $\Delta T = \sum_{i=1}^{l_2}(h_{i,T[i]}' - h_{i,T^*[i]}')$.

Phase 1: $\mathcal{A}$ adaptively requests a polynomial number of private key, update key, and decryption key queries. If this is a private key query for an identity ID, then $\mathcal{B}$ proceeds as follows:

- **Case** $ID \in R^*$: It first retrieves a tuple (ID, d) from ST where the index $d = (d_x, d_y)$ is associated with ID. Note that the tuple (ID, d) exists since all identities in R^* were added to ST in the initialization step. Next, it selects a random exponent $r_1 \in \mathbb{Z}_p$ and creates a private key SK_{ID} as

$$K_0 = (g_1^{a^{d_x}})^{\theta_{d_y}}\big(\prod_{j\in SI_{d_y}^*} g_1^{a^{m+1-j+d_x}}\big)^{-1} F_1(ID)^{-r_1},\ K_1 = g_1^{-r_1}.$$

- **Case** $ID \notin R^*$: In this case, we have $ID \ne ID^*$ from the restriction of Definition 2.2. It first selects an index $d \in \mathcal{N}$ such that $(-, d) \notin ST$ and adds (ID, d) to ST. Note that the index d can be represented as (d_x, d_y). It selects a random exponents $r_1' \in \mathbb{Z}_p$ and creates a private key SK_{ID} by implicitly setting $r_1 = -a/\Delta ID + r_1'$ as

$$K_0 = (g_1^{a^{d_x}})^{\theta_{d_y}}\prod_{j\in SI_{d_y}^*\setminus\{d_x\}} g_1^{-a^{m+1-j+d_x}}(g_1^a)^{f_0'/\Delta ID} F_1(ID)^{-r_1'},\ K_1 = (g_1^a)^{-1/\Delta ID} g_1^{r_1'}.$$

If this is an update key query for a time T, then $\mathcal{B}$ defines a revoked identity set R at the time T from RL and proceeds as follows:

- **Case** $T \ne T^*$: It first sets a revoked index set RI of R by using ST. It also sets $SI = \mathcal{N}\setminus RI$. Note that SI is divided to $SI_1,\ldots,SI_m$. Next, it selects random exponents $r_{2,1}',\ldots,r_{2,m}' \in \mathbb{Z}_p$ and creates an update key $UK_{T,R}$ by implicitly setting $\{r_{2,k} = -(-\sum_{j\in SI_k^*\setminus SI_k} a^{m+1-j} + \sum_{j\in SI_k\setminus SI_k^*} a^{m+1-j})/\Delta T + r_{2,k}'\}$ as

$$\{g_1^{\alpha_j}\}_{1\le j, j\ne m+1\le 2m},\ g_1^{\beta},$$
$$\Big\{U_{k,0} = (g_1^b)^{\theta_k}\big(\prod_{j\in SI_k^*\setminus SI_k} g_1^{-a^{m+1-j}}\prod_{j\in SI_k\setminus SI_k^*} g^{a^{m+1-j}}\big)^{-h_0'/\Delta T} H_1(T)^{r_{2,k}'},$$
$$U_{k,1} = \big(\prod_{j\in SI_k^*\setminus SI_k} g_1^{-a^{m+1-j}}\prod_{j\in SI_k\setminus SI_k^*} g_1^{a^{m+1-j}}\big)^{-1/\Delta T} g^{r_{2,k}'}\Big\}_{1\le k\le m}.$$

- **Case** $T = T^*$: In this case, we have $R = R^*$. For each $ID \in R^*$, it adds (ID, T^*) to RL if $(ID, T') \notin RL$ for any $T' \leq T^*$. It selects random exponents $r_{2,1}, \ldots, r_{2,m} \in \mathbb{Z}_p$ and creates an update key $UK_{T,R}$ as

$$\{g_1^{\alpha_j}\}_{1 \leq j, j \neq m+1 \leq 2m},\ g_1^{\beta},\ \Big\{U_{k,0} = (g_1^b)^{\theta_k} H_1(T^*)^{r_{2,k}},\ U_{k,1} = g_1^{-r_{2,k}}\Big\}_{1 \leq k \leq m}.$$

If this is a decryption key query for an identity ID and a time T, then $\mathcal{B}$ proceeds as follows:

- **Case** $ID \neq ID^*$: If $(ID, -) \notin ST$, then it selects an index $d \in \mathcal{N}$ such that $(-, d) \notin ST$ and adds (ID, d) to ST. It selects random exponents $r_1', r_2 \in \mathbb{Z}_p$ and creates a decryption key $DK_{ID,T}$ by implicitly setting $r_1 = (-a/\Delta ID + r_1')b$ as

$$D_0 = e\big((g_1^a)^{-f_0'/\Delta ID} F_1(ID)^{r_1'}, g_1^b\big) H_2(T)^{r_2},\ D_1 = e((g_1^a)^{-1/\Delta ID} g_1^{r_1'}, g^b),\ D_2 = g_2^{r_2}.$$

- **Case** $ID = ID^*$: In this case, we have $T \neq T^*$ from the restriction of Definition 2.2. It selects random exponents $r_1, r_2' \in \mathbb{Z}_p$ and creates a decryption key $DK_{ID,T}$ by implicitly setting $r_2 = (-a/\Delta T + r_2')a^m$ as

$$D_0 = e\big((g_1^a)^{-h_0'/\Delta T} H_1(T)^{r_2'}, g_1^{a^m}\big) F_2(ID)^{r_1},\ D_1 = g_2^{r_1},\ D_2 = e((g_1^a)^{-1/\Delta T} g_1^{r_2'}, g_1^{a^m}).$$

Challenge: $\mathcal{A}$ submits two challenge messages M_0^*, M_1^*. $\mathcal{B}$ chooses a random bit $\delta \in \{0,1\}$ and creates the challenge ciphertext CT^* by implicitly setting $s = c$ as

$$C = Z \cdot M_\delta^*,\ C_0 = g_1^c,\ C_1 = (g_1^c)^{f_0'},\ C_2 = (g_1^c)^{h_0'}.$$

Phase 2: Same as Phase 1.

Guess: Finally, $\mathcal{A}$ outputs a guess $\delta' \in \{0,1\}$. $\mathcal{B}$ outputs 0 if $\delta = \delta'$ or 1 otherwise. □

D Security in Generic Multilinear Groups

In this section, we introduce the definition of generic multilinear groups and discuss the difficulty of our new assumptions in generic multilinear groups.

D.1 Generic Multilinear Groups

We define the generic multilinear groups by following the generic group model [6, 38]. Let k be the target integer. Let $\xi : \mathbb{Z}_p \times \mathbb{Z} \to \{0,1\}^m$ be a random injective encoding that maps elements of the additive group $\mathbb{Z}_p$ and an integer $\mathbb{Z}$ into strings of length m. We define the groups $\mathbb{G}_i = \{\xi(x,i) | x \in \mathbb{Z}_p\}$. We are given oracles to compute the multiplication and pairing operations. That is, an algorithm in the generic multilinear groups is given the following oracles:

Encode(x,i) If i is a non-negative integer such that $i \leq k$, then it returns $\xi(x,i)$. Otherwise it returns $\perp$. Note that the generator g_i for the group $\mathbb{G}_i$ can be obtained as **Encode**$(1,i)$.

Mult(ξ_1, ξ_2, b) If $\xi_1 = \xi(x_1, i)$ and $\xi_2 = \xi(x_2, j)$ where $i = j$, then it returns $\xi(x_1 + (-1)^b x_2, i)$. Otherwise, it returns $\perp$.

Pair(ξ_1, ξ_2) If $\xi_1 = \xi(x_1, i)$ and $\xi_2 = \xi(x_2, j)$ where $i + j \leq k$, then it returns $\xi(x_1 \cdot x_2, i + j)$. Otherwise it returns $\perp$.

D.2 Analysis of New Assumptions

The master theorem of Boneh, Boyen, and Goh [6] is widely used to prove the difficulty of an assumption in generic bilinear groups. It is relatively straightforward to extend the master theorem of Boneh et al. in generic multilinear groups as pointed by Boneh, Waters, and Zhandry [12]. The master theorem informally states that if the target polynomial is independent of given polynomials in the assumption, then the advantage of an adversary in generic groups is bounded by $q^2 d/p$, where q is the maximum number of queries, d is the maximum degree of polynomials that the adversary can obtain by performing pairing operations, and p is in $\mathbb{Z}_p$.

In the (k,N)-MDHE assumption, the target polynomial f is $a^{N+1}\prod_{i=1}^{k-1} c_i$ where a and c_i are variables. We need two polynomial multiplications (pairings) to obtain a^{N+1} since a^{N+1} is not directly given in the assumption, and we need $k-1$ polynomial multiplications to obtain $\prod_{i=1}^{k-1} c_i$. Thus, we need $k+1$ polynomial multiplications (pairings) to obtain the target polynomial, but this is not allowed in the k-leveled multilinear maps. Therefore, the target polynomial is independent of given polynomials. We have the degree of polynomials $d = O(kN)$ since the adversary can obtain elements with high-degree a^{kN} by performing pairing operations. For λ-bit security, we can set $p \approx 2^{\lambda}$ since N is a polynomial value in a security parameter.

The difficulty of the assumption of Boneh, Waters, and Zhandry [12] is already given in generic multilinear groups. We can also follow their analysis since our cMDHE assumption is a slight modification of their assumption. In the (k,n,l)-cMDHE assumption, the target polynomial f is $a^{2^n-1}bc$ where a,b, and c are variables. We need n polynomial multiplications (pairings) to obtain a^{2^n-1} since $\{a^{2^i}\}_{i\in[0,n]}$ are only given in the assumption. Thus the target polynomial $a^{2^n-1}bc$ should reside in $2n+l-1$-level since b is a polynomial in the l-level and c is a polynomial in the $n-1$-level, but this is not allowed in the $2n+l-2$-leveled multilinear maps. Therefore, the target polynomial is independent of given polynomials. We have $d = O(n2^n)$ since the assumption includes elements with high degree a^{2^n}. For λ-bit security, we can set $p \approx 2^{3\lambda}$ instead of $p \approx 2^{\lambda}$.