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** 3
   1.1 Our Results 3
   1.2 Our Technique 4
   1.3 Related Work 6

2 **Preliminaries** 7
   2.1 Revocable Identity-Based Encryption 7
   2.2 Leveled Multilinear Maps 9
   2.3 Complexity Assumptions 9

3 **Revocable IBE with Shorter Keys** 10
   3.1 Construction 10
   3.2 Correctness 12
   3.3 Security Analysis 13
   3.4 Discussions 16

4 **Revocable IBE with Shorter Parameters** 17
   4.1 Construction 17
   4.2 Correctness 19
   4.3 Security Analysis 20
   4.4 Discussions 22

5 **Conclusion** 22

A **Revocable IBE in Graded Encoding Systems** 26
   A.1 Graded Encoding Systems 26
   A.2 Construction 28

B **Revocable IBE for Small Universe** 29
   B.1 Construction 29
   B.2 Security Analysis 30
   B.3 Construction in the GGH Framework 32
   B.4 Security Analysis in the GGH Framework 33

C **Revocable IBE with Trade-Offs** 36
   C.1 Construction 36
   C.2 Security Analysis 37

D **Security in Generic Multilinear Groups** 39
   D.1 Generic Multilinear Groups 39
   D.2 Analysis of New Assumptions 40
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 Public 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,
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 [7] 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 = (g^{αT}F(ID)^{r_1}, g^{r_2}), which is a careful combination of the PKBE private key SK_{BE,d} = g^{αT} and the HIBE private key SK_{HIBE,ID} = (g^{αT}F(ID)^{r_1}, g^{r_2}), where an index d is associated with the identity ID and F(·) is a function from identities to group elements. That is, we replace the master key part g^d of the HIBE private key component with the PKBE private key component. Next, we set the RIBE update key as UK_{T,R} = ((g^T)^{(i)N}, R g^{N+1-j}) \cdot H(T)^{r_2}, g^{r_2}), which is a careful combination of the PKBE ciphertext CT_{BE,R} = (g^{T}, (g^T)^{(i)N} \cdot R g^{N+1-j}) \cdot H(T)^{r_2}) and the HIBE private key SK_{HIBE,T} = (g^{T}\cdot H(T)^{r_2}, g^{r_2}) where R is a revocation set, T is an update time period, and H(·) is a function from times to group elements. That is, we replace the master key part g^d 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} = (g^{αT+1}F(ID)^{r_1}H(T)^{r_2}, g^{r_2}) for his identity ID and the time T. This decryption key can be used to decrypt a ciphertext CT_{ID,T} = (e(g^{αT+1}, g^{β})^t \cdot M, g^{β} \cdot F(ID)^{ty}, H(T)^{ty})

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 \( \mathbb{G}_2 \)

### Table 1: Comparison of revocable identity-based encryption schemes

<table>
<thead>
<tr>
<th>Scheme</th>
<th>PP Size</th>
<th>SK Size</th>
<th>UK Size</th>
<th>Model</th>
<th>Maps</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF</td>
<td>(O(1))</td>
<td>(O(1))</td>
<td>(O(N-r))</td>
<td>Full</td>
<td>BLM</td>
<td>RO, BDH</td>
</tr>
<tr>
<td>BGK</td>
<td>(O(1))</td>
<td>(O(\log N))</td>
<td>(O(r \log (N/r)))</td>
<td>Selective</td>
<td>BLM</td>
<td>DBDH</td>
</tr>
<tr>
<td>LV</td>
<td>(O(\lambda))</td>
<td>(O(\log N))</td>
<td>(O(r \log (N/r)))</td>
<td>Full</td>
<td>BLM</td>
<td>DBDH</td>
</tr>
<tr>
<td>SE</td>
<td>(O(\lambda))</td>
<td>(O(\log N))</td>
<td>(O(r \log (N/r)))</td>
<td>Full</td>
<td>BLM</td>
<td>DBDH</td>
</tr>
<tr>
<td>LLP</td>
<td>(O(1))</td>
<td>(O(\log^{1.5} N))</td>
<td>(O(r))</td>
<td>Full</td>
<td>BLM</td>
<td>Static</td>
</tr>
<tr>
<td>Our-1</td>
<td>(O(N+\lambda))</td>
<td>(O(1))</td>
<td>(O(1))</td>
<td>Selective</td>
<td>MLM</td>
<td>MDHE</td>
</tr>
<tr>
<td>Our-2</td>
<td>(O(\log N+\lambda))</td>
<td>(O(1))</td>
<td>(O(1))</td>
<td>Selective</td>
<td>MLM</td>
<td>cMDHE</td>
</tr>
</tbody>
</table>

Let \( \lambda \) 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.
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 $R$. 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 $ID \parallel T$ 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 \( I \), the time space \( T \), and the message space \( M \), 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 \( 1^\lambda \) 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 \( ID \in I \), 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 \( T \in T \), 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 \( ⊥ \).

- **Encrypt**\( (ID, T, M, PP) \): The encryption algorithm takes as input an identity \( ID \in I \), a time \( T \), a message \( M \in M \), 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 \( ⊥ \).

- **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 \( RL \), \( CT_{ID,T} \) generated by **Encrypt**\( (ID, T, M, PP) \) for any \( ID, T, M \), and \( PP \), it is required that

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

- Define $\mathbf{PPT}$ adversary $A$ under chosen plaintext attacks is defined in terms of the following experiment between a challenger.

- The following is the formal definition of the selective revocation security.

- If $(ID \in R)$, then $\text{DeriveKey}(SK_{ID}, UK_{T,R}, PP) = \bot$ with all but negligible probability.

- If $(ID_0 = ID) \land (T_0 = T)$, then $\text{Decrypt}(CT_{ID_0,T_0}, DK_{ID,T}) = M$.

- If $(ID_0 \neq ID) \lor (T_0 \neq T)$, then $\text{Decrypt}(CT_{ID,T}, DK_{ID,T}, PP) = \bot$ 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 $C$ and a PPT adversary $A$:

1. **Init**: $A$ initially submits a challenge identity $ID^* \in I$, a challenge time $T^* \in T$, and a revoked identity set $R^* \subseteq I$ at the time $T^*$.

2. **Setup**: $C$ generates a master key $MK$, a revocation list $RL$, a state $ST$, and public parameters $PP$ by running $\text{Setup}(1^k, N)$. It keeps $MK, RL, ST$ to itself and gives $PP$ to $A$.

3. **Phase 1**: $A$ 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 $A$ by running $\text{GenKey}(ID, MK, ST, PP)$ 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 $A$ by running $\text{UpdateKey}(T, RL, MK, ST, PP)$ 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 $A$ by running $\text{DeriveKey}(SK_{ID}, UK_{T,R}, PP)$ 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 $\text{Revoke}(ID, T, RL, ST)$ 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 $A$ 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**: $A$ submits two challenge messages $M_0^*, M_1^* \in M$ with equal length. $C$ flips a random coin $b \in \{0, 1\}$ and gives the challenge ciphertext $CT^*$ to $A$ by running $\text{Encrypt}(ID^*, T^*, M_b^*, PP)$. 


5. **Phase 2**: 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, A outputs a guess $b' \in \{0, 1\}$, and wins the game if $b = b'$.

The advantage of A is defined as $\text{Adv}^{\text{IND-RIBE, A}}_A(\lambda) = \left| \Pr[b = b'] - \frac{1}{2} \right|$ 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 A, the advantage of A in the above experiment is negligible in the security parameter $\lambda$.

**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.

### 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 $\lambda$ and a positive integer $k$. Let $\tilde{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_k)$ 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 $\{e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \rightarrow \mathbb{G}_{i+j}|i,j \geq 1;i+j \leq k\}$ 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 $\tilde{\mathbb{G}}$ is a multilinear group if the group operations in $\tilde{\mathbb{G}}$ 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 $(p, \tilde{\mathbb{G}}, \{e_{i,j}|i,j \geq 1;i+j \leq k\})$ 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

$$D = \left(g_1^{a_1}, g_1^{a_2}, \ldots, g_1^{a_N}, g_1^{a_{N+2}}, \ldots, g_1^{a_{2N}}, g_1^{c_1}, \ldots, g_1^{c_{k-1}}\right)$$

are given, no PPT algorithm $A$ can distinguish $Z = Z_0 = g_k^{a_1} \prod_{i=1}^{k-1} c_i$ from a random element $Z = Z_1 \in \mathbb{G}_k$ with more than a negligible advantage. The advantage of A is defined as $\text{Adv}^{(k,N)\text{-MDHE}}_A(\lambda) = \left| \Pr[A(D,Z_0) = 0] - \Pr[A(D,Z_1) = 0] \right|$ where the probability is taken over random choices of $a, c_1, \ldots, c_{k-1} \in \mathbb{Z}_p$.

---

1Boldyreva 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 \((p, \vec{G}, e_{1,1}, e_{1,2}, e_{2,1})\) 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

\[
D = \left(g_1^{a^k}, g_1^{d^k}, \ldots, g_1^{x^k}, g_1^{y^k}, \ldots, g_1^{N^k}, \ldots, g_1^{a^{N^k}}, g_1^{b^k}, g_1^{c^k}\right)
\]

are given, no PPT algorithm \(A\) 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 \(A\) is defined as

\[
\text{Adv}^{(3,N)\text{-MDHE}}_A (\lambda) = \left| \Pr[A(D, Z_0) = 0] - \Pr[A(D, Z_1) = 0] \right|
\]

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 \cite{BWZ} 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 \((p, \vec{G}, \{e_{ij} \mid i, j \geq 1; i + j \leq k\})\) 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

\[
D = \left(g_1^{a^{2^n}}, g_1^{d^{2^n}}, \ldots, g_1^{x^{2^n}}, g_1^{y^{2^n}}, \ldots, g_1^{N^{2^n}}, \ldots, g_1^{a^{N^{2^n}}}, g_1^{b^{2^n}}, g_1^{c^{2^n}}\right)
\]

are given, no PPT algorithm \(A\) can distinguish \(Z = Z_0 = g_3^{a^{2^n+1}bc}\) from a random element \(Z = Z_1 \in \mathbb{G}_{2n+l-2}\) with more than a negligible advantage. The advantage of \(A\) is defined as

\[
\text{Adv}^{(k,n,l)\text{-MDHE}}_A (\lambda) = \left| \Pr[A(D, Z_0) = 0] - \Pr[A(D, Z_1) = 0] \right|
\]

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 \([\text{D}]\).

## 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 \cite{BGW}, which uses bilinear maps.

### 3.1 Construction

Let \(N = \{1, \ldots, N\}\) where \(N\) is the (polynomial) number of users. Let \(I = \{0, 1\}^{l_1}\) be the identity space and \(T = \{0, 1\}^{l_2}\) be the time space where \(l_1 = 2\lambda\) and \(l_2 = \lambda\) for a security parameter \(\lambda\). 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 \(1^\lambda\) and the maximum number \(N\) of users. It generates a 3-leveled multilinear group \(\vec{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3)\) 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 \cite{BWZ}, 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 $f_{i,0}, \{f_{i,j}\}_{1 \leq i \leq h, j \in \{0,1\}}, h_{1,0}, \{h_{1,j}\}_{1 \leq i \leq h, j \in \{0,1\}} \in \mathbb{G}_1$. Let $\bar{f}_k = (f_{k,0}, \{f_{k,j}\}_{1 \leq i \leq h, j \in \{0,1\}})$ and $\bar{h}_k = (h_{k,0}, \{h_{k,j}\}_{1 \leq i \leq h, j \in \{0,1\}})$ for a level $k$. Note that we can obtain $\bar{f}_2$ and $\bar{h}_2$ from $\bar{f}_1$ and $\bar{h}_1$ by performing pairing operations. We define $F_k(ID) = \prod_{i=1}^{h_i} \bar{f}_{k,i,ID[i]}$ and $H_k(T) = h_{k,0} \prod_{i=1}^{h_i} h_{k,i,T[i]}$ 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 $PP$.

\[ PP = \left( PP_{\text{MLM}}, \{ g_{1,1}^\alpha \}_{1 \leq i, j \neq N+1 \leq 2N}, g_1^\beta, \bar{f}_1, \bar{h}_1, \Omega = g_3^{\alpha^{N+1}\beta} \right) \in \mathbb{G}_1^{2N+2l_1+2l_2+3} \times \mathbb{G}_3. \]

**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

\[ SK_{ID} = \left( K_0 = g_1^{\alpha \gamma} F_1(ID)^{-r_1}, K_1 = g_1^{-r_1} \right) \in \mathbb{G}_1^2. \]

**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 $T' \leq T$, 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 $SI = \mathcal{N} \setminus RI$.

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

\[ UK_{T,R} = \left( U_0 = \left( g_1^\gamma \prod_{j \in RI} g_1^{\alpha^{N+1-j}} \right)^\beta H_1(T)^{r_2}, U_1 = g_1^{-r_2} \right) \in \mathbb{G}_1^2. \]

**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 $\bot$ 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

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

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 $D_0 = g_2^{\alpha \gamma + \beta} F_2(ID)^{r'_1} H_2(T)^{r'_2}, D_1 = g_2^{-r'_1}, D_2 = g_2^{-r'_2}$ 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 $DK_{ID,T} = (D_0, D_1, D_2) \in \mathbb{G}_2^3$.

**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

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

**RIBE.Decrypt(CT, DK, 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 $(ID = ID') \land (T = T')$, then it outputs the encrypted message $M$ as $M = C \cdot (\prod_{i=0}^{2} e_{1,2}(C_i, D_i))^{-1}$. Otherwise, it outputs $\bot$.

**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 $\bot$ 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

$$T_0 = e(g_1^{\alpha d}, U_0) \cdot e(g_1^\beta, K_0) \prod_{j \in SI, j \neq d} g_1^{\alpha^{n+1-j} \cdot d} \cdot e(g_1^\beta, g_1^{\alpha d} F_1(ID)^{-r_1} \cdot \prod_{j \in SI, j \neq d} g_1^{\alpha^{n+1-j} \cdot d} \cdot e(g_1^\beta, g_1^{-r_1}) = g_2^{-\alpha d r_2}$$

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

$$D_0 = T_0 \cdot F_2(ID)^{\beta_1 \cdot H_2(T)^{\alpha d r_2}} = g_2^{\alpha^{n+1} \beta_1 F_2(ID)^{\beta_1 \cdot H_2(T)^{\alpha d r_2}} \cdot F_2(ID)^{\beta_1 \cdot H_2(T)^{\alpha d r_2}}} = g_2^{\alpha^{n+1} \beta_1 F_2(ID)^{\beta_1 \cdot H_2(T)^{\alpha d r_2}}}$$

$$D_1 = T_1 \cdot g_2^{\beta_1} = g_2^{-\alpha d r_1}, \quad D_2 = T_2 \cdot g_2^{-r_2} = g_2^{-\alpha d r_2} = g_2^{-r_2}$$

where $r_1 = \beta_1 r_1 + r_1'$ and $r_2 = \alpha d r_2 + r_2'$.

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

$$\prod_{i=0}^{2} e(C_i, D_i) = e(g_1^{\alpha d}, g_2^{\alpha^{n+1} \beta_1 F_2(ID)^{\alpha d r_2}}) \cdot e(F_1(ID)^{\beta_1 \cdot H_2(T)^{\alpha d r_2}}) \cdot e(H_1(T)^{\beta_1 \cdot H_2(T)^{\alpha d r_2}}) \cdot e(H_1(T)^{\beta_1 \cdot H_2(T)^{\alpha d r_2}})$$

12
Theorem 3.1. The above RIBE scheme is secure in the selective revocation list model under chosen plaintext
scheme of Boneh, Gentry, and Waters [8] and the HIBE scheme of Boneh and Boyen [5]. To prove the security of our RIBE scheme, we carefully combine the partitioning methods of the PKBE
3.3 Security Analysis
attacks if the
BT
Init:
T
at the time
A
sets
ID
Phase 1:
T
probability if
f
∆
̸≡
,
T
probability if
f
′
, ID = (\[\prod_{i=1}^{\mathbf{\Delta}} (g_1^{a_i}, \cdots, g_1^{a_{\mathbf{\Delta}+1}}, \cdots, g_1^{b_{\mathbf{\Delta}}}, g_1^{c_{\mathbf{\Delta}}}) \] and Z where Z = Z_0 = g_3^{\alpha+1}bc \text{ or } Z = Z_1 \in \mathbb{G}_3. \text{ Then } B \text{ that interacts with } A \text{ is described as follows:}

\textbf{Init:} A initially submits a challenge identity ID^*, a challenge time T^*, and a revoked identity set R^* at the
time T^*. \text{ It first sets a state } ST \text{ and a revocation list } RL \text{ as empty one. For each } ID \in \{ID^* \} \cup R^*, \text{ it selects an index } d \in \mathcal{N} \text{ such that } (\neg, d) \notin ST \text{ and adds } (ID, d) \text{ to } ST. \text{ Let } RI^* \subseteq \mathcal{N} \text{ be the revoked index set of } R^* \text{ at the time } T^* \text{ and } SI^* \text{ be the non-revoked index set at the time } T^* \text{ such that } SI^* = \mathcal{N} \setminus RI^*. \text{ With } SI^* \text{, B chooses random exponents } f_0, {f_{i,j}}_{1 \leq i \leq l, j \in \{0,1\}}, h_0, {h_{i,j}}_{1 \leq i \leq l, j \in \{0,1\}}, \theta \in \mathbb{Z}_p. \text{ It implicitly sets } \alpha = a, \beta = b, \gamma = \theta - \sum_{j \in SI^*} \mathbf{a}^{N+1-j} \text{ and publishes the public parameters } PP \text{ as}

\{g_1^\alpha, g_1^\beta\}_{1 \leq i, j \neq N+1 \leq 2N}, g_1^\beta = g_1^\beta, \bar{f}_1 = \left( f_{1,0} = g_1^\theta, \left( \prod_{i=1}^{\mathbf{\Delta}} f_{1,\mathbf{ID}^*[i]} \right)^{-1}, \right)_{1 \leq i \leq l, j \in \{0,1\}}, \bar{h}_1 = \left( h_{1,0} = g_1^\beta, \left( \prod_{i=1}^{\mathbf{\Delta}} h_{1,\mathbf{ID}^*[i]} \right)^{-1}, \right)_{1 \leq i \leq l, j \in \{0,1\}}, \Omega = e(g_1^\alpha, g_1^\gamma, g_1^\beta) = g_3^{\alpha+1}. \text{ For notational simplicity, we define } \Delta ID = \sum_{i=1}^l f_{i,ID^*[i]} - f_{i,ID^*[i]}, \text{ and } \Delta T = \sum_{i=1}^l (h_{i,T^*[i]} - h_{i,T^*[i]}). \text{ We have } \Delta ID \neq 0 \mod p \text{ except with negligible probability if } ID \neq ID^* \text{ since there exists at least one index } i \text{ such that } f_{i,ID^*[i]} \neq f_{i,ID^*[i]} \text{ and } f_{i,ID^*[i]} \text{ are randomly chosen. We also have } \Delta T \neq 0 \mod p \text{ except with negligible probability if } T \neq T^*. \text{ Phase 1: } A \text{ 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 \text{, then } B \text{ 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

\[K_0 = (g_1^\alpha)^\theta \left( \prod_{j \in SI^*} g_1^{\alpha+1-j} \right)^{-1} F_1(ID)^{-r_1}, K_1 = g_1^{-r_1}.\]
• Case $ID \not\in R^*$: 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 $(-,d) \not\in ST$ 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

$$K_0 = g_1^{d^b} \prod_{j \in SI \setminus \{d\}} g_1^{a^{N+1-j-d}} (g_1^{a^{N+j}} f_0/\Delta ID F_1(ID)^{-r'_1}, K_1 = (g_1^{a^N})^{-1/\Delta ID} b_1^r.$$ If this is an update key query for a time $T$, then $B$ 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

$$U_0 = (g_1^b)^\theta \left( \prod_{j \in SI \setminus SI} g_1^{-a^{N+1-j}} \prod_{j \in SI \setminus SI} g_1^{a^{N+1-j}} \right)^{-h_0/\Delta T} H_1(T)^{r'_2},$$

$$U_1 = \left( \prod_{j \in SI \setminus SI} g_1^{-a^{N+1-j}} \prod_{j \in SI \setminus SI} g_1^{a^{N+1-j}} \right)^{-1/\Delta T} b_1^r.$$ If this is a decryption key query for an identity $ID$ and a time $T$, then $B$ proceeds as follows:

• Case $ID \neq ID^*$: In this case, the simulator can use the partitioning method of Boneh and Boyen [5]. If $(ID, -) \not\in ST$, then it selects an index $d \in \mathcal{N}$ such that $(-,d) \not\in ST$ 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

$$D_0 = e((g_1^a)^{-h_0/\Delta ID} F_1(ID)^{r'_1}, g_1^{b^T}) H_2(T)^{r'_2}, D_1 = e((g_1^a)^{-1/\Delta ID} b_1^r g_1^{b^T}, D_2 = g_1^{r'_2}.$$ • 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

$$D_0 = e((g_1^a)^{-h_0/\Delta T} H_1(T)^{r'_2}, g_1^{b^N}) \cdot F_2(ID)^{r_1}, D_1 = g_1^{r'_1}, D_2 = e((g_1^a)^{-1/\Delta T} b_1^r g_1^{b^N}).$$

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

$$C = Z \cdot M_\delta', C_0 = g_1^c, C_1 = (g_1^c)^{h_0}, C_2 = (g_1^c)^{h_0}.$$ 14
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 \( \eta \) be a random bit for \( Z_\eta \). From the above simulation, we have \( \Pr[\delta = \delta'|\eta = 0] = \frac{1}{2} + \operatorname{Adv}^{\mathsf{IND-RL-CPA}}_{\mathsf{RIBE}, \mathcal{A}}(\lambda) \) since the distribution of the simulation is correct, and we also have \( \Pr[\delta = \delta'|\eta = 1] = \frac{1}{2} \) since \( \delta \) is completely hidden to \( \mathcal{A} \). Therefore we can obtain the following equation

\[
\operatorname{Adv}^{\mathsf{(3,N)-MDHE}}_{\mathcal{B}}(\lambda) = \left| \Pr[\mathcal{B}(D, Z_0) = 0] - \Pr[\mathcal{B}(D, Z_1) = 0] \right| \geq \left| \Pr[\delta = \delta'|\eta = 0] - \Pr[\delta = \delta'|\eta = 1] \right| = \frac{1}{2} + \operatorname{Adv}^{\mathsf{IND-RL-CPA}}_{\mathsf{RIBE}, \mathcal{A}}(\lambda) - \frac{1}{2} = \operatorname{Adv}^{\mathsf{IND-RL-CPA}}_{\mathsf{RIBE}, \mathcal{A}}(\lambda).
\]

This completes our proof.

Lemma 3.2. The distribution of the above simulation is correct if \( Z = Z_0 \), and the challenge ciphertext is independent of \( \delta \) in the adversary’s view if \( Z = Z_1 \).

Proof. The distribution of public parameters is correct since random exponents \( f'_i, \{h'_i, \}, \{h''_i, \}, \theta \in \mathbb{Z}_p \) are chosen. We show that the distribution of private keys is correct. In case of \( ID \in \mathbb{R}^* \), 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

\[
K_0 = g_1^{\alpha_1 \gamma} F_1(ID)^{-r_1} = g_1^{\alpha_1} \left( \prod_{j \in SI} g_1^{a^{N+1-j+d}}(f_{1,0} \prod_{i=1}^l f_{1,1, \text{ID}[\emptyset]})^{-r_1} \right) = g_1^{\alpha_1} \left( \prod_{j \in SI} g_1^{a^{N+1-j+d}}(g_1^{f_{1,0} a^{N+1} \Delta D})^{-r_1} \right).
\]

In case of \( ID \notin \mathbb{R}^* \), we have that the private key is correctly distributed from the setting \( \gamma = \theta - \sum_{j \in SI} a^{N+1-j} + r_1 = -a/\Delta D + r'_1 \) as the following equation

\[
K_0 = g_1^{\alpha_1} F_1(ID)^{-r_1} = g_1^{\alpha_1} \left( \prod_{j \in SI \setminus \{d\}} g_1^{a^{N+1-j+d}}(g_1^{f_{1,0} a^{N+1} \Delta D})^{-r_1} \right) = g_1^{\alpha_1 \gamma} \left( \prod_{j \in SI \setminus \{d\}} g_1^{a^{N+1-j+d}}(g_1^{f_{1,0} a^{N+1} \Delta D})^{-r_1} \right).
\]

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 \( \gamma = \theta - \sum_{j \in SI} a^{N+1-j} + \sum_{j \in SI \setminus SI^*} a^{N+1-j} / \Delta T + r'_2 \) as the following equation

\[
U_0 = (g_1^{\gamma} \prod_{j \in SI} g_1^{a^{N+1-j}})^{b_h} H_1(T)^{r_2} = (g_1^{\gamma} \left( \prod_{j \in SI} g_1^{a^{N+1-j}} \right)^{b_h} \prod_{j \in SI} g_1^{a^{N+1-j}})^{b_h} \left( \prod_{j \in SI} h_0^{a^{N+1-j}} \right)^{b_h} (h_0^{a^{N+1-j} / \Delta T})^{b_h} \left( \prod_{j \in SI \setminus SI^*} a^{N+1-j} / \Delta T + r'_2 \right) = (g_1^{\gamma} \prod_{j \in SI \setminus SI^*} g_1^{a^{N+1-j}})^{b_h} H_1(T)^{r_2}.
\]
\[ U_1 = g_1^{r_2} = \left( \prod_{j \in \mathcal{S} \setminus \mathcal{S}_I} g_1^{-a^{N+1-j}} \prod_{j \in \mathcal{S}_I} g_1^{a^{N+1-j}} \right)^{-1/\Delta T} g_1^{r_2}. \]

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

\[
U_0 = (g_1^\gamma \prod_{j \in \mathcal{S}_I^*} g_1^{a^{N+1-j}})^\beta \cdot H_1(T^*)^{r_2} = \left( g_1^\theta \left( \prod_{j \in \mathcal{S}_I^*} g_1^{a^{N+1-j}} \right)^{-1} \cdot \prod_{j \in \mathcal{S}_I^*} g_1^{a^{N+1-j}} \right)^b H_1(T^*)^{r_2} = (g_1^\theta)^b H_1(T^*)^{r_2}.
\]

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 \( \log_{g_2} F_2(ID) = \alpha N ID \) and \( r_1 = (-\alpha/\Delta ID + \beta) b \) as the following equation

\[
D_0 = g_2^{a^{N+1}} F_2(ID)^{r_1} H_2(T)^{r_2} = g_2^{a^{N+1}} b \left( f_2, 0 \prod_{i=1}^l f_{2,1,ID[i]} \right)^{-a/\Delta ID + \beta} H_2(T)^{r_2} = e \left( g_1^{a^{N+1}} g_1^{\Delta ID} \right)^{-a/\Delta ID + \beta} H_2(T)^{r_2} = e \left( g_1^{a^{N+1}} \right)^{-a/\Delta ID + \beta} H_2(T)^{r_2},
\]

\[
D_1 = g_2^{r_1} = e(g_1, g_1)^{-a/\Delta ID} = e(g_1^\alpha)^{-1/\Delta ID} S_1^{r_1} g_1^{b^N}.
\]

In case of \( ID = ID^* \), the decryption key is correctly distributed from the setting \( \log_{g_2} H_2(T) = b\Delta T \) and \( r_2 = (-\alpha/\Delta T + \beta) a^N \) as the following equation

\[
D_0 = g_2^{a^{N+1}} F_2(ID)^{r_1} H_2(T)^{r_2} = g_2^{a^{N+1}} b \left( f_2, 0 \prod_{i=1}^l f_{2,1,ID[i]} \right)^{-a/\Delta T + \beta} a^N H_2(T)^{r_2} = e \left( g_1^{a^{N+1}} g_1^{\Delta T} \right)^{-a/\Delta T + \beta} H_2(T)^{r_2} = e \left( g_1^{a^{N+1}} \right)^{-a/\Delta T + \beta} H_2(T)^{r_2},
\]

\[
D_2 = g_2^{r_2} = e(g_1, g_1)^{-a/\Delta T} a^N = e(g_1^\alpha)^{-1/\Delta T} S_1^{r_2} g_1^{b^N}.
\]

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

\[
C = \Omega^\delta \cdot M_\delta = g_3^{a^{N+1}b} \cdot M_\delta = Z_0 \cdot M_\delta, \quad C_0 = g_1^c = g_1^c,
\]

\[
C_1 = \left( g_1^{\delta} \prod_{i=1}^l f_{1,1,ID[i]} \prod_{i=1}^l f_{1,1,ID[i]}^{-1} \right)^c = (g_1^c)^{\delta}, \quad C_2 = \left( g_1^{\delta} \prod_{i=1}^l h_{1,1,ID[i]} \prod_{i=1}^l h_{1,1,ID[i]}^{-1} \right)^c = (g_1^c)^{\delta}.
\]

Otherwise, the component \( C \) of the challenge ciphertext is independent of \( \delta \) in the \( A \)'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 \( O(k^3\lambda^5 \log(k\lambda)) \approx 7.1 \times 10^{11} \) and the bit size of group elements is \( O(k^2\lambda^3) \approx 4.6 \times 10^6 \). 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 \( O(k^3\lambda \log^2(k\lambda)) \approx 1.8 \times 10^5 \) and the bit size of the group elements is \( O(k^2\lambda \log^2(k\lambda)) \approx 4.6 \times 10^4 \). 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 \( \lambda \). 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 \( \lambda \). 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:

\[ \text{RIBE.Setup}(1^\lambda, N) \text{: This algorithm takes as input a security parameter } 1^\lambda \text{ and the maximum number } N \text{ of users. It generates a } 2n + l - 2 \text{-leveled multilinear group } \mathcal{G} = (\mathcal{G}_1, \ldots, \mathcal{G}_{2n+l-2}) \text{ of prime order } p. \]
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 $\tilde{f}_k = (f_k,0, \{f_{k,i,j}\}_{1 \leq i \leq l_1, j \in \{0,1\}})$ and $\tilde{h}_k = (h_k,0, \{h_{k,i,j}\}_{1 \leq i \leq l_2, j \in \{0,1\}})$ for a level $k \geq n - 1$. Note that we can obtain $\tilde{f}_k$ and $\tilde{h}_k$ from $f_{n-1}$ and $h_{n-1}$ 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. It computes $g^r_1$ 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

$$SK_{ID} = \left( K_0 = g^d g_1^{r_1}, K_1 = g_1^{-r_1} \right) \in \mathbb{G}_{n-1}^2.$$

### RIBE.GenKey($ID, MK, ST, PP$)

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 $g^{d^i}_{n-1}$ 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

$$SK_{ID} = \left( K_0 = g^{d^i} g_{n-1}^{r_1}, K_1 = g_{n-1}^{-r_1} \right) \in \mathbb{G}_{n-1}^2.$$

### RIBE.UpdateKey($T, RL, MK, ST, PP$)

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 $RI \subseteq N$ of the revoked identity set $R$ by using the state $ST$ since $ST$ contains $(ID, d)$. It also defines the non-revoked index set $SI = N \setminus RI$. Note that $|SI|$ is polynomial since $N$ is polynomial.

2. It computes $\{g^{\alpha^j}_{n-1}\}_{j \in SI}$ 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

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

### RIBE.DeriveKey($UK_{T,R}, PP$)

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 $\bot$ 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 \( SI = N \setminus RI \). Note that \( |SI| \) is polynomial since \( N \) is polynomial.

2. It computes \( g_\ell^{s_d} \cdot \{ g_{n-1}^{\alpha_{2i-1+j\cdot d}} \}_{j \in SI, j \neq d} \) 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

\[
T_0 = e(g_\ell^{s_d}, U_0) \cdot e(g_\ell^\beta, K_0 \prod_{j \in SI, j \neq d} g_{n-1}^{\alpha_{2i-1+j\cdot d}})^{-1}, \quad T_1 = e(g_\ell^\beta, K_1), \quad T_2 = e(g_\ell^{s_d}, U_1).
\]

3. Next, it chooses random exponents \( r'_1, r'_2 \in \mathbb{Z}_p \) and re-randomizes these components as

\[
D_0 = T_0 \cdot F_{n+l-1}(ID)^r_1 H_{n+l-1}(T)^{r_2}, \quad D_1 = T_1 \cdot g_{n+l-1}^{-r'_1}, \quad D_2 = T_2 \cdot g_{n+l-1}^{-r'_2}.
\]

Note that these components are formed as

\[
D_0 = g_{n+l-1}^{\alpha_{2i-1+j\cdot d}}, \quad D_1 = g_{n+l-1}^{-r'_1}, \quad D_2 = g_{n+l-1}^{-r'_2}
\]

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 \( DK_{ID,T} = (D_0, D_1, D_2) \in \mathbb{G}_n^{3} \).

**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

\[
CT_{ID,T} = \left( C = \Omega^s \cdot M, \quad C_0 = g_{n-1}^s, \quad C_1 = F_{n-1}(ID)^{s}, \quad C_2 = H_{n-1}(T)^{s} \right) \in \mathbb{G}_{2n+l-2} \times \mathbb{G}_n^{3}.
\]

**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 \((ID = ID') \land (T = T')\), then it outputs the encrypted message \( M \) as \( M = C \cdot \left( \prod_{i=0}^{2} e(C_i, D_i) \right)^{-1} \). 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 \((ID, T) \notin RL \), 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** (\cite{12}). Using group multiplications and pairing operations on the \( g_1^{\alpha_{i}} \) for \( i \in [0, n] \), it is possible to compute \( g_\ell^{\alpha_{j}} \) for \( j \in [1, 2^n - 2] \) of weight exactly \( l \), \( g_{n-1}^{\alpha_{2i-1+j}} \) for \( j \in [1, 2^n - 2] \) of weight exactly \( l \), and \( g_{n-1}^{\alpha_{2i-1+j\cdot u}} \) 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

\[
T_0 = e(g_\ell^{s_d}, U_0) \cdot e(g_\ell^\beta, K_0 \prod_{j \in SI, j \neq d} g_{n-1}^{\alpha_{2i-1+j\cdot d}})^{-1}
\]

\[
= e(g_\ell^{s_d}, g_{n-1}^\beta \prod_{j \in SI} g_{n-1}^{\alpha_{2i-1+j\cdot d}}) \cdot e(g_\ell^\beta, g_{n-1}^\alpha F_{n-1}(ID)^{r_1} \cdot \prod_{j \in SI, j \neq d} g_{n-1}^{\alpha_{2i-1+j\cdot d}})^{-1}
\]
where \( RI \) is the revoked index set of \( R \) and \( SI = N \setminus RI \). Next, a decryption key is correctly derived from these components by performing re-randomization as

\[
D_0 = T_0 \cdot F_{n+l-1}(ID)^{r_1} H_{n+l-1}(T)^{r_2} \\
= g_{n+l-1}^{\alpha^{n-1} \beta} F_{n+l-1}(ID)^{r_1} H_{n+l-1}(T)^{r_2} \\
= g_{n+l-1}^{\alpha^{n-1} \beta} F_{n+l-1}(ID)^{r_1^*} H_{n+l-1}(T)^{r_2^*} \\
D_1 = T_1 \cdot g_{n+l-1}^{\alpha^{n-1} \beta} = g_{n+l-1}^{\alpha^{n-1} \beta} \\
D_2 = T_2 \cdot g_{n+l-1}^{\alpha^{n-1} \beta} = g_{n+l-1}^{\alpha^{n-1} \beta}
\]

Let \( C_{ID,T} \) be a ciphertext for an identity \( ID \) and a time \( T \), and \( D_{ID,T} \) be a decryption key for an identity \( ID' \) and a time \( T' \). If \( \langle ID, ID' \rangle \cap \langle T, T' \rangle \neq \emptyset \), then the decryption algorithm correctly outputs an encrypted message by the following equation.

\[
\bigg(\prod_{i=0}^{2} e(C_i, D_i) \bigg) = e(g_{n-1}^{x}, g_{n-1}^{\alpha^{n-1} \beta} F_{n+l-1}(ID)^{r_1} H_{n+l-1}(T)^{r_2}) \cdot e(F_{n-1}(ID)^{s}, g_{n+l-1}^{r_1^*}) \cdot e(H_{n-1}(T)^{s}, g_{n+l-1}^{r_2^*}) \\
= e(g_{n-1}^{x}, g_{n-1}^{\alpha^{n-1} \beta} F_{n+l-1}(ID)^{r_1}) \cdot e(g_{n-1}^{x}, H_{n+l-1}(T)^{r_2}) \\
= e(g_{n-1}^{x}, g_{n-1}^{\alpha^{n-1} \beta}) = (g_{2n+l-2}^{2})^{x} = \Omega^{x}
\]

### 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 \((k,n,l)\)-MDHE assumption holds where \( N = 2^n - 2 \) is the maximum number of users and \( k = 2n + l - 2 \). That is, for any PPT adversary \( A \), we have that \( \text{Adv}_{RIBE,A}^{IND \text{-} RIBE \text{-} CPA} (\lambda) \leq \text{Adv}_{RIBE}^{(k,n,l) \text{-} MDHE} (\lambda) \).

**Proof:** Suppose there exists an adversary \( A \) that attacks the above RIBE scheme with a non-negligible advantage. A simulator \( B \) that solves the MDHE assumption using \( A \) is given: a challenge tuple \( D = (g_1, g_1^\beta, \ldots, g_1^{2n}, g_l^{2n-1}, g_l^\beta, g_{n-1}^{2n-1}) \) and \( Z \) where \( Z = Z_0 = g_{2n+l-2}^{2n-1} \) or \( Z = Z_1 = g_{2n+l-2} \). Then \( B \) that interacts with \( A \) is described as follows:

**Init:** \( A \) 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 \( d \in N \) with Hamming weight \( l \) such that \((-d) \notin ST \) and adds \( (ID, d) \) to \( ST \). Let \( RI^* \subseteq N \) 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 \( SI^* = N \setminus RI^* \).

**Setup:** \( B \) first chooses random exponents \( f_l, \{ f_{ij} \}_{0 \leq i, j \leq n} \), \( h_l, \{ h_{ij} \}_{0 \leq i, j \leq n} \) and publishes the public parameters \( PP \) as

\[
\{ g_1^{2l}, g_1^{2^l} \}_{0 \leq i \leq n}, g_l^{2n-1} = g_l^{2n-1}
\]

20
For notational simplicity, we define \( ID \) if this is a private key query for an identity \( ID \). If this is a time update key query for a time \( T \), it selects a random exponent \( r \). Case \( ID \) are randomly chosen. We also have \( AT \) if \( T \neq T^* \).

**Phase 1:** \( A \) 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 \( B \) 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
  \[
  K_0 = (g_0^{d'})^\theta \prod_{j \in ST} (g_0^{(2^n-1)d})^{r_1} F_{n-1}(ID)^{-r_1}, \quad K_1 = g_n^{-r_1}.
  \]

- **Case \( ID \not\in R^* \):** In this case, we have \( ID \not\in R^* \) from the restriction of Definition 2.2. It first selects an index \( d \in N \) such that \((-d, d) \not\in ST \) and adds \((ID, d)\) to \( ST \). Next, it selects a random exponent \( r_1' \in \mathbb{Z}_p \) and creates a private key \( SK_{ID} \) by implicitly setting \( r_1 = -a/\Delta T + r_1' \) as
  \[
  K_0 = g_n^{a/r_1'} \prod_{j \in ST' \setminus (d)} (g_n^{2^n-1d})^{r_1} F_{n-1}(ID)^{-r_1'}, \quad K_1 = (g_n^{a/r_1'})^{-1/\Delta T} g_n^{-r_1'}.
  \]

If this is an update key query for a time \( T \), then \( B \) 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 \( SI = N \setminus RI \). Next, it selects a random exponent \( r_2' \in \mathbb{Z}_p \) and creates an update key \( UK_{T,R} \) by implicitly setting \( r_2 = -(-\sum_{j \in SI \setminus SL} a_{2^n-1j} + \sum_{j \in SI \setminus SL} a_{2^n-1j} )/\Delta T + r_2' \) as
  \[
  U_0 = (g_n^{a/r_2'})^\theta \prod_{j \in SI \setminus SL} (g_n^{a_{2^n-1j}})^{-h_0/\Delta T} H_{n-1}(T)^{r_2'}, \quad U_1 = (g_n^{a/r_2'})^{-1/\Delta T} g_n^{-r_2'}.
  \]

- **Case \( T = T^* \):** In this case, we have \( R = R^* \). For each \( ID \in R^* \), it adds \((ID, T^*)\) to \( RL \) if \((ID, T^*) \not\in RL \) 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
  \[
  U_0 = (g_n^{a/r_2})^\theta H_{n-1}(T^*)^{r_2}, \quad U_1 = g_n^{-r_2}.
  \]

If this is a decryption key query for an identity \( ID \) and a time \( T \), then \( B \) proceeds as follows:
• **Case ID ≠ 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

\[
D_0 = e\left((g_{n-1}^a)^{-f_0/\Delta ID}F_{n-1}(ID)^{r'_1}, g_{l_1}^b\right) \cdot H_{n+l-1}(T)^{r_2},
\]
\[
D_1 = e\left((g_{n-1}^a)^{-1/\Delta ID}g_{n-1}^{r'_1}, g_{l_1}^b\right), D_2 = g_{n+l-1}^{r_2}.
\]

• **Case ID = ID**: In this case, we have \(T \neq T^*\) from the restriction of Definition \ref{def:challenge}. It selects random exponents \(r_1, r'_2 \in \mathbb{Z}_p\) and creates a decryption key \(DK_{ID,T}\) by implicitly setting \(r_2 = (-a/\Delta T + r'_2)a^{r_2-2}\) as

\[
D_0 = e\left((g_{l_1}^a)^{-f_0/\Delta T}H_{l}(T)^{r'_2}g_{2n-2}^{r_2-2} \cdot F_{n+l-1}(ID)^{r'_1}, D_1 = g_{n+l-1}^{r_1},
\]
\[
D_2 = e\left((g_{l_1}^a)^{-1/\Delta T}g_{l_1}^{r'_2}, g_{2n-2}^{r_2-2}\right).
\]

Note that it can computes \(H_{l}(T)\) since \(g_{l_1}^b\) is given in the assumption.

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

\[
C = Z \cdot M_{\delta}^s, C_0 = g_{n-1}^c, C_1 = (g_{n-1}^c)^{f_0}, C_2 = (g_{n-1}^c)^{b_0}.
\]

**Phase 2**: Same as Phase 1.

**Guess**: Finally, \(A\) outputs a guess \(\delta' \in \{0, 1\}\). \(B\) 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 \ref{lem:security}

except that it uses multilinear maps and the Claim \ref{claim:security}. 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 \cite{12}. If we use the improved multilinear maps of Langlois et al. \cite{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 \(2.4 \times 10^5\). In the BGK-RIBE scheme \cite{3}, the bit size of a private key and an update key is approximately \(3.2 \times 10^2\) and \(1.6 \times 10^6\) respectively, where the number of revoked users is \(r = 2^{10}\). Therefore, our second RIBE scheme equipped with the currently best leveled multilinear maps \cite{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 \cite{Boneh2004} and the BGW-PKBE scheme \cite{Boneh2005}, 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 \cite{Boneh2012}, 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 \cite{Seo2008} 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


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 $S = \{S'_i(\alpha) \subset \{0,1\}^* : i \in [0,k], \alpha \in R\}$, 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 $\oplus$ and $\ominus$ (on $\{0,1\}^*$) such that for every $\alpha_1, \alpha_2 \in R$, every $i \in [0, \ell]$, and every $u_1 \in S_i^\alpha$ and $u_2 \in S_i^\beta$, 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 $0 \leq i_1 + i_2 \leq \ell$, and every $u_1 \in S_i^{\alpha_1}$ and $u_2 \in S_i^{\alpha_2}$, it holds that $u_1 \times u_2 \in S_i^{\alpha_1 \times \alpha_2}$ where $\alpha_1 \times \alpha_2$ is multiplication in $R$.

The $k$-graded encoding system for a ring $R$ includes a system for sets $S = \{S_i^\alpha \subseteq \{0,1\}^* : i \in [0, \ell], \alpha \in R\}$. The set $S_i^\alpha$ consists of the “level-$i$ encodings of $\alpha$”. 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 $\text{InstGen}(1^k, \ell)$ takes as inputs the parameters $\lambda$ and $k$, and outputs $(\text{params}, p_{\text{inst}})$, where $\text{params}$ is a description of a $k$-Graded Encoding System as above, and $p_{\text{inst}}$ is a zero-test parameter.

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

**Encoding.** The (possibly randomized) $\text{enc}(\text{params}, a)$ 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 $\alpha$.

**Re-Randomization.** The randomized $\text{rerand}(\text{params}, i, u)$ 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 $\text{rerand}(\text{params}, i, u_1)$ and $\text{rerand}(\text{params}, i, u_2)$ are nearly the same.

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

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

**Zero-test.** The procedure $\text{isZero}(\text{params}, p_{\text{inst}}, u)$ 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 $\text{ext}(\text{params}, p_{\text{inst}}, u)$ outputs $s \in \{0,1\}^\lambda$, such that:

1. For any $\alpha \in R$ and two $u_1, u_2 \in S_k^\alpha$, $\text{ext}(\text{params}, p_{\text{inst}}, u_1) = \text{ext}(\text{params}, p_{\text{inst}}, u_2)$.

2. The distribution $\{\text{ext}(\text{params}, p_{\text{inst}}, u) : \alpha \in R, u \in S_k^\alpha\}$ is nearly uniform over $\{0,1\}^\lambda$.

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

27
A.2 Construction

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

RIBE.Setup\((1^k, N)\): This algorithm obtains \((\text{params}, p_\sigma)\) by running \text{InstGen}\((1^k, 1^\lambda)\). Note that \text{params} includes a level 1 encoding of 1, which is denoted as \( g_1 \). It chooses random encodings \( f_0, \{f_{i,j} \}_{1 \leq i \leq l_1, j \in \{0, 1\}} \), \( h_0, \{h_{i,j} \}_{1 \leq i \leq l_2, j \in \{0, 1\}} \) by freshly calling \text{ samp}() and sets

\[
\begin{align*}
  f_{1,0} &= \text{rerand}(1, \text{enc}(1, f_0)), \ \{f_{1,i,j} = \text{rerand}(1, \text{enc}(1, f_{i,j}))\}_{1 \leq i \leq l_1, j \in \{0, 1\}}, \\
  h_{1,0} &= \text{rerand}(1, \text{enc}(1, h_0)), \ \{h_{1,i,j} = \text{rerand}(1, \text{enc}(1, h_{i,j}))\}_{1 \leq i \leq l_2, j \in \{0, 1\}}.
\end{align*}
\]

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

\[
PP = \left( (\text{params}, p_\sigma), \{A_j = \text{rerand}(1, \text{enc}(1, \alpha^j))\}_{1 \leq j \neq N+1 \leq 2N}, \right.
\]

\[
B = \text{rerand}(1, \text{enc}(1, \beta)), \ \tilde{f}_1, \tilde{h}_1, \ \Omega = \text{rerand}(3, \text{enc}(3, \alpha^{N+1} \beta)) \right).
\]

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 \text{ samp}() and outputs a private key by implicitly including \( ID \) and \( d \) as

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

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 \text{ samp}() and outputs an update key by implicitly including \( T \), \( R \), and \( RI \) as

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

RIBE.DeriveKey\((SK_\text{ID}, UK_{T,R}, PP)\): Let \( SK_\text{ID} = (K_0, K_1) \) 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

\[
T_0 = \text{rerand}(2, (A_d \cdot U_0 - B \cdot (K_0 + \prod_{j \in SI, j \neq d} A_{N+1-j+d}))), \quad T_1 = \text{rerand}(2, B \cdot K_1), \quad T_2 = \text{rerand}(2, A_d \cdot U_1).
\]

28
Next, it selects random encodings \( r'_1, r'_2 \) by freshly calling \( \text{samp()} \) and re-randomizes the temporal components as

\[
D_0 = \text{rand}(2, T_0 + (f_{2,0} + \sum_{i=1}^{l_1} f_{2,i,ID[i]} \cdot r'_1 + (h_{2,0} + \sum_{i=1}^{l_2} h_{2,i,T[i]} \cdot r'_2),
\]

\[
D_1 = \text{rand}(2, T_1 + \text{enc}(2, -r'_1)), D_2 = \text{rand}(2, T_2 + \text{enc}(2, -r'_2)).
\]

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

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

\[
CT_{ID,T} = \left( C, C_0 = \text{rand}(1, \text{enc}(1,s)), C_1 = \text{rand}(1, (f_{1,0} + \sum_{i=1}^{l_1} f_{1,i,ID[i]} \cdot s),
\]

\[
C_2 = \text{rand}(1, (h_{1,0} + \sum_{i=1}^{l_2} h_{1,i,T[i]} \cdot s)) \right).
\]

**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 \( (ID = ID') \land (T = T') \), then it computes \( C' = C_1 \cdot D_1 + C_2 \cdot D_2 \) and outputs \( M = 1 \) if \( C = C' \) by using \( \text{isZero}(p_{3,1} - C - C') \) and \( M = 0 \) otherwise. Otherwise, it outputs \( \bot \).

**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 \( \mathcal{N} = \{1,\ldots,N\}, \mathcal{I} = \{ID_1,\ldots,ID_n\} \), and \( \mathcal{T} = \{T_1,\ldots,T_m\} \). Let \( \rho_1 \) be a mapping from identity space \( \mathcal{I} \) to integers \( \{1,\ldots,n\} \) and \( \rho_2 \) be a mapping from time space \( \mathcal{T} \) to integers \( \{1,\ldots,n\} \). Our RIBE scheme for small universe is described as follows:

**RIBE.Setup\((1^k,N)\):** 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 \( v_1, \{f_{1,j}\}_{1 \leq j \leq n_1}, \{h_{1,j}\}_{1 \leq j \leq n_2} \in \mathbb{G}_1 \) 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

\[
PP = \left( PP_{MLM}, v_1, \{g_{1,j}^{\alpha}, v_{1,j}^{\alpha} \}_{1 \leq j \neq N+1 \leq 2N}, g_1^\beta, f_1, h_1, \Omega = v_3^{\alpha N+1} \right).
\]
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

\[
SK_{ID} = \left( K_0 = g_1^{\alpha_1^d r_1} f_1^{p_1(ID)}, \; K_1 = g_1^{-r_1} \right).
\]

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

\[
UK_{T,R} = \left( U_0 = \left( v_1 \prod_{j \in SI} v_1^{\alpha_1^d j - 1} \right)^{\beta_{1,p_2(T)}}, \; U_1 = g_1^{-r_2} \right).
\]

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 \( \bot \). 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 \( SI = N \setminus RI \) and derives components \( T_0, T_1, T_2, T_{0}, T_{1}, T_{2}, T_{0}, T_{1}, T_{2} \) as

\[
T_0 = e\left(g_1^{\alpha d}, U_0\right) \cdot e\left(g_1^b, K_0 \right) \prod_{j \in SI, j \neq d} v_1^{\alpha_1^d j - 1}, \; T_1 = e\left(g_1^b, K_1\right), \; T_2 = e\left(g_1^{\alpha d}, U_1\right).
\]

Next, it chooses random exponents \( r_1', r_2' \in \mathbb{Z}_p \) and re-randomizes these components as

\[
D_0 = T_0 \cdot f_2^{r_1, p_1(ID)} \cdot h_2^{r_2, p_2(T)}, \; D_1 = T_1 \cdot g_2^{-r_1'}, \; D_2 = T_2 \cdot g_2^{-r_2'}. \]

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

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

\[
CT_{ID,T} = \left( C = \Omega \cdot M, C_0 = g_1^s, C_1 = f_1^{p_1(ID)}, C_2 = h_1^{p_2(T)} \right).
\]

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 \( A \) that attacks the above RIBE scheme with a non-negligible advantage. A simulator \( B \) that solves the MDHE assumption using \( A \) is given: a challenge tuple \( D = (g_1, g_1^a, g_1^d, \ldots, g_1^{\alpha(d)}, g_1^{\alpha(d)}, \ldots, g_1^d, g_1^d, g_1^d) \) and \( Z \) where \( Z = Z_0 = g_3^{bc} \) or \( Z = Z_1 \in \mathbb{G}_3 \). Then \( B \) that interacts with \( A \) is described as follows:

**Init:** \( A \) 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 \( d \in N \) such that \( (-, d) \notin ST \) and adds \((ID, d)\) to \( ST \). Let \( RI^* \subseteq N \) 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 \( SI^* = N \setminus RI^* \).
Setup: $\mathcal{B}$ first chooses random exponents $v'_1, \{f'_j\}_{1 \leq j \leq n_1}, \{h'_j\}_{1 \leq j \leq n_2}, \theta \in \mathbb{Z}_p$. For notational simplicity, we use $\prod f'_j = \prod_{j=1}^{n_1} f'_j$. \(\prod_{j \neq k} f'_j = \prod_{1 \leq j \leq n_1, j \neq k} f'_j\). \(\prod h'_j = \prod_{j=1}^{n_2} h'_j\), and \(\prod_{j \neq k} h'_j = \prod_{1 \leq j \leq n_2, j \neq k} h'_j\). It implicitly sets $a = a, \beta = b \prod h'_j, r = \theta - \sum_{i \in S'} a^{N+1-j}$ and publishes the public parameters $PP$ as

$$v_1 = g_1^{v'_1 \prod f'_j}, \{g_1^{a^i}, v_1^{a^i} = (g_1^{a^i})^{v'_1 \prod f'_j}\}_{1 \leq i, i \neq N+1 \leq 2N}, g_1^\beta = g_1^{b \prod h'_j},$$

$$\{f_{1,i} = (g_1^{v'_1})_{i \notin p_1(ID)}, f_{1,i}^{p_1(ID)} = g_1^{\prod_{j \neq k} f'_j}\}_{1 \leq i \leq n_1, i \neq p_1(ID)}, f_1, p_1(ID) = g_1^{\prod_{j \neq k} f'_j},$$

$$\{h_{1,i} = (g_1^{b^j})_{i \notin p_2(T')}, h_{1,i}^{p_2(T')} = g_1^{\prod_{j \neq k} h'_j}\}_{1 \leq i \leq n_2, i \neq p_2(T')}, h_1, p_2(T') = g_1^{\prod_{j \neq k} h'_j},$$

$$\Omega = e(g_1^{a^r}, g_1^{\prod f'_j \prod h'_j}) = g_1^{\alpha^{x-1} b \prod f'_j \prod h'_j}.$$

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

$$K_0 = (v_1^{a^d})^\theta \left( \prod_{j \in S'} v_1^{a^{N+1-j-d}} f_{1,p_1(ID)}^{-r_1} \right), K_1 = g_1^{v_1^{a^d}}.$$ 

- **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 $r_1 = -a f_{1,p_1(ID)}^r + r'_1$ as

$$K_0 = v_1^{a^d} \prod_{j \in S' \setminus [d]} v_1^{a^{N+1-j-d}} f_{1,p_1(ID)}^{r'_1}, K_1 = (g_1^{b^d})_{p_1(ID)} g_1^{a^{r'_1}}.$$ 

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 $r_2 = -(\sum_{j \in S' \setminus SI} a^{N+1-j} + \sum_{j \in SI \setminus S'} a^{N+1-j}) h_{1,p_2(T)} f_{1,p_2(T)}^r + r'_2$ as

$$U_0 = (g_1^{b^d})_{p_1(p_2(T))}, U_1 = \left( \prod_{j \in S' \setminus SI} g_1^{a^{N+1-j}} \prod_{j \in SI \setminus S'} g_1^{a^{N+1-j}} \right)^{-h_{1,p_2(T)} f_{1,p_2(T)}^r + r'_2} g_1^{a^{r'_2}}.$$ 

- **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 exponent $r_2 \in \mathbb{Z}_p$ and creates an update key $UK_{T,R}$ as

$$U_0 = (g_1^{b^d})_{p_1(p_2(T))}, U_1 = g_1^{a^{r_2}}.$$ 

If this is a decryption key query for an identity $ID$ and a time $T$, then $\mathcal{B}$ proceeds as follows:
B.3 Construction in the GGH Framework

The three-graded encoding system is described as follows:

\[ \text{RIBE.GenKey}(\text{ID}, \text{ST}) = (\text{MK}, \text{SK}_\text{ID}, \text{RL}) \]

**Case ID \( \neq ID^* \):** If \((\text{ID}, -) \notin \text{ST}\), then it selects an index \( d \in \mathcal{N} \) such that \((-d) \notin \text{ST}\) and adds \((\text{ID}, d)\) to \(\text{ST}\). Next, it selects random exponents \( r_1', r_2 \in \mathbb{Z}_p \) and creates a decryption key \( \text{DK}_{\text{ID},T} \) by implicitly setting \( r_1 = -af_{p_1(\text{ID})} + r_1' \) as

\[ D_0 = e\left(\left( g_1^{a^v g_1^j} \right)^{f_{p_1(\text{ID})}} \right), \quad D_1 = e\left(\left( g_1^{a^v g_1^j} \right)^{f_{p_1(\text{ID})} + r_1'} \right), \quad D_2 = g_2^{-r_2}. \]

**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 \( \text{DK}_{\text{ID},T} \) by implicitly setting \( r_2 = -af_{p_1(T)} + r_2'a^N \) as

\[ D_0 = e\left(\left( g_1^{b^v g_1^j} \right)^{f_{p_1(T)}} \right), \quad D_1 = g_2^{-r_1}, \quad D_2 = e\left(\left( g_1^{b^v g_1^j} \right)^{f_{p_1(T)}}, g_2^{-r_2'a^N} \right). \]

**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 \( c = e \) as

\[ C = Z^{\prod f_j p_j} \cdot M_\delta, \quad C_0 = g_1, \quad C_1 = (g_1^{\prod f_j p_j})^{f_{p_1(\text{ID})}}, \quad C_2 = (g_1^{\prod f_j p_j})^{f_{p_1(T)}}, \]

**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.

---

### B.3 Construction in the GGH Framework

Let \( \mathcal{N} = \{1, \ldots, N\} \), \( \mathcal{T} = \{\text{ID}_1, \ldots, \text{ID}_{n_1}\} \), 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(\(1^k,N\)):** This algorithm obtains \((\text{params}, p_\gamma)\) by running \text{InstGen}(1^k, 1^2). Note that \text{params} includes a level 1 encoding of 1, which is denoted as \( g_1 \). It chooses random encodings \( v_1', \{f_j\}_1 \leq i \leq n_1 \), \( \{h_j\}_1 \leq i \leq n_2 \) by freshly calling \text{samp}() and sets

\[ \{f_1, j = \text{rand}(1, \text{enc}(1, f_j')) \}, \quad f_2, j = \text{rand}(2, g_1 \cdot f_1, j) \}_{1 \leq i \leq n_1}, \]

\[ \{h_1, j = \text{rand}(1, \text{enc}(1, h_j')) \}, \quad h_2, j = \text{rand}(2, g_1 \cdot h_1, j) \}_{1 \leq i \leq n_2}. \]

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

\[ PP = \left( (\text{params}, p_\gamma), \{A_j = \text{rand}(1, \text{enc}(1, v_1 \cdot \alpha^j)) \}_{1 \leq j, j \neq N + 1 \leq 2N}, \right) \]

\[ B = \text{rand}(1, \text{enc}(1, v_1 \cdot \beta)), \quad v_1, f_1, h_1, f_2, h_2, \quad \Omega = \text{rand}(3, \text{enc}(3, v_1 \cdot \alpha^{N+1} \beta)). \]

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

\[ SK_{\text{ID}} = \left( K_0 = \text{rand}(1, \text{enc}(1, v_1 \cdot \alpha^d \cdot \gamma) + (f_1, p_1(\text{ID})) \cdot (-r_1)), \quad K_1 = \text{rand}(1, \text{enc}(1, -r_1)) \right). \]
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 $\text{samp}()$ and outputs an update key by implicitly including $T$, $R$, and the revoked index set $RI$ as

$$UK_{T,R} = \left( U_0 = \text{rand}(1, \text{enc}(1, (v_1 \cdot \gamma + v_1 \cdot \sum_{j \in SI} \alpha^{N+1-j} \cdot \beta) + (h_1, p_2(T)) \cdot r_2)), U_1 = \text{rand}(1, \text{enc}(1, -r_2)) \right).$$

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 $\bot$. 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 $SI = N \setminus RI$ and derives components $T_0$, $T_1$ and $T_2$ as

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

Next, it selects random encodings $r_1', r_2'$ by freshly calling $\text{samp}()$ and re-randomizes the temporal components as

$$D_0 = \text{rand}(2, T_0 + (f_2, p_1(ID)) \cdot r_1' + (h_2, p_2(T)) \cdot r_2'), D_1 = \text{rand}(2, T_1 + \text{enc}(2, -r_1')), D_2 = \text{rand}(2, T_2 + \text{enc}(2, -r_2')).$$

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

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

$$CT_{ID,T} = \left( C, C_0 = \text{rand}(1, \text{enc}(1, s)), C_1 = \text{rand}(1, (f_1, p_1(ID)) \cdot s), C_2 = \text{rand}(1, (h_1, p_2(T)) \cdot s) \right).$$

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 (params, $p_{\sigma}$) by running $\text{InstGen}(1^k, 1^k)$ and chooses random encodings $a, c_1, \ldots, c_{k-1}$ by calling $\text{samp}()$. The GGH analogue of decisional ($k,l$)-MDHE assumption is that if the challenge tuple

$$D = \{(\text{params}, p_{\sigma}), \{\text{rand}(1, \text{enc}(1, a^j))\}_{1 \leq j \neq 1 \leq 2l}, \{\text{rand}(1, \text{enc}(1, c_i))\}_{1 \leq i \leq k-1}\}$$

are given, no PPT algorithm $A$ can distinguish $Z = Z_0 = \text{rand}(k, \text{enc}(k, a^l \prod_{i=1}^{k-1} c_i))$ from $Z = Z_1 = \text{rand}(k, \text{enc}(k, \text{samp}()))$ with more than a negligible advantage. The advantage of $A$ is defined as

$$\text{Adv}^{GGH (k,l)-MDHE}_A (\lambda) = | \Pr[A(D, Z_0) = 0] - \Pr[A(D, Z_1) = 0] |.$$
Assumption B.3 (GGH analogue of Decisional Three-Leveled Multilinear Diffie-Hellman Exponent, GGH (3, l)-MDHE). A challenger obtains (params, p_gl) by running InstGen(1^l, 1^3) and chooses random encodings a, b, c by calling samp(). The GGH analogue of decisional (3, l)-MDHE assumption is that if the challenge tuple

\[ D = ((params, p_g), \{\text{rerand}(1, \text{enc}(1, a^j))\}_{1 \leq j, j \neq l+1 \leq 2l}, \text{rerand}(1, \text{enc}(1, b)), \text{rerand}(1, \text{enc}(1, c))) \text{ and } Z \]

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

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 \( A \) that attacks the above RIBE scheme in graded encoding systems with a non-negligible advantage. A simulator \( B \) that solves the GGH MDHE assumption using \( A \) is given: a challenge tuple \( D = ((params, p_g), \{\text{rerand}(1, \text{enc}(1, a^j))\}_{1 \leq j, j \neq N+1 \leq 2N}, \text{rerand}(1, \text{enc}(1, b)), \text{rerand}(1, \text{enc}(1, c))) \) and \( Z \) where \( Z = Z_0 = \text{rerand}(3, \text{enc}(3, a^{N+1} bc)) \) or \( Z = Z_1 = \text{rerand}(3, \text{enc}(3, \text{samp}())) \). Then \( B \) that interacts with \( A \) is described as follows:

Init: \( A \) 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 \( (−d, d) \notin ST \) 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: \( B \) first chooses random encodings \( \{f_i\}_{1 \leq i \leq n_1}, \{h_i\}_{1 \leq i \leq n_2}, \theta \) by freshly calling samp(). It sets \( \Gamma = \text{rerand}(1, \text{enc}(1, (−\sum_{j \in SI^*} a^{N+1−j})) \) 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

\[(\text{params}, p_g), v_1 = \text{rerand}(1, \text{enc}(1, \prod f_i)),
\{A_j = \text{rerand}(1, \text{enc}(1, a^j \cdot \prod f_i))\}_{1 \leq j, j \neq N+1 \leq 2N}, B = \text{rerand}(1, \text{enc}(1, b \cdot \prod f_i \cdot \prod h_i)) \]
\{f_i, i = \text{rerand}(1, A_N \cdot \prod_{i \neq p_1(ID)} f_i)\}_{1 \leq i \leq n_1, i \neq p_1(ID*), f_i, p(ID) = \text{rerand}(1, \prod_{i \neq p_1(ID)} f_i)}
\{h_{1,i} = \text{rerand}(1, B \cdot \prod_{i \neq p_2(T)} h_i)\}_{1 \leq i \leq n_2, i \neq p_2(T, T^*), h_{1, p_2(T)} = \text{rerand}(1, \prod_{i \neq p_2(T)} h_i)}
\Omega = \text{rerand}(3, A_1 \cdot A_N \cdot B \cdot \prod f_i \cdot \prod h_i).

Phase 1: \( A \) 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 \( B \) 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 samp() and creates a private key \( SK_{ID} \) as

\[ K_0 = \text{rerand}(1, A_d \cdot \prod f_i \cdot \theta − \sum_{j \in SI^*} A_{N+1−j+d} \cdot \prod f_i + f_1, p_1(ID) \cdot (−r_1)), \]
\[ K_1 = \text{rerand}(1, \text{enc}(1, −r_1)). \]
• Case $ID \notin R^*$: In this case, we have $ID \neq ID^*$ from the restriction of Definition \cite{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 $\text{samp}()$ and creates a private key $SK_{ID}$ by implicitly setting $r_1 = -af'_{1, p_1(ID)} + r'_1$ as

$$K_0 = \text{rand}(1, A_d \cdot \prod_{j \in ST \setminus \{d\}} f'_j \cdot \theta - \sum_{j \in ST \setminus \{d\}} A_{N+1- j + d} \cdot \prod_{j \in ST \setminus \{d\}} f'_j + f_{1, p_1(ID)} \cdot r'_1),$$

$$K_1 = \text{rand}(1, A_1 \cdot (f'_{1, p_1(ID)} + \text{enc}(1, -r'_1))).$$

If this is an update key query for a time $T$, then $B$ 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 $\text{samp}()$ and creates an update key $UK_{T,R}$ by implicitly setting $r_2 = -(\sum_{j \in ST \setminus SI} a^{N+1-j} + \sum_{j \in ST \setminus SI} a^{N+1-j}) h'_{1, p_2(T)} \prod f'_i + r'_2$ as

$$U_0 = \text{rand}(1, B \cdot \theta \cdot \prod f'_i + h_{1, p_2(T)} \cdot r'_2),$$

$$U_1 = \text{rand}(1, (- \sum_{j \in ST \setminus SI} A_{N+1-j} + \sum_{j \in ST \setminus SI} A_{N+1-j}) \cdot (h'_{1, p_2(T)} \prod f'_i + \text{enc}(1, -r'_2)).$$

• 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 $\text{samp}()$ and creates an update key $UK_{T,R}$ as

$$U_0 = \text{rand}(1, B \cdot \theta \cdot \prod f'_i + h_{1, p_2(T)} \cdot r'_2),$$

$$U_1 = \text{rand}(1, \text{enc}(1, -r'_2)).$$

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

• Case $ID \neq ID^*$: If $(ID, \cdot) \notin ST$, 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 $\text{samp}()$ and creates a decryption key $DK_{ID,T}$ by implicitly setting $r_1 = (-af'_{1, p_1(ID)} + r'_1)b$ as

$$D_0 = \text{rand}(2, (A_N \cdot \prod_{i \neq p_1(ID)} f'_i \cdot r'_1) \cdot B \cdot \prod h'_i + h_{2, p_2(T)} \cdot r'_2),$$

$$D_1 = \text{rand}(2, (A_1 \cdot (f'_{1, p_1(ID)} + \text{enc}(1, -r'_1)) \cdot B),$$

$$D_2 = \text{rand}(2, \text{enc}(2, -r_2)).$$

• Case $ID = ID^*$: In this case, we have $T \neq T^*$ from the restriction of Definition \cite{2,2}. It selects random encodings $r_1, r'_2$ by freshly calling $\text{samp}()$ and creates a decryption key $DK_{ID,T}$ by implicitly setting $r_2 = (ah'_{1, p_2(T)} + r'_2) \alpha$ as

$$D_0 = \text{rand}(2, (B \cdot \prod_{i \neq p_2(T)} h'_i \cdot r'_2) \cdot A_N \cdot \prod f'_i + f_{2, p_1(ID)} \cdot r'_1),$$

$$D_1 = \text{rand}(2, \text{enc}(2, -r_2)),$$

$$D_2 = \text{rand}(2, (A_1 \cdot (h'_{1, p_2(T)} + \text{enc}(1, -r'_2)) \cdot A_N).$$

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

$$C = \text{rand}(1, Z \cdot \prod f'_i \prod h'_i),$$

$$C_0 = \text{rand}(1, \text{enc}(1, c)).$$

$$C_1 = \text{rand}(1, C_0 \cdot \prod_{i \neq p_1(ID)} f'_i),$$

$$C_2 = \text{rand}(1, C_0 \cdot \prod_{i \neq p_2(T)} h'_i).$$
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, \( A \) outputs a guess \( \delta \in \{0, 1\} \). \( B \) outputs 0 if \( \delta = 0 \) or 1 otherwise.

## 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 \( d = (d_x - 1)m + d_y \) for some \( 1 \leq d_x \leq m \) and \( 1 \leq d_y \leq m \). Let \( SI_i \) be a subset of \( \{1, \ldots, N\} \), and define \( SI'_i = SI_i \cap \{(k-1)m+1, \ldots, (k-1)m+m\} \) and \( S_k = \{x-(k-1)m| x \in SI'_k\} \subseteq \{1, \ldots, m\} \). A subset \( SI_i \) is divided to subsets \( SI_1, \ldots, SI_m \). Let \( \mathcal{N} = \{1, \ldots, N\} \), \( \mathcal{I} = \{0, 1\}^{d_y} \), and \( \mathcal{T} = \{0, 1\}^{d_x} \).

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 \( 1^\lambda \) and the maximum number \( N \) of users. It generates a 3-leveled multilinear group \( \mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3) \) of prime order \( p \). Let \( \mathbb{g}_1, \mathbb{g}_2, \mathbb{g}_3 \) 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 \( f_{1,0}, \{f_{1,i,j}\}_{1 \leq i \leq l_1, j \in \{0,1\}}, h_{1,0}, \{h_{1,i,j}\}_{1 \leq i \leq l_2, j \in \{0,1\}} \in \mathbb{G}_1 \). Let \( \vec{f}_k = (f_{k,0}, \{f_{k,i,j}\}_{1 \leq i \leq l_1, j \in \{0,1\}}) \) and \( \vec{h}_k = (h_{k,0}, \{h_{k,i,j}\}_{1 \leq i \leq l_2, j \in \{0,1\}}) \) 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 \( MK = (\alpha, \beta, \{\gamma_j\}_{1 \leq j \leq m}, \{g_1^\alpha\}_{1 \leq i, j \neq m \leq 2m}, g_1^\beta, \{g_1^{\gamma_m}\}_{1 \leq k \leq m}) \), an empty revocation list \( RL \), an empty state \( ST \), and public parameters as

\[
PP = (PP_{MLM}, \vec{f}_1, \vec{h}_1, \Omega = g_3^{\epsilon^{x+1}y}).
\]

**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

\[
SK_ID = \left(K_0 = g_1^{\epsilon^d x y} F_1(ID)^{-r_1}, K_1 = g_1^{-r_1}\right).
\]

**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, for any PPT adversary $A$, 

$$RIBE.\text{Revoke}((RIBE.\text{DeriveKey}(B, T), T')) \in RL \text{ for any } T' \leq T, \text{ then } ID' \in R.$$ 

Next, it defines the revoked index set $RI \subseteq N$ of the revoked identity set $R$ by using the state $ST$ since $ST$ contains $(ID, d)$. It also defines the non-revoked index set $SI = N \setminus RI$ 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 

$$UK_{T,R} = \left\{ \left\{ g_1^{a_1}, \cdots, g_1^{a_m} \right\} \right\} \mid g_1^a, \cdots, g_1^b \in \mathbb{Z}_p.$$ 

RIBE.\text{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 $UK_{T,R} = (\{ g_1^{a_i} \}, g_2^\beta, \{ U_{k,0}, U_{k,1} \})$ 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 $d = (d_1, d_2)$ 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 $SI = N \setminus RI$ such that $SI = SI_1 \cup \cdots \cup SI_m$ and derives temporal components $T_0, T_1$ and $T_2$ as 

$$T_0 = e(g_1^{a_1}, U_{d_1,0}) \cdot e(g_1^{\beta}, K_0, g_1^{\alpha_1+j-d_1}), \quad T_1 = e(g_1^{\beta}, K_1, T_0 + 1, g_1^{\alpha_1+j-d_1})^{-1}, \quad T_2 = e(g_1^{a_1}, U_{d_1,1}).$$

2. Next, it chooses random exponents $r'_1, r'_2 \in \mathbb{Z}_p$ and re-randomizes the temporal components as 

$$D_0 = T_0 \cdot F_2(ID)^{r'_1} \cdot H_2(T)^{r'_2}, \quad D_1 = T_1 \cdot g_2^{r'_1}, \quad D_2 = T_2 \cdot g_2^{r'_2}.$$ 

Finally, it outputs a decryption key as 

$$DK_{ID,T} = (D_0, D_1, D_2).$$

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

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

RIBE.\text{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 $A$, we have that $Adv^{\text{IND-RIBE}-\text{CPA}}_{RIBE,A} \leq Adv^{(3,m)-\text{MDHE}}_{\text{B}}$.

Proof. Suppose there exists an adversary $A$ that attacks the above RIBE scheme with a non-negligible advantage. A simulator $B$ that solves the MDHE assumption using $A$ is given: a challenge tuple $D = (g_1, g_1^{a_1}, g_1^{a_2}, \cdots, g_1^{a_{m+1}}, g_1^{a_2}, g_1^{b_1}, g_1^{b_2}, g_1^{b_3})$ and $Z$ where $Z = Z_0 = g_3^{a_{m+1}+bc}$ or $Z = Z_1 \in \mathbb{G}_3$. Then $B$ that interacts with $A$ is described as follows:

Init: $A$ 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 $d \in N$ such that $(\cdot, d) \notin ST$ and adds $(ID, d)$ to $ST$. Let $RI^* \subseteq N$ 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 $SI^* = N \setminus RI^*$. Note that $SI^*$ is divided to subsets $SI_1^*, \cdots, SI_m^*.$
Setup: 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

$$\{ g_1^{\alpha_i} = s_i^{\alpha_i} \}_{1 \leq j, i \neq m+1 \leq 2m}, g_1^\beta = s_1^\beta, \{ s_i^\gamma = g_1^{\theta_k} \prod_{j \in SI_k} (g_1^{a^{m+1-j}})^{-1} \}_{1 \leq k \leq m}.$$ 

Next, it selects random exponents $f'_i, \{ f'_i \}_{1 \leq i \leq l, i \neq 0(1), h'_i, \{ h'_i \}_{1 \leq j \leq l, i \neq 0(1)} \in \mathbb{Z}_p$ and publishes the public parameters $PP$ as

$$f_i = (f_{1,0} - g_1^{l_1} \prod_{i=1}^{l_1} f_{1,i,ID^*})^{-1}, \{ f_{1,i,j} = (g_1^{a_{ij}})^{f_{i,j}} \}_{1 \leq i, j \leq l, i \neq 0(1)}, \ h_i = (h_{1,0} - g_1^{l_2} \prod_{i=1}^{l_2} h_{1,i,T^*})^{-1}, \{ h_{1,i,j} = (g_1^{a_{ij}})^{h_{i,j}} \}_{1 \leq i, j \leq l, i \neq 0(1)}, \ 

\Omega = e(g_1^\alpha, g_1^\beta, g_1^\gamma) = g_3^{a+b}.$$ 

For notational simplicity, we define $\Delta ID = \sum_{i=1}^{l_1} (f_{1,i, ID^*} - f_{1,i, ID^*})$ and $\Delta T = \sum_{i=1}^{l_2} (h_{1,i, T^*} - h_{1,i, T^*})$.

**Phase 1:** 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 $B$ 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

$$K_0 = (g_1^\alpha)^{\theta_1} \prod_{j \in SI_{dp}} g_1^{a_{m+1-j} + d_x} F_1(ID)^{-r_1}, K_1 = g_1^{r_1}.$$ 

**Case $ID \notin R^*$:** In this case, we have $ID \notin R^*$ 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

$$K_0 = (g_1^\alpha)^{\theta_1} \prod_{j \in SI_{dp} \setminus \{d\}} g_1^{a_{m+1-j} + d_x} (g_1^\beta)^{f_{i,j} / \Delta ID} F_1(ID)^{-r'_1}, K_1 = (g_1^\alpha)^{-1/\Delta ID} g_1^{a_{m+1-j} / \Delta T}.$$ 

If this is an update key query for a time $T$, then $B$ 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 $SI = \mathcal{N} \setminus RI$. Note that $SI$ is divided to $SI_1, \ldots, SI_m$. Next, it selects random exponents $r''_{1,1}, \ldots, r''_{1,m} \in \mathbb{Z}_p$ and creates an update key $UK_{T,R}$ by implicitly setting $\{ r''_{1,k} = -\left( -\sum_{j \in SI_k \setminus SI_0} a_{m+1-j} + \sum_{j \in SI_k \setminus SI_0} a_{m+1-j} \right) / \Delta T + r''_{1,k} \}$ as

$$\{ g_1^{\alpha_i} \}_{1 \leq j, i \neq m+1 \leq 2m}, g_1^\beta, \ 

\{ U_{k,0} = (g_1^\beta)^{\theta_1} \prod_{j \in SI_k \setminus SI_0} g_1^{-a_{m+1-j}} \prod_{j \in SI_k \setminus SI_0} g_1^{a_{m+1-j}} - h'_i / \Delta T H_1(T)^r_{2,k}, \ 

U_{k,1} = \left( \prod_{j \in SI_k \setminus SI_0} g_1^{-a_{m+1-j}} \prod_{j \in SI_k \setminus SI_0} g_1^{a_{m+1-j}} \right)^{-1/\Delta T} g_1^{a_{m+1-j}} \}_{1 \leq k \leq m}.$$ 

38
We define the generic multilinear groups by following the generic group model \cite{GGR:01,GG:03}. Let new assumptions in generic multilinear groups.

**D Security in Generic Multilinear Groups**

Encode groups is given the following oracles: \(\xi\) integer. Let \(p\) \(Z\) • \(Z\)  

\[\xi \in \{H \left( T^* \right)^{\gamma} \}, \ U_{k,0} = g_1^{b_i} H_1 \left( T^* \right)^{\gamma}, \ U_{k,1} = g_1^{-r_k} \}

\[1 \leq k \leq m.\]

If this is a decryption key query for an identity \(ID\) and a time \(T\), then \(B\) 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

\[D_0 = e^{\left( g_1^{a_1} \right)^{-d} / \Delta ID} F_1(ID)^{r'_1}, g_1^{b_i} H_2(T)^{r_2}, D_1 = e^{\left( g_1^{a_1} \right)^{-1/\Delta ID} g_1^{b_i}}, D_2 = g_2^{r_2}.\]

- **Case \(ID = ID^*\):** In this case, we have \(T \neq T^*\) from the restriction of Definition \cite{GG:03}. 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

\[D_0 = e^{\left( g_1^{a_1} \right)^{-d} / \Delta T} H_1 \left( T^* \right)^{r'_2}, g_1^{a_1} F_2(ID)^{r_1}, D_1 = g_2^{r_1}, D_2 = e^{\left( g_1^{a_1} \right)^{-1/\Delta T} g_1^{b_i}}.\]

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

\[C = Z \cdot M_0^\delta, \ C_0 = g_1^c, \ C_1 = (g_1^c)^{\delta}, \ C_2 = (g_1^c)^{\bar{\delta}}.\]

**Phase 2:** Same as Phase 1.

**Guess:** Finally, \(A\) outputs a guess \(\delta' \in \{0, 1\}\). \(B\) 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 \cite{GGR:01,GG:03}. Let \(k\) be the target integer. Let \(\xi : \mathbb{Z}_p \times \mathbb{Z} \rightarrow \{0, 1\}^m\) 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 \(\bot\). Note that the generator \(g_i\) for the group \(\mathbb{G}_i\) can be obtained as **Encode** \((1, i)\).

**Mult** \((\xi_1, \xi_2, b)\) 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 \(\bot\).

**Pair** \((\xi_1, \xi_2)\) 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 \(\bot\).
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 \( \lambda \)-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 \( a^{2n-1} bc \) where \( a, b, \) and \( c \) are variables. We need \( n \) polynomial multiplications (pairings) to obtain \( a^{2n-1} \) since \( \{a^{2i}\}_{i \in [0,n]} \) are only given in the assumption. Thus the target polynomial \( a^{2n-1} bc \) 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 \( \lambda \)-bit security, we can set \( p \approx 2^{3\lambda} \) instead of \( p \approx 2^\lambda \).