Identity-based Broadcast Encryption with Efficient Revocation

Aijun Ge[1,2,3], Puwen Wei[1]

[1]Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, China
[2] State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China
[3] Henan Key Laboratory of Network Cryptography Technology, Zhengzhou, China
{geaijun@163.com, pwei@sdu.edu.cn}

Abstract. Identity-based broadcast encryption (IBBE) is an effective method to protect the data security and privacy in multi-receiver scenarios, which can make broadcast encryption more practical. This paper further expands the study of scalable revocation methodology in the setting of IBBE, where a key authority releases a key update material periodically in such a way that only non-revoked users can update their decryption keys. Following the binary tree data structure approach, a concrete instantiation of revocable IBBE scheme is proposed using asymmetric pairings of prime order bilinear groups. Moreover, this scheme can withstand decryption key exposure, which is proven to be semi-adaptively secure under chosen plaintext attacks in the standard model by reduction to static complexity assumptions. In particular, the proposed scheme is very efficient both in terms of computation costs and communication bandwidth, as the ciphertext size is constant, regardless of the number of recipients. To demonstrate the practicality, it is further implemented in `Charm`, a framework for rapid prototyping of cryptographic primitives.

Keywords: Broadcast encryption, revocation, asymmetric pairings, provable security, constant size ciphertext

1 Introduction

Broadcast encryption (BE), first introduced by Fiat and Naor [13], is a cryptographic paradigm that enables delivering encrypted content over a broadcast channel in a way that only qualified users are able to decrypt the content. For a BE in the public key setting, there is a dealer which is empolyed to generate and distribute decryption keys for users. A sender can encrypt to a set of receivers by choosing their public keys adaptively, and the encrypted data can be decrypted only by the user with the private key in the set of receivers. A BE scheme is collusion resistant if no information about the encrypted data is leaked, even if all users that are not qualified collude. BE has a wide range of applications such as pay-TV, encrypted file systems and digital right management.

Identity-based encryption (IBE) is an advanced form of public key encryption in which the public key of a user is some unique information about the identity of the user (e.g., a user's IP or email address). Moreover, as public keys are derived from identifiers, IBE scheme eliminates the need for a public key infrastructure (PKI). In the IBE system, a trusted third party called the private key generator (PKG) can generate the corresponding secret keys associated with each user's public identities. A sender who has access to the public parameters of the system can encrypt a message using the receiver's identity as the public key, and only the intended receiver who obtains its decryption key from PKG can decrypt.

Identity-based broadcast encryption (IBBE) can be seen as a natural generalization of IBE, i.e., BE in the identity-based setting, which recognizes the users in a BE scheme with their identities, instead of indexes assigned by the system. The number of valid identities in the IBBE scheme can be exponential with the security parameter, while the number of public keys in the public key broadcast setting is only polynomial with the security parameter. IBBE is an effective method to protect the data security and privacy in multi-receiver scenarios. In an IBBE scheme, a sender can broadcast an encrypted message to any set of intended users, which is called privileged set. If the size of the privileged set is 1, the resulting IBBE scheme would be an IBE scheme obviously. For the trivial solution to construct an IBBE scheme which encrypts the message once for each identity using an IBE scheme, the resulting ciphertext would be linear in the privileged set, which is inefficient especially for a large set of receivers.

In 2007, Delerablée [11] presented the first IBBE scheme with constant size ciphertext, though it is only weak selective-ID secure in the random oracle model. This construction makes use of the hybrid encryption paradigm: key encapsulation mechanism (KEM) and data encapsulation mechanism (DEM) framework where the broadcast ciphertext only encrypts a short symmetric key used to encrypt the long messages, which is also adopted by most BE schemes. Very recently, Ramanna [31] proposed a novel IBBE scheme with constant size ciphertext that can achieve adaptive security in the standard model.

One desirable functionality of multi-user cryptosystems is the support for membership revocation. For example, malicious users should be driven out immediately from the system, and even for the honest users should be revoked if their private keys get stolen or lost. Key revocation is well studied in BE such as [28,19]. However, realizing efficient user revocation mechanism in the IBE setting turned out to be very challenging. Compared with traditional public key encryption in the PKI setting, IBE simplifies the key management problem by avoiding public key certificates. Therefore, users cannot be easily revoked by digital certificates and certificate revocation lists. As a result, the key revocation problem in IBE is not as simple as in tradition PKI setting.

The first practical IBE scheme, proposed by Boneh and Franklin [5] from the Weil pairing, also suggested a straightforward revocation method for IBE schemes: dividing the lifetime of the system into discrete time periods and refreshing the private key for non-revoked users periodically. Unfortunately, this approach is not scalable and very inefficient because all non-revoked users should

update their private keys via a secure channel, and the workload on the PKG grows linearly in the number of non-revoked users. To address this problem, Boldyreva et al. [6] proposed a scalable revocable IBE (RIBE) scheme which employed the tree based revocation techniques from [28] to reduce the PKG's workload to only logarithmic (instead of linear) in the number of users. Moreover, each non-revoked user can derive a decryption key from the public update key, while revoked users cannot compute their decryption keys. There is no secure channel that is required for non-revoked users to update their private keys.

After the work of Boldyreva et al. [6], Seo and Emura [33] introduced a new security notion called decryption key exposure resistance (DKER), which can better capture the the realistic threat of IBE system. Generally speaking, this security definition can guarantee that the confidentiality of ciphertexts is not compromised even if a user's decryption key at some periods has been exposed. Though DKER seems to be a natural security notion, Seo and Emura have proved that Boldyreva et al.'s RIBE scheme is vulnerable against decryption key exposure. Using Boldyreva et al.'s revocation methodology, Seo and Emura [33] also proposed the first RIBE scheme that is adaptive secure with DKER, which has become the default security requirements for RIBE scheme. Since then, a lot of followup works of RIBE schemes with DKER have been proposed. Among them, the most recently scheme by Watanabe et al. [38] based on the modified Jutla-Roy IBE scheme [16] is the first adaptively secure RIBE scheme with DKER that can achieve short public parameters in prime order groups.

As the set of qualified users can change in each broadcast emission, efficient revocation of individual users or user groups is the primary objective of broadcast encryption. For the IBBE scheme, which is a natural generalization of BE in the identity-based setting, however, there is still no provably secure scalable revocation methodology has been proposed so far, even in Boldyreva et al.'s security model. Motivated by this, we further expand the study of revocable IBBE (RIBBE). We mainly focus on the construction of RIBBE scheme with DKER. In particular, we would like to have a construction that has constant size ciphertexts, which is more efficient and less bandwidth consuming compared with schemes of ciphertexts that are linear in the set of receivers.

Our Contribution In this paper, we propose a novel construction of revocable IBBE scheme with constant size ciphertexts. To prove its security with DKER, we first define the syntax of revocable IBBE scheme using KEM-DEM paradigm and its security model, which takes into account the realistic threat of decryption key exposure for the scenario of IBBE. To the best of our knowledge, this is the first construction of revocable IBBE with provable security. Specifically, our revocable IBBE scheme has the following merits.

1. Our scheme is a KEM which can produce a symmetric key along with a header, thus long messages can be encrypted under the short symmetric key. For simplicity, we only discuss the header size in the KEM, which is constant in our construction, regardless of the number of underlying receivers, which is very efficient both in the communication overheads and computational costs. Furthermore, only 4 group elements together with a tag are needed in the

ciphertexts header of our revocable IBBE scheme, which can be comparable to the revocable IBE scheme in [38]. Moreover, we implement it in `Charm` framework [3], more details of which can be deferred to Section 6.

2. The public parameters in our scheme is linear in the maximum size of the privileged identities set: m, which is predetermined and fixed in the setup phase. The private key for each user is linear in the value of $m*\log_2 N$, where the maximal value of system users N is also a predetermined value in the setup phase of the revocable IBBE system.
3. Our scheme also follows Boldyreval et al.'s revocation methodology [6] with the binary-tree data structure approach, which reduces the amount of work in key update from linear to logarithmic complexity in the maximal number of system users N. For each time period, the PKG will broadcast update key information through a public channel, which is useless for already revoked users. Only the non-revoked user can combine the update key and his private key to derive a decryption key that can be used to decrypt proper ciphertexts. More precisely, according to [33], the size of update key is $O(r\log_2(N/r))$ if $r \leqslant N/2$, or $O(N-r)$ if $r > N/2$, where r is the number of revoked users.
4. Our construction is built upon prime order bilinear groups of Type-3 pairings under mild variants of the Symmetric eXternal Diffie-Hellman (SXDH) assumption: the Augmented Decisional Diffie-Hellman on $\mathbb{G}_1$ (ADDH1) and Decisional Diffie-Hellman on $\mathbb{G}_2$ (DDH2). Note that ADDH1 assumption is first defined by Watanabe et al. in [38], which is proved in the generic bilinear group model.
5. With regard to the security, our revocable IBBE scheme is semi-adaptively secure with DKER under chosen plaintext attacks. Semi-adaptive security, first proposed by Chen and Wee [10], is a notion of security that lies between selective and adaptive security for functional encryption systems. More particularly, if we set the maximum size of receivers m to be $m = 1$, the resulting revocable IBBE scheme is a revocable IBE system, which can achieve adaptive security with DKER.

At a high level, our design approach is very similar to the Seo and Emura's technique of transforming IBE to RIBE in [33,38]. Firstly, there should be a basic IBE scheme that satisfies the requirement of (1) the secret key re-randomization property and (2) applicability of Boneh-Boyen technique [4]. Then, an adaptively secure RIBE scheme with DKER is constructed by applying the Seo-Emura technique. Similarly, we also employ a basic IBBE scheme and the Boneh-Boyen IBE scheme [4] as the building blocks. To achieve short ciphertexts and fast decryption, the basic IBBE is derived from the most recently proposed IBBE scheme of Ramanna [31], with necessary modifications mainly for the public parameters part to achieve the secret key re-randomization property. The security of the revocable IBBE scheme with DKER can be reduced to the adaptive security of the basic IBBE scheme. We note that it is not a trivial work to construct a revocable IBBE with adaptive security even given a revocable IBE scheme. The primary challenge in the security proof is how to simulate decryption keys for identities of the privileged recipients. While there is only one target identity

in the setting of RIBE, there will be multiple private keys and decryption keys that can be used to decrypt the challenge ciphertext. Note that the privileged recipients are chosen adaptively by the adversary, even the number of privileged recipients is unknown until the challenge phase, which makes it more complicated to simulate in the security proof. We partially overcome these issues by using the semi-adaptive security model, where the adversary should submit the privileged recipients just after receiving the public parameters. More technique will be needed to achieve adaptive security for revocable IBBE. As a side product, we also propose a new construction of revocable IBE scheme with adaptive security, which can be as a complementary of Watanabe et al.'s revocable IBE scheme [38]. In addition, because of using a different strategy, the adaptive security proof of the resulting RIBE scheme in this paper seems more succinct, compared with the security proof in the full version of [38].

Related Work. Hierarchical identity-based encryption (HIBE) is a simple extension of IBE which further supports a key delegation functionality. Revocable HIBE can support the revocation of user's private keys to manage the dynamic credentials of users in an HIBE system. Several improvement and variants with different properties have been proposed since the first revocable HIBE scheme with DKER introduced by Seo and Emura in [34]. Among them, the most popular revocable HIBE must be those given in [35,12,24], the security of which are proven in the selective model where an adversary should submits the challenge identity or the revocation list before he receives the public parameters. Revocable HIBE with DKER that is secure in the adaptive adversary model has been proposed in [36,20]. Unfortunately, these constructions are built upon composite order (product of three primes) bilinear groups, which is inefficient to implement compared with prime order groups implementation. We note that, contrary to HIBE, no organization of the users is needed in our revocable IBBE scheme to have constant size of ciphertexts, i.e., no hierarchy between identities in our revocable IBBE system.

Besides bilinear maps on elliptic curve, lattice is also a powerful tool to build cryptographic primitives. Lattice-based constructions, which are conjectured to be resistant to attacks by both classical and quantum computers, are currently important candidates for post-quantum cryptography. Chen et al. [8] proposed the first revocable IBE scheme (without DKER) in the lattice setting. Recently, Katsumata et al. [17] solved the open problem of achieving revocable (H)IBE with DKER in the lattice setting by proposing a new tool called the level conversion keys without relying on the key re-randomization property. In addition, revocable IBE scheme from codes with rank metric is proposed in [7], which is only proven selective security in the random oracle model.

We stress that the notion of revocation in this paper is referred to indirect revocation sometimes, since the key authority indirectly enables revocation by forcing revoked users to be unable to update their keys. A direct revocation mechanism has been studied for attribute-based encryption [2] and predicate encryption [27]. This approach requires the sender to carry out the revocation by specifying a set of revoked users in the ciphertext, and hence it does not need any

private key update procedures on the recipients's side. Recently, another notion of recipient-revocable identity-based broadcast encryption has been proposed in [32,22], which mainly focuses on how to remove some of the recipients from the set of receivers stated in the original ciphertext after the ciphertext has been generated, but without revealing the message content. Therefore, these systems [32,22] cannot follow the notion of revocable IBBE in this paper.

Server-aided revocable IBE, recently proposed by Qin et al. [30], is a novel system where most of the workloads on users are outsourced to an untrusted server. The server manages users' public key and key updates sent by the PKG periodically, and users can compute decryption keys without communicating with either the PKG or the server. Server-aided revocable IBE [29] and server-aide directly revocable predicate encryption [23] in the lattice setting have been proposed recently, which can satisfy selective security without DKER. It is possible to employ this construction methodology in our revocable IBBE scheme, which can obtain a server-aided revocable IBBE scheme with DKER.

Organization. The rest of the paper is organized as follows. In the next section, we review some preliminaries used throughout this paper, including the rigorous definitions and security model of revocable IBBE scheme. In Section 3, we present an adaptive secure IBBE scheme with short ciphertexts modified from Ramanna's original inner production encryption scheme [31], which is used as the core building block of our revocable IBBE scheme. In Section 4, we propose a concrete construction of revocable IBBE with DKER that can achieve constant size of ciphertext, together with proof of security in Section 5. To show its practicability we implement the proposed scheme in Section 6. Finally, Section 7 concludes this paper.

2 Preliminaries

2.1 Asymmetric Pairings and Hardness Assumptions

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be cyclic multiplicative groups of the same prime order p. Let g be a generator of $\mathbb{G}_1$ and h be a generator of $\mathbb{G}_2$. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ has the following properties:

- **Bilinearity**: For all $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$ and all $a, b \in \mathbb{Z}_p^*$, $e(g^a, h^b) = e(g, h)^{ab}$.
- **Non-degeneracy**: $e(g, h) \neq 1$.
- **Computability**: It is efficient to compute $e(u, v)$ for any $u \in \mathbb{G}_1$ and $v \in \mathbb{G}_2$.

It is called *symmetric* (or Type-1) pairing if $\mathbb{G}_1 = \mathbb{G}_2$; otherwise, the pairing is *asymmetric*. Two types of asymmetric pairing can be further classified: Type-2 and Type-3. If there is an efficiently computable isomorphism either from $\mathbb{G}_2$ to $\mathbb{G}_1$ or from $\mathbb{G}_1$ to $\mathbb{G}_2$, then the bilinear map e is called a Type-2 pairing. If no efficiently computable isomorphism is known, then we call it Type-3 pairing. Our constructions in this work are based on Type-3 pairing, which is the most efficient setting from an implementation point according to [14,9].

The security of our construction is based on the Augmented Decisional Diffie-Hellman on $\mathbb{G}_1$ (ADDH1), which is proved security in the generic bilinear group

model by Watanabe et al. [38], and Decisional Diffie-Hellman on $\mathbb{G}_2$ (DDH2) assumptions. Below, we describe these assumptions.

Let $\mathcal{G} = (p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ be a Type-3 pairing with generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$. Denote $D = (g_1, g_1^{\mu}, g_1^{\alpha_2}, g_1^{\beta\alpha}, g_2, g_2^{\alpha}, g_2^{\beta\alpha}, g_2^{\beta\alpha_2}, g_2^{1/\beta})$ with the following distribution: $\alpha, \alpha_2, \mu \xleftarrow{R} \mathbb{Z}_p, \beta, \eta \xleftarrow{R} \mathbb{Z}_p^*$. A PPT algorithm $\mathcal{A}$ given D and Z, whose task is to distinguish $Z = Z_0 = g_1^{\mu\alpha_2}$ (the case $\mathcal{A}$ will output 0) or $Z = Z_1 = g_1^{\mu\alpha_2+\eta}$ (the case $\mathcal{A}$ will output 1), has advantage $Adv_{\mathcal{G},\mathcal{A}}^{ADDH1}(\lambda)$ in solving the ADDH1 problem as:

$$Adv_{\mathcal{G},\mathcal{A}}^{ADDH1}(\lambda) = \left|\Pr\left[\mathcal{A}(\mathcal{G}, D, Z_0) = 1\right] - \Pr\left[\mathcal{A}(\mathcal{G}, D, Z_1) = 1\right]\right|.$$

Definition 1. *We say that the ADDH1 assumption holds if the advantage for all PPT adversaries $Adv_{\mathcal{G},\mathcal{A}}^{ADDH1}(\lambda)$ is negligible in the security parameter λ in solving the ADDH1 problem relative to a Type-3 pairing $\mathcal{G}$ of the group $\mathbb{G}_1$.*

Now we introduce the DDH2 assumption, which is defined as follows.

Definition 2. *We say that the DDH2 assumption holds for the group $\mathbb{G}_2$ of Type-3 pairing $\mathcal{G} = (p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ if the advantage $Adv_{\mathcal{G},\mathcal{A}}^{DDH2}(\lambda)$ which equals $\left|\Pr\left[\mathcal{A}(\mathcal{G}, D, g_2^{\mu\alpha}) = 1\right] - \Pr\left[\mathcal{A}(\mathcal{G}, D, g_2^{\mu\alpha+\eta}) = 1\right]\right|$ is negligible in λ for all PPT algorithms $\mathcal{A}$ with $D = (g_1, g_2, g_2^{\mu}, g_2^{\alpha})$ and the distribution: $\alpha, \mu \xleftarrow{R} \mathbb{Z}_p$, $\eta \xleftarrow{R} \mathbb{Z}_p^*$.*

Note that the dual of the above definition 2 with the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ reversed is Decisional Diffie-Hellman in $\mathbb{G}_1$ (DDH1) assumption. The Symmetric eXternal Diffie-Hellman (SXDH) assumption holds if both DDH1 and DDH2 problems are intracable. It can be easily verified that ADDH1 problem is not harder than DDH1, as an instance of DDH1 is embedded in the instance of ADDH1, and an algorithm to solve DDH1 can also be used to solve the ADDH1.

2.2 KUnodes Algorithm

To achieve scalable user revocation, we follow the node selection algorithm `KUNode` algorithm by using a binary tree data structure as in the previous RIBE schemes [6,33,21,38]. We employ similar notations as follows. For a binary tree `BT` with N leaves, we denote by `root` the root node of `BT`. For a non-leaf node θ, we write θ_L and θ_R as the left and right child of θ, respectively. For a leaf node η, we write $\texttt{Path}(\eta)$ as the set of nodes on the path from η to `root` (both η and `root` are inclusive). Each user is assigned to a leaf node η of `BT`. If a user who is associated with η is revoked on a time period t, then (η, t) is in the revocation list `RL`, i.e., $(\eta, t) \in$`RL`.

The `KUNode` algorithm which takes as input a binary tree `BT`, a revocation list `RL` as well as a time period t, is executed as follows. It first sets $X := Y := \emptyset$. For each $(\eta_i, t_i) \in RL$, if $t_i \leqslant t$ then it adds $\texttt{Path}(\eta_i)$ to X as: $X := X \cup \texttt{Path}(\eta_i)$. Then, for each $x \in X$, it will add x_L to Y for the case $x_L \notin X$, and it will add x_R to Y for the case $x_R \notin X$. Finally, it will output Y if $Y \neq \emptyset$. Otherwise, for the case $Y = \emptyset$, it will output $Y = \{root\}$.

Note that the output of `KUNode` algorithm Y is a minimal set of nodes in `BT` such that for any leaf node η listed in `RL`, it must hold that $\texttt{Path}(\eta) \cap Y = \emptyset$. But for the non-revoked leaf node η', there is exactly one node $\theta \in Y$ such that θ is an ancestor of η'. Two instances of the `KUNode` algorithm for the graphical description are illustrated below in Figure 1.

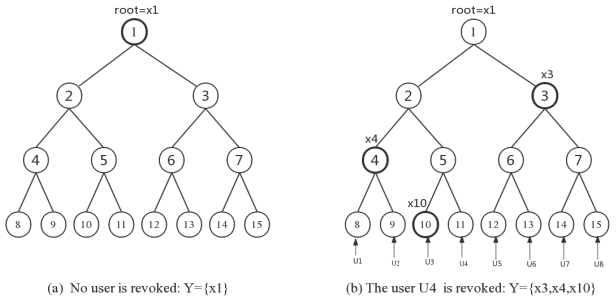

Fig. 1. Two instances of the `KUNode` algorithm

2.3 Syntax of Revocable IBBE Scheme

A revocable IBBE scheme $\mathcal{RIBBE}$ is described as follows: For simplicity, we omit the description of the security parameter λ and the public parameter PP in the input of all algorithms except for the **Setup** algorithm.

- **Setup**(λ, m, N): The setup algorithm takes as input the security parameter λ, the maximum size m of the set of privileged identities for one encryption together with the number of users N, and it returns the public parameters PP, the master secret key MSK, the initial revocation list $RL = \emptyset$ and a state ST. The algorithm also defines the identity space $\mathcal{ID}$, the time space $\mathcal{T}$ and the key space $\mathcal{K}$ for the DEM.
- **SKGen**(MSK, ID, ST): The secret key generation algorithm takes as input the master secret key MSK, an identity $ID \in \mathcal{ID}$, and the state information ST. It outputs a private key SK_{ID} associated with ID and updated ST.
- **KeyUp**(MSK, T, RL, ST): The key update generation algorithm takes as input the master secret key MSK, a key update time $T \in \mathcal{T}$, the revocation list RL and the state ST, and then outputs the update key KU_T.
- **DKGen**(SK_{ID}, KU_T): The decryption key generation algorithm takes a secret key SK_{ID} and key update KU_T as input, and outputs a decryption key $DK_{ID,T}$ or a symbol $\perp$ indicating that ID has been revoked by time T.
- **Encap**(T, S): The encapsulation algorithm takes as input the current time $T \in \mathcal{T}$ and a set of identities $S = \{ID_1, ..., ID_n\}$ with $n \leq m$, and it returns

a pair (Hdr, K), where Hdr is called the header and $K \in \mathcal{K}$ is the session key for the symmetric encryption scheme.
When a message $M \in \{0,1\}^*$ is broadcasted to receivers in S, the sender can compute the encryption C_M of M under the symmetric key $K \in \mathcal{K}$ of DEM and broadcasts (T, S, Hdr, C_M). We will refer to (T, S, Hdr) as the full header and C_M as the broadcast body.

- **Decap**$(T, S, Hdr, DK_{ID,T})$: This algorithm takes as input the full header (T, S, Hdr) with a set of identities $S = \{ID_1, ..., ID_n\}$ (satisfying that $n \leq m$), a decryption key $DK_{ID,T} \neq \perp$ corresponding an identity ID and time T. If $ID \in S$ the algorithm outputs the session key K which is then used to decrypt the broadcast body C_M to obtain the original message M.
- **Revoke**(ID, T, RL, ST): The stateful revocation algorithm takes an identity to be revoked $ID \in \mathcal{ID}$, a revocation time $T \in \mathcal{T}$, the current revocation list RL and the state ST as input, and outputs an updated revocation list RL.

Correctness. The correctness property requires that for all security parameter $\lambda \in \mathbb{N}$, all $(PP, MSK) \leftarrow Setup(\lambda, m, N)$, all possible state ST, a revocation list RL and for all sets $S \subseteq \mathcal{ID}$ with $|S| \leq m$, if $ID \in S$ is not revoked on the time $T \in \mathcal{T}$, then for $(SK_{ID}, ST) \leftarrow SKGen(MSK, ID, ST)$, $(KU_T, ST) \leftarrow KeyUp(MSK, T, RL, ST)$, $DK_{ID,T} \leftarrow DKGen(SK_{ID}, KU_T)$, $(Hdr, K) \leftarrow Encap(T, S)$, it should be satisfied that: $Decap(T, S, Hdr, DK_{ID,T}) = K$.

REMARK. Note that for $m = 1$, the above definition of revocable IBBE scheme is equal to a revocable IBE system, as is used in [33,38].

2.4 Security Models

The security model of RIBE was first introduced by Boldyreva et al. [6] and it was refined by Seo and Emura [33] by considering the realistic threat of decryption key exposure. We define IND-CPA security of a revocable IBBE system with decryption key exposure resistant, which is indistinguishable against chosen plaintext attacks for adaptive adversary. We basically refine the definition of [33], by adding extra restrictions for the scenario of broadcast encryption. We describe the security model using the following **IND-CPA** game between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Setup: The challenger $\mathcal{C}$ runs Setup(λ, m, N) algorithm of the revocable IBBE scheme $\mathcal{RIBBE}$ to get the public parameters PP, the master secret key MSK, a revocation list RL and a state ST. $\mathcal{C}$ keeps MSK, RL, ST to itself and gives PP to the adversary $\mathcal{A}$.

Key Extraction Phase 1: The adversary $\mathcal{A}$ can make a polynomial number of key extraction queries adaptively, which are processed as follows:

- If this is a private key query for an identity ID, then it gives the corresponding private key SK_{ID} to $\mathcal{A}$ by running SKGen(MSK, ID, ST) algorithm;
- If this is an update key query for the time T, then it gives the corresponding update key KU_T to $\mathcal{A}$ by running KeyUp(MSK, T, RL, ST) algorithm;
- If this is a decryption key query for ID and T, then it gives the corresponding decryption key $DK_{ID,T}$ to $\mathcal{A}$ by running DKGen(SK_{ID}, KU_T) algorithm;

– If this is a revocation key query for an identity to be revoked ID and a revocation time T, then it updates the revocation list RL by running Revoke(ID, T, RL, ST) algorithm with the following restriction: The revocation query cannot be queried at a time period T if the update key query for T was issued.

We note that the update key query and the revocation query can be queried at a time period which is later or equal to that of all previous queries, which means they are requested in non-decreasing order of time. In addition, the decryption key query cannot be queried at T before issuing T to the update key query.

Challenge: When $\mathcal{A}$ decides that phase 1 is over, a challenge time T^* and a challenge privileged set $S^* = \{ID_1, ID_2, ..., ID_n\}$ with $n \leq m$ are provided with the following constraints:

– If a private key query for an identity $ID \in S^*$ has been requested, then this identity ID must be revoked at some time $T \leq T^*$;
– There is no decryption key query for any ID in S^* and T^* has been requested.

$\mathcal{C}$ runs *Encap* algorithm to obtain $(Hdr, K_0) = \text{Encap}(S^*, T^*)$ and choose a random K_1 from the key space $\mathcal{K}$. $\mathcal{C}$ then picks a random coin $\beta \in \{0, 1\}$ and returns (Hdr, K_β) to $\mathcal{A}$.

Key Extraction Phase 2: The adversary $\mathcal{A}$ can continue to issue a polynomial number of additional key extraction queries as phase 1 with the same constraints, $\mathcal{C}$ will respond as before.

Guess: Eventually, the adversary $\mathcal{A}$ outputs a guess $\beta' \in \{0, 1\}$, and wins the game if $\beta' = \beta$.

Definition 3. *(Adaptive Security) Let* $Adv^{IND\text{-}CPA}_{\mathcal{A},RIBBE} = |\Pr(\beta' = \beta) - 1/2|$ *be the advantage for* $\mathcal{A}$ *in winning the* IND-CPA *game defined above. We say that a revocable IBBE scheme is adaptively secure under chosen plaintext attacks if for all polynomial-time adversary* $\mathcal{A}$*, the advantage in winning the above experiment* $Adv^{IND\text{-}CPA}_{\mathcal{A},RIBBE}$ *is negligible with respect to the security parameter* λ*.*

This security model above can capture realistic threat of decryption key exposure, as the adversary can make decryption key queries. This model reflects the scenario where all users get together and collude as in ordinary IBBE, since the adversary can get any user's private key except for S^*. Furthermore, even users in S^* can be corrupted, as long as they are revoked before the challenge time T^*. This is called adaptive security as the privileged set S^* is not chosen at the beginning. We can also define the selective security that is weaker than adaptive security similarly, except that the challenge S^* and T^* must be declared by the adversary before it sees the public parameters. In addition, we can define the semi-adaptive security that lies between selective and adaptive security.

Definition 4. *(Selective Security) The selective security of revocable IBBE under chosen plaintext attacks is similar to the adaptive security except that the adversary* $\mathcal{A}$ *should submit a challenge set* S^* *and challenge time* T^* *before it receives the public parameters. The advantage is defined as* $Adv^{sIND\text{-}CPA}_{\mathcal{A},RIBBE} =$

$|\Pr(\beta' = \beta) - 1/2|$. *We say that a revocable IBBE scheme is secure under chosen plaintext attacks in the selective model if for all polynomial-time adversary* $\mathcal{A}$*, the advantage in winning the above experiment* $Adv^{sIND\text{-}CPA}_{\mathcal{A},RIBBE}$ *is negligible with respect to the security parameter* λ.

Definition 5. *(Semi-adaptive Security) The semi-adaptive security of revocable IBBE under chosen plaintext attacks is similar to the adaptive security except that the adversary* $\mathcal{A}$ *should submit a challenge set* S^* *after it receives the public parameters but before it makes any key extracton query. The advantage is defined as* $Adv^{saIND\text{-}CPA}_{\mathcal{A},RIBBE} = |\Pr(\beta' = \beta) - 1/2|$. *We say that a revocable IBBE scheme is semi-adaptively secure under chosen plaintext attacks if for all polynomial-time adversary* $\mathcal{A}$*, the advantage in winning the above experiment* $Adv^{saIND\text{-}CPA}_{\mathcal{A},RIBBE}$ *is negligible with respect to the security parameter* λ.

3 The Basic IBBE Scheme

We now present our construction of identity-based broadcast encryption scheme with short ciphertexts. The core of our construction relies on realizing the inclusion relationship between one identity and a subset of identities from inner product. Note that the technique of deriving an IBBE scheme from the inner production encryption can be traced to the work of Katz et al. [18]. For each identity $ID \in \mathbb{Z}_p$, we can express it by setting a vector $\boldsymbol{x} = (x_0, x_1, \cdots, x_m)$, where $x_i = ID^i \bmod p$ for $i = 0, 1, ..., m$. For a subset $S = \{ID_1, ID_2, ..., ID_n\}$ with $n \leqslant m$, we can define a vector $\boldsymbol{y} = (y_0, y_1, \cdots, y_m)$, where $P_S[Z] = \prod_{ID_j \in S}(Z - ID_j) = \sum_{i=0}^{n} y_i Z^i$. If $n < m$, the coordinates $y_{n+1}, \cdots, y_m$ are all set to 0. It is easy to verify that $P_S[ID] = \sum_{i=0}^{m} y_i (ID)^i = \langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$ if and only if $ID \in S$.

3.1 Construction

As stated before, our basic IBBE scheme shares the same high level structure as the construction in [31]. In order to achieve the secret key re-randomization peoperty, each component of the master secret key needs to be available in the public parameters in some form of elements in source groups. We note that the extra public group elements, especially for the part of $(g_1^{\beta\alpha}, g_2^{\beta\alpha_1}, g_2^{\beta\alpha_2}, g_2^{1/\beta})$, will play an important role in the security proof of the subsequent revocable IBBE scheme. It is also worth mentioning that the security proof cannot be immediately applied, since some materials of the master secret key from [31] have been exposed in the public parameters. More precisely, our basic IBBE scheme $\prod_{\text{IBBE}}$ is constructed as follows.

- **Setup**(λ, m): Generate a Type-3 pairing $\mathcal{G} = (p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ with two random generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$. Le m be the maximum size of the legitimate set for one encryption, two random $(m+1)$-dimensional vectors are chosen from $\mathbb{Z}_p$ with $\boldsymbol{u}_1 = (u_{1,0}, u_{1,1}, \cdots, u_{1,m})$, $\boldsymbol{u}_2 = (u_{2,0}, u_{2,1}, \cdots, u_{2,m})$. Choose $\alpha_1, \alpha_2, w_1, w_2 \xleftarrow{R} \mathbb{Z}_p$, $b, \beta \xleftarrow{R} \mathbb{Z}_p^*$, set

$\boldsymbol{u} = \boldsymbol{u}_1 + b\boldsymbol{u}_2 = (u_0, u_1, \cdots, u_m)$, $w = w_1 + bw_2$, $\alpha = \alpha_1 + b\alpha_2$, and compute $\boldsymbol{U}_1 = g_1^{\boldsymbol{u}}$, $W_1 = g_1^w$, $g_T = e(g_1, g_2)^\alpha$. The master key is $MSK = (g_2^{\alpha_1}, g_2^{\alpha_2})$, and the public parameter PP is defined to be:
$PP = (g_1, g_1^b, \boldsymbol{U}_1, W_1, g_T, g_2, g_2^{\boldsymbol{u}_1}, g_2^{\boldsymbol{u}_2}, g_2^{w_1}, g_2^{w_2}, g_1^{\beta\alpha}, g_2^{\beta\alpha_1}, g_2^{\beta\alpha_2}, g_2^{1/\beta})$.

- **KeyGen**(PP, MSK, ID): For a user with an identity $ID \in \mathbb{Z}_p$, this algorithm chooses $r \xleftarrow{R} \mathbb{Z}_p$ and random tags $ktag_1, \cdots, ktag_m \xleftarrow{R} \mathbb{Z}_p$. The private key $SK_{ID} = \{K_1, K_2, K_3, (K_{4,i}, K_{5,i}, ktag_i)_{i=1}^m\}$ is defined to be:
$K_1 = g_2^{\alpha_1} \cdot (g_2^{w_1})^r, K_2 = g_2^{\alpha_2} \cdot (g_2^{w_2})^r, K_3 = g_2^r$. For $i = 1, 2, ..., m$:
$K_{4,i} = ((g_2^{w_1})^{ktag_i} \cdot g_2^{u_{1,i}} / (g_2^{u_{1,0}})^{(ID)^i})^r$, $K_{5,i} = ((g_2^{w_2})^{ktag_i} \cdot g_2^{u_{2,i}} / (g_2^{u_{2,0}})^{(ID)^i})^r$.
- **Encap**(PP, S): Assuming that the privileged set is $S = \{ID_1, ID_2, ..., ID_n\}$ with $n \leqslant m$ for notational simplicity. The algorithm defines a vector $\boldsymbol{y} = (y_0, y_1, \cdots, y_m)$ as the coefficient from $P_S[Z] = \prod_{ID_j \in S}(Z - ID_j) = \sum_{i=0}^n y_i Z^i$. It then picks randomly $s, ctag \in \mathbb{Z}_p$, and computes the session key $K = {g_T}^s$ which is used to encrypt the message, together with the header $Hdr = (C_1, C_2, C_3, ctag)$, where $C_1 = g_1^s, C_2 = (g_1^b)^s, C_3 = (W_1^{ctag} \cdot \prod_{i=0}^n (g_1^{u_i})^{y_i})^s$.
- **Decap**(PP, S, Hdr, SK_{ID}): The algorithm defines the vector $\boldsymbol{y} = (y_0, y_1, \cdots, y_m)$ according to the set S from the polynomial $P_S[Z]$ as above. It then computes $ktag = \sum_{i=1}^m y_i \cdot ktag_i$. If $ktag = ctag$, the output is $\perp$. Otherwise it computes:
$A = (e(C_1, \prod_{i=1}^m K_{4,i}^{y_i}) \cdot e(C_2, \prod_{i=1}^m K_{5,i}^{y_i}) / e(C_3, K_3))^{\frac{1}{ktag - ctag}}$, and returns the session key: $K = e(C_1, K_1) \cdot e(C_2, K_2) \cdot A^{-1}$.

CORRECTNESS. We observe that if $ID \in S$, we have $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=0}^m y_i (ID)^i = 0$, and $y_0 = -\sum_{i=1}^m y_i (ID)^i$. Then we have:

$$\prod_{i=1}^m K_{4,i}^{y_i} = ((g_2^{w_1})^{\sum_{i=1}^m y_i ktag_i} \cdot \prod_{i=1}^m (g_2^{u_{1,i}})^{y_i} / \prod_{i=1}^m (g_2^{u_{1,0}})^{\sum_{i=1}^m y_i (ID)^i})^r$$
$$= (g_2^{w_1 \sum_{i=1}^m y_i ktag_i} \cdot g_2^{\sum_{i=0}^m y_i u_{1,i}})^r = (g_2^{w_1 \cdot ktag} \cdot g_2^{\sum_{i=0}^m y_i u_{1,i}})^r;$$
$$\prod_{i=1}^m K_{5,i}^{y_i} = (g_2^{w_2 \cdot ktag} \cdot g_2^{\sum_{i=0}^m y_i u_{2,i}})^r;$$
$$e(C_1, \prod_{i=1}^m K_{4,i}^{y_i}) e(C_2, \prod_{i=1}^m K_{5,i}^{y_i}) = e(g_1^s, g_2^{r(w \cdot ktag + \sum_{i=0}^m y_i u_i)});$$
$$A = (e(C_1, \prod_{i=1}^m K_{4,i}^{y_i}) e(C_2, \prod_{i=1}^m K_{5,i}^{y_i}) \cdot e(C_3, K_3)^{-1})^{\frac{1}{ktag - ctag}} = e(g_1^s, g_2^{rw});$$
$$K = e(C_1, K_1) e(C_2, K_2) A^{-1} = e(g_1^s, g_2^{\alpha_1 + rw_1}) e((g_1^b)^s, g_2^{\alpha_2 + rw_2}) / e(g_1^s, g_2^{rw})$$
$$= e(g_1^s, g_2^{\alpha_1}) e((g_1^b)^s, g_2^{\alpha_2}) = e(g_1^s, g_2^{\alpha_1 + b\alpha_2}) = {g_T}^s.$$

3.2 Security Proof

We prove the security of the above basic IBBE scheme inspired from Ramanna's original inner production encryption scheme [31] following the theorem:

Theorem 1. *Suppose the ADDH1 and DDH2 assumptions hold in the Type-3 pairing $\mathcal{G} = (p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$, the basic IBBE scheme $\prod_{IBBE}$ in Section 3.1 is adaptively secure.*

Our security proof is obtained by applying the Waters' dual system methodology [37] via a hybrid argument over a sequence of games. Before we describe these games, we introduce the semi-functional headers and secret keys in terms of a transformation on a normal header or key. Note that these algorithms are provided for definitional purposes and only used in the security proof, but not in a real system. In particular, they do not need to be efficiently computable from the public parameters.

SFEncap$(PP, MSK, S, g_1^{w_1}, g_1^{\boldsymbol{u}_1})$: The algorithm first runs the **Encap** algorithm on a set $S = \{ID_1, ..., ID_n\}$ to generate a normal header-session key pair (Hdr', K') with $Hdr' = (C_1', C_2', C_3', ctag')$. Then it randomly chooses $\mu \in \mathbb{Z}_p$, and sets the semi-functional session key $K = K' \cdot e(g_1^{\mu}, g_2^{\alpha_1})$, together with $C_2 = C_2', ctag = ctag'$. It then sets $C_1 = C_1' \cdot g_1^{\mu}, C_3 = C_3' \cdot g_1^{\mu(\langle \boldsymbol{y}, \boldsymbol{u}_1 \rangle + ctag \cdot w_1)}$, where $\boldsymbol{y} = (y_0, y_1, \cdots, y_m)$ is the coefficient from $P_S[Z] = \prod_{ID_j \in S} (Z - ID_j) = \sum_{i=0}^{n} y_i Z^i$. The resulting header $Hdr = (C_1, C_2, C_3, ctag)$ is returned as the semi-functional header. Additionally, $g_1^{w_1}$ and $g_1^{\boldsymbol{u}_1}$ are needed to generate the semi-functional header in this algorithm.

SFKeyGen$(PP, MSK, ID, g_2^{1/b})$: The algorithm first runs the **KeyGen** algorithm to generate a normal private key $SK'_{ID} = \{K_1', K_2', K_3', (K_{4,i}', K_{5,i}', ktag_i')_{i=1}^m\}$. Then it chooses a random $\gamma \in \mathbb{Z}_p$, and sets $K_1 = K_1' \cdot g_2^{\gamma}, K_2 = K_2' / g_2^{\gamma/b}$, leaving the other elements $\{K_3, (K_{4,i}, K_{5,i})_{i=1}^m\}$ and the tags $\{(ktag_i)_{i=1}^m\}$ unchanged. The resulting key $SK_{ID} = \{K_1, K_2, K_3, (K_{4,i}, K_{5,i}, ktag_i)_{i=1}^m\}$ is returned as the semi-functional secret key. Note that $g_2^{1/b}$ is also needed in this algorithm.

We observe that if one applies the decapsulation procedure with a semi-functional key and a normal header, decapsulation will succeed as $e(C_1, g_2^{\gamma}) = e(C_2, g_2^{\gamma/b})$. That is, a normal header when decapsulated with a semi-functional user key returns the corresponding normal session key. Similarly, decapsulation of a semi-functional header by a normal key will also succeed because of:

$A' = e(g_1^{\mu}, \prod_{i=1}^{m} K_{4,i}^{y_i}) / e(g_1^{\mu(\langle \boldsymbol{y}, \boldsymbol{u}_1 \rangle + ctag \cdot w_1)}, K_3)^{\frac{1}{ktag - ctag}} = e(g_1^{\mu \cdot w_1}, g_2^r)$,
$e(g_1^{\mu}, K_1) / A' = e(g_1^{\mu}, g_2^{\alpha_1} \cdot (g_2^{w_1})^r) / e(g_1^{\mu \cdot w_1}, g_2^r) = e(g_1^{\mu}, g_2^{\alpha_1})$,

which equals the extra component of the semi-functional session key. However, when a semi-functional key is used to decapsulate a semi-functional header, the resulting session key will have an additional term of $e(g_1^{\mu}, g_2^{\gamma})$, which means decapsulation will fail when both the header and user's key are semi-functional.

We now present a sequence of games between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ defined as follows:

- $\text{Game}_{\text{Real}}$: The real IBBE security game, which is basically follows the adaptive security model of [15].
- Game_0: The same as $\text{Game}_{\text{Real}}$, except that the challenge header and session key are semi-functional.
- Game_k: The same as Game_0, except that the first k private keys are semi-functional for $1 \leqslant k \leqslant q$, where q is the number of key extraction queries made by the adversary $\mathcal{A}$.
- $\text{Game}_{\text{Final}}$: The same as Game_q, except that the challenge session key is a random element of $\mathbb{G}_T$.

Our proof will progress as follows, which can show that each game defined above is indistinguishable from the next under a complexity assumption. First, we transit from $\text{Game}_{\text{Real}}$ to Game_0, where the adversary $\mathcal{A}$'s advantage is bounded by the DDH1 assumption. Then we transit from Game_{k-1} to Game_k for each $1 \leqslant k \leqslant q$, and the adversary $\mathcal{A}$'s advantage is bounded by the DDH2 assumption. We note that in Game_q both the challenge header and all the private keys are semi-functional. At this point any private keys the challenger $\mathcal{C}$ gives out are not useful in decapsulating the header. Finally, we transit Game_q to $\text{Game}_{\text{Final}}$ under the ADDH1 assumption. It is easy to check that the header-session key pair given to the adversary $\mathcal{A}$ is independent with β in $\text{Game}_{\text{Final}}$, where the adversary has no advantage unconditionally.

We denote $Adv_{\mathcal{A},\text{IBBE}}^{\text{Game}_{\text{Real}}}$, $Adv_{\mathcal{A},\text{IBBE}}^{\text{Game}_0}$, $Adv_{\mathcal{A},\text{IBBE}}^{\text{Game}_k}$ $(1 \leqslant k \leqslant q)$ and $Adv_{\mathcal{A},\text{IBBE}}^{\text{Game}_{\text{Final}}}$ as the advantage in $\text{Game}_{\text{Real}}$, Game_0, Game_k and $\text{Game}_{\text{Final}}$, respectively. Our hybrid argument is accomplished in the following lemmas:

Lemma 1. *If there is an adversary $\mathcal{A}$ with $\left|Adv_{\mathcal{A},IBBE}^{Game_{Real}} - Adv_{\mathcal{A},IBBE}^{Game_0}\right| = \varepsilon$, we can build an algorithm $\mathcal{C}_0$ with advantage $Adv_{\mathcal{C}_0,\mathcal{G}}^{DDH1} = \varepsilon$ in breaking the DDH1 assumption for the Type-3 pairing $\mathcal{G}$.*

Lemma 2. *Suppose that there exists an adversary $\mathcal{A}$ that makes at most q queries with advantage $\left|Adv_{\mathcal{A},IBBE}^{Game_{k-1}} - Adv_{\mathcal{A},IBBE}^{Game_k}\right| = \varepsilon$ for some k where $1 \leqslant k \leqslant q$. Then we can build an algorithm $\mathcal{C}_k$ with advantage $Adv_{\mathcal{C}_k,\mathcal{G}}^{DDH2} = \varepsilon$ in breaking the DDH2 assumption for the Type-3 pairing $\mathcal{G}$.*

Lemma 3. *If there is an adversary $\mathcal{A}$ with $\left|Adv_{\mathcal{A},IBBE}^{Game_q} - Adv_{\mathcal{A},IBBE}^{Game_{Final}}\right| = \varepsilon$, we can build an algorithm $\mathcal{C}$ with advantage $Adv_{\mathcal{C},\mathcal{G}}^{ADDH1} = \varepsilon$ in breaking the ADDH1 assumption for the Type-3 pairing $\mathcal{G}$.*

The indistinguishability of $\text{Game}_{\text{Real}}$ and Game_0 as well as that of Game_{k-1} and Game_k for $1 \leqslant k \leqslant q$ can be proved similarly as the way in [31]. Due to space constraints, the proof for Lemma 1 and Lemma 2 is omitted here, but can be found in the full version. Here we only present the proof for Lemma 3 in Appendix A, which is the most non-trivial part in the theorem.

In addition, we note that the value of β is information theoretically hidden from the adversary $\mathcal{A}$ in $\text{Game}_{\text{Final}}$, the probability in which $\mathcal{A}$ wins is exactly $\frac{1}{2}$. Hence, $\mathcal{A}$ has no advantage in $\text{Game}_{\text{Final}}$: $\left|Adv_{\mathcal{A}\text{IBBE}}^{\text{Game}_{\text{Final}}} - \frac{1}{2}\right| = 0$. Thus, we have the advantage of $\mathcal{A}$ in breaking the security of our basic IBBE scheme $\prod_{\text{IBBE}}$:

$$
\begin{aligned}
Adv_{\mathcal{A},\text{IBBE}}^{\text{IND-CPA}} &= \left|Adv_{\mathcal{A},\text{IBBE}}^{\text{Game}_{\text{Real}}} - \tfrac{1}{2}\right| \leqslant \left|Adv_{\mathcal{A},\text{IBBE}}^{\text{Game}_{\text{Real}}} - Adv_{\mathcal{A},\text{IBBE}}^{\text{Game}_{\text{Final}}}\right| + \left|Adv_{\mathcal{A},\text{IBBE}}^{\text{Game}_{\text{Final}}} - \tfrac{1}{2}\right| \\
&\leqslant \left|Adv_{\mathcal{A}\text{IBBE}}^{\text{Game}_{\text{Real}}} - Adv_{\mathcal{A}\text{IBBE}}^{\text{Game}_0}\right| + \sum_{k=1}^{q}\left|Adv_{\mathcal{A}\text{IBBE}}^{\text{Game}_{k-1}} - Adv_{\mathcal{A},\text{IBBE}}^{\text{Game}_k}\right| + \left|Adv_{\mathcal{A}\text{IBBE}}^{\text{Game}_q} - Adv_{\mathcal{A}\text{IBBE}}^{\text{Game}_{\text{Final}}}\right| \\
&= Adv_{\mathcal{C},\mathcal{G}}^{\text{DDH1}} + q \cdot Adv_{\mathcal{C},\mathcal{G}}^{\text{DDH2}} + Adv_{\mathcal{C},\mathcal{G}}^{\text{ADDH1}}.
\end{aligned}
$$

Since the number of key extraction queries q is bounded by polynomial size, the advantage $Adv^{\text{IND-CPA}}_{\mathcal{A},\text{IBBE}}$ defined above is negligible under the DDH1, DDH2 and ADDH1 assumptions. This completes the proof of Theorem 1. □

4 Construction of Revocable IBBE Scheme

In this section, we present an efficient revocable IBBE scheme with constant size of headers, which is proven semi-adaptively secure in the standard model based on the IBBE scheme described in Section 3.1. As mentioned before, we basically follow the simple two-level HIBE (without delegating property) strategy in our construction. That is, the first level using the adaptively secure IBBE scheme, is assigned for identity, and the second level using the selectively secure Boneh-Boyen IBE [4], is assigned for the polynomial bounded time period. Our revocable IBBE scheme $\prod_{\text{RIBBE}}$ is described as follows:

- **Setup**(λ, m, N): Given the security parameter λ, $\mathcal{PKG}$ generates a Type-3 pairing $\mathcal{G} = (p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of prime order p. Also, two random generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are chosen as well as $e(g_1, g_2) \in \mathbb{G}_T$ is computed. As the maximum number of privileged identities is m, $\mathcal{PKG}$ then chooses from $\mathbb{Z}_p$ two random $(m+1)$-dimensional vectors $\boldsymbol{u}_1 = (u_{1,0}, u_{1,1}, \cdots, u_{1,m})$, $\boldsymbol{u}_2 = (u_{2,0}, u_{2,1}, \cdots, u_{2,m})$. Assuming that there are at most N users in the revocable IBBE systems, where N is a power of two for simplicity, a binary tree BT with N leaves is chosen. To generate the system public parameters PP, the authority $\mathcal{PKG}$ does the following:
  1. Choose randomly $b \xleftarrow{R} \mathbb{Z}_p^*$, $\alpha_1, \alpha_2, w_1, w_2, z_1, z_2, \hat{z}_1, \hat{z}_2 \xleftarrow{R} \mathbb{Z}_p$;
  2. Set $\boldsymbol{u} = \boldsymbol{u}_1 + b\boldsymbol{u}_2$, $w = w_1 + bw_2$, $\alpha = \alpha_1 + b\alpha_2$;
  3. Compute $\boldsymbol{U}_1 = g_1^{\boldsymbol{u}}, W_1 = g_1^{w}, g_T = e(g_1, g_2)^{\alpha}, Z_1 = g_1^{z_1 + b \cdot z_2}, \hat{Z}_1 = g_1^{\hat{z}_1 + b \cdot \hat{z}_2}$;
  4. Finally, output public parameters to be:
     $PP = (g_1, g_1^b, \boldsymbol{U}_1, W_1, Z_1, \hat{Z}_1, g_T, g_2, g_2^{\boldsymbol{u}_1}, g_2^{\boldsymbol{u}_2}, g_2^{w_1}, g_2^{w_2}, g_2^{z_1}, g_2^{z_2}, g_2^{\hat{z}_1}, g_2^{\hat{z}_2})$.

  The master key is defined $MSK = (g_2^{\alpha_1}, g_2^{\alpha_2})$, and the revocation list is $RL = \emptyset$.
- **SKGen**(PP, MSK, ID, ST): For a user associated with an identity $ID \in \mathbb{Z}_p$, $\mathcal{PKG}$ first chooses m random tags: $ktag_1, \cdots, ktag_m$ from $\mathbb{Z}_p$. It then picks an unassigned leaf node η randomly from BT and stores ID in this node η. For each node $\theta \in Path(BT, \eta)$, the authority does the following:
  1. Recall $H_\theta = (H_{1,\theta}, H_{2,\theta})$ from BT if it was defined. Otherwise, choose $H_{1,\theta}, H_{2,\theta} \xleftarrow{R} \mathbb{G}_2$ and store H_θ in the node θ;
  2. Choose r_θ randomly from $\mathbb{Z}_p$, and compute:
     $K_{1,\theta} = H_{1,\theta} \cdot (g_2^{w_1})^{r_\theta}$, $K_{2,\theta} = H_{2,\theta} \cdot (g_2^{w_2})^{r_\theta}$, $K_{3,\theta} = g_2^{r_\theta}$; For each $i = 1, 2, ..., m$:
     $K_{4,i,\theta} = g_2^{r_\theta (u_{1,i} - (ID)^i \cdot u_{1,0} + ktag_i \cdot w_1)}$, $K_{5,i,\theta} = g_2^{r_\theta (u_{2,i} - (ID)^i \cdot u_{2,0} + ktag_i \cdot w_2)}$.

  Return the private secret key SK_{ID} and an updated state ST with $SK_{ID} = \{K_{1,\theta}, K_{2,\theta}, K_{3,\theta}, (K_{4,i,\theta}, K_{5,i,\theta}, ktag_i)_{i=1}^m\}_{\theta \in Path(BT,\eta)}$.
- **KeyUp**(PP, MSK, T, RL, ST): $\mathcal{PKG}$ parses MSK as $(g_2^{\alpha_1}, g_2^{\alpha_2})$, and publishes key updates at time period T for each node $\theta \in KUNode(BT, RL, T)$ in the following steps:

1. Retrieve $H_\theta = (H_{1,\theta}, H_{2,\theta})$ from the state ST (As noted in [33], H_θ is always pre-defined in the **SKGen** algorithm).
2. Choose $s_\theta \xleftarrow{R} \mathbb{Z}_p$, and compute:
$KU_{1,\theta} = g_2^{\alpha_1} g_2^{s_\theta(z_1 + T\cdot\hat{z}_1)} H_{1,\theta}^{-1}$, $KU_{2,\theta} = g_2^{\alpha_2} g_2^{s_\theta(z_2+T\cdot\hat{z}_2)} H_{2,\theta}^{-1}$, $KU_{3,\theta} = g_2^{s_\theta}$.

Return the key update $KU_T = \{KU_{1,\theta}, KU_{2,\theta}, KU_{3,\theta}\}_{\theta \in KUNode(BT,RL,T)}$.

- **DKGen**(PP, SK_{ID}, KU_T): Parse $KU_T = \{KU_{1,\theta}, KU_{2,\theta}, KU_{3,\theta}\}_{\theta\in J}$ and $SK_{ID} = \{K_{1,\theta}, K_{2,\theta}, K_{3,\theta}, (K_{4,i,\theta}, K_{5,i,\theta}, ktag_i)_{i=1}^m\}_{\theta\in I}$ for some set of nodes I and J. The user will return $\perp$ if $I \cap J = \emptyset$. Otherwise, choose $\theta \in I \cap J$, $r'_\theta, s'_\theta \in \mathbb{Z}_p$, compute $DK_{ID,T} = \{DK_1, DK_2, DK_3, DK'_3, (DK_{4,i}, DK_{5,i}, ktag_i)_{i=1}^m\}$:
$DK_1 = K_{1,\theta}\cdot KU_{1,\theta}\cdot g_2^{r'_\theta\cdot w_1} g_2^{s'_\theta\cdot(z_1+T\cdot\hat{z}_1)}$, $DK_2 = K_{2,\theta}\cdot KU_{2,\theta}\cdot g_2^{r'_\theta w_2} g_2^{s'_\theta\cdot(z_2+T\cdot\hat{z}_2)}$,
$DK_3 = K_{3,\theta}\cdot g_2^{r'_\theta}$, $DK'_3 = KU_{3,\theta}\cdot g_2^{s'_\theta}$. For $i = 1, 2, ..., m$:
$DK_{4,i} = K_{4,i,\theta}\cdot g_2^{r'_\theta(u_{1,i} - (ID)^i\cdot u_{1,0} + ktag_i\cdot w_1)}$, $DK_{5,i} = K_{5,i,\theta}\cdot g_2^{r'_\theta\cdot(u_{2,i} - (ID)^i\cdot u_{2,0} + ktag_i\cdot w_2)}$.
- **Encap**(PP, T, S): To encrypt the privileged identity set $S = \{ID_1, ID_2, ..., ID_n\}$ with $n \leqslant m$, the algorithm defines a vector $\boldsymbol{y} = (y_0, y_1, \cdots, y_m)$ as the associated coefficient from: $P_S[Z] = \prod_{ID_j\in S}(Z - ID_j) = \sum_{i=0}^n y_i Z^i$. Note that the coordinates $y_{n+1}, \cdots, y_m$ are all set to 0 if $n < m$. Given the public parameters PP with $\boldsymbol{U}_1 = g_1^{\boldsymbol{u}} = (g_1^{u_0}, g_1^{u_1}, \cdots, g_1^{u_m})$, it picks $s, ctag \xleftarrow{R} \mathbb{Z}_p$, and computes the session key $K = g_T{}^s$ and the header $Hdr = (C_1, C_2, C_3, C_4, ctag)$ with $C_1 = g_1^s, C_2 = (g_1^b)^s, C_3 = (Z_1\cdot\hat{Z}_1^T)^s, C_4 = (W_1^{ctag}\cdot\prod_{i=0}^n (g_1^{u_i})^{y_i})^s$.
- **Decap**$(PP, T, S, Hdr, DK_{ID,T})$: For $ID \in S$, it parses $DK_{ID,T}$ and Hdr as $\{DK_1, DK_2, DK_3, DK'_3, (DK_{4,i}, DK_{5,i}, ktag_i)_{i=1}^m\}$ and $(C_1, C_2, C_3, C_4, ctag)$, respectively. This algorithm then computes $ktag = \sum_{i=1}^m y_i\cdot ktag_i$, where y_i is the coefficient of the polynomial $P_S[Z]$. If $ktag = ctag$, the output is $\perp$. Otherwise it computes:
$A = e(C_1, DK_1)\cdot e(C_2, DK_2)/e(C_3, DK'_3)$,
$B = (e(C_1, \prod_{i=1}^n (DK_{4,i})^{y_i})\cdot e(C_2, \prod_{i=1}^n (DK_{5,i})^{y_i})/e(C_4, DK_3))^{\frac{1}{ktag - ctag}}$,
and returns $K = A/B$ as the session key.
- **Revoke**(ID, T, RL, ST): This revocation algorithm updates the revocation list RL by adding (η, ID, T), where η is the leaf node associated with ID.

CORRECTNESS. The correctness of our revocable RIBBE scheme follows from the correctness analysis of IBBE in Section 3.1, and it is omitted here.

5 Security Analysis

Theorem 2. *If the ADDH1 assumption and DDH2 assumption hold, the proposed revocable IBBE scheme $\prod_{RIBBE}$ is semi-adaptively secure under chosen plaintext attacks. More particularly, if we set the maximum size of the set of receivers for one encryption m to be $m = 1$, the above revocable IBBE scheme $\prod_{RIBBE}$ is a revocable IBE system $\prod_{RIBE}$, which is adaptively secure against chosen plaintext attacks under the same assumptions.*

The proof of Theorem 2 proceeds in the following two Theorems: Theorem 3 and Theorem 4. We first provide a reduction in the semi-adaptive model to the

(non-revocable) basic IBBE scheme $\prod_{\text{IBBE}}$ described in Section 3.1, which has been proven to be adaptively secure in Section 3.2 under the ADDH1 and DDH2 assumptions. Therefore, the revocable IBBE scheme $\prod_{\text{RIBBE}}$ is semi-adaptivey secure under the ADDH1 and DDH2 assumptions.

Theorem 3. *If the underlying IBBE scheme $\prod_{IBBE}$ described in Section 3.1 is adaptively secure against chosen plaintext attacks, then the proposed revocable IBBE scheme $\prod_{RIBBE}$ in Section 4 is semi-adaptively secure.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that attacks the above revocable IBBE scheme $\prod_{\text{RIBBE}}$ with a non-negligible advantage ε, we will construct a PPT algorithm $\mathcal{C}$ to break the adaptive security of the basic IBBE scheme $\prod_{\text{IBBE}}$ described in Section 3.1, following the strategy-dividing lemma in [17]. Namely, when the challenge set $S^* = \{ID_1^*, ID_2^*, ..., ID_n^*\}$ is given by the (semi-adaptive) adversary $\mathcal{A}$, we can divide $\mathcal{A}$'s strategy into the following $n+1$ types: Type-0,Type-1,...,Type-n. $\mathcal{A}$ is a Type-k adversary if the number of identities for $ID \in S^*$ that $\mathcal{A}$ issues private key query is exactly k. In particular, $\mathcal{A}$ is a Type-0 adversary if $\mathcal{A}$ has not issued any $ID \in S^*$ for the private key query, and $\mathcal{A}$ is a Type-n adversary if $\mathcal{A}$ has queried private keys of all users in S^*. It is easy to check that Type-0, Type-1,..., Type-n can cover all possible strategies of $\mathcal{A}$, and each Type-k is mutually exclusive with publicly detectable in the security game. For each $k = 0, 1, ..., n$, let $\mathcal{A}_k$ denote an adversary that always follow the Type-k strategy (and never break the promise). We can prove that the advantage for each $\mathcal{A}_k$ in attacking the revocable IBBE scheme $\prod_{\text{RIBBE}}$ is negligible if the underlying IBBE scheme $\prod_{\text{IBBE}}$ described in Section 3.1 is adaptively secure. Thus, for general adversary $\mathcal{A}$ following an arbitrary strategy, $\prod_{\text{RIBBE}}$ satisfies semi-adaptive security, as the advantage $Adv^{\text{saIND-CPA}}_{\mathcal{A},RIBBE} \leqslant \sum_{k=0}^{n} Adv^{\text{saIND-CPA}}_{\mathcal{A}_k,RIBBE}$ is also negligible for polynomial size of n. In other words, we can divide the proof of Theorem 3 into $n+1$ sub-proofs of Lemma 4.

Lemma 4. *If the underlying IBBE scheme $\prod_{IBBE}$ described in Section 3.1 is adaptively secure against chosen plaintext attacks, then the advantage of a Type-k adversary $\mathcal{A}_k$ against the proposed revocable IBBE scheme $\prod_{RIBBE}$ in Section 4 is negligible.*

Proof of Lemma 4. At the beginning, $\mathcal{C}$ receives public parameters of $\prod_{\text{IBBE}}$: $(g_1, g_1^b, \boldsymbol{U}_1 = g_1^{\boldsymbol{u}_1 + b\boldsymbol{u}_2}, W_1 = g_1^{w_1 + bw_2}, g_T, g_2, g_2^{\boldsymbol{u}_1}, g_2^{\boldsymbol{u}_2}, g_2^{w_1}, g_2^{w_2}, g_1^{\beta\alpha}, g_2^{\beta\alpha_1}, g_2^{\beta\alpha_2}, g_2^{1/\beta})$. During the process, $\mathcal{C}$ can access to the secret key generation oracle $\text{KeyGen}_{\text{IBBE}}(\cdot)$, that is, if $\mathcal{C}$ sends this oracle $\text{KeyGen}_{\text{IBBE}}(\cdot)$ an identity ID, then it will receive a private key of $SK_{ID} = \{K_1, K_2, K_3, (K_{4,i}, K_{5,i}, ktag_i)_{i=1}^m\}$ with:
$K_1 = g_2^{\alpha_1} \cdot (g_2^{w_1})^r, K_2 = g_2^{\alpha_2} \cdot (g_2^{w_2})^r, K_3 = g_2^r,$
$K_{4,i} = ((g_2^{w_1})^{ktag_i} \cdot g_2^{u_{1,i}} / (g_2^{u_{1,0}})^{(ID)^i})^r$, $K_{5,i} = ((g_2^{w_2})^{ktag_i} \cdot g_2^{u_{2,i}} / (g_2^{u_{2,0}})^{(ID)^i})^r$.

Then $\mathcal{C}$ interacts with $\mathcal{A}_k$ as follows:

Setup: $\mathcal{C}$ should guess the right time period T^* that $\mathcal{A}_k$ will submit the target identity in the challenge ciphertext phase. For the rest of the proof, assuming that $\mathcal{C}$'s guess is correct, which holds with probability $1/|\mathcal{T}|$. Note that $\mathcal{C}$ will

terminate the simulation once $\mathcal{C}$ finds that the guess is wrong, and a random bit β' will be outputted. $\mathcal{C}$ then proceeds as follows:

1. It first creates a binary tree BT with N leaves. It initializes RL and ST as an empty set respectively.
2. $\mathcal{C}$ chooses $z_1', z_2', \hat{z}_1', \hat{z}_2' \xleftarrow{R} \mathbb{Z}_p$ and computes:
   $Z_1 = g_1^{z_1+b\cdot z_2} = g_1^{z_1'}(g_1^b)^{z_2'}/(g_1^{\beta\alpha})^{T^*}, \hat{Z}_1 = g_1^{\hat{z}_1+b\cdot\hat{z}_2} = g_1^{\hat{z}_1'}(g_1^b)^{\hat{z}_2'}(g_1^{\beta\alpha})$,
   $g_2^{z_1} = g_2^{z_1'}/(g_2^{\beta\alpha_1})^{T^*}, g_2^{z_2} = g_2^{z_2'}/(g_2^{\beta\alpha_2})^{T^*}, g_2^{\hat{z}_1} = g_2^{\hat{z}_1'}g_2^{\beta\alpha_1}, g_2^{\hat{z}_2} = g_2^{\hat{z}_2'}g_2^{\beta\alpha_2}$,
   which implicitly sets:
   $z_1 = z_1' - T^*\cdot\beta\alpha_1, z_2 = z_2' - T^*\cdot\beta\alpha_2, \hat{z}_1 = \hat{z}_1' + \beta\alpha_1, \hat{z}_2 = \hat{z}_2' + \beta\alpha_2$.
3. $\mathcal{C}$ then sends to $\mathcal{A}_k$ the public parameters of $\prod_{\text{RIBBE}}$ as:
   $PP = (g_1, g_1^b, \boldsymbol{U}_1, W_1, Z_1, \hat{Z}_1, g_T, g_2, g_2^{\boldsymbol{u}_1}, g_2^{\boldsymbol{u}_2}, g_2^{w_1}, g_2^{w_2}, g_2^{z_1}, g_2^{z_2}, g_2^{\hat{z}_1}, g_2^{\hat{z}_2})$.

Challenge Set: $\mathcal{A}_k$ submits a challenge set $S^* = \{ID_1^*, ID_2^*, ..., ID_n^*\}$ to the challenger $\mathcal{C}$, which will be used in the challenge ciphertexts. As the Type-k adversary $\mathcal{A}_k$ only issues k private key query for any identity $ID^* \in S^*$, $\mathcal{C}$ will choose k random leaf nodes $L_k^* = \{\eta_1^*, \eta_2^*, ..., \eta_k^*\}$ from BT. η_i^* will be assigned to an identity $ID^* \in S^*$ that is issued by $\mathcal{A}_k$ for the private key. We emphasize that η_i^* is not necessary assigned to $ID_i^* \in S^*$.

Key Extraction Phase 1: $\mathcal{A}_k$ may adaptively make a polynomial number of queries, which are processed as follows:

– If this is a private key query for an identity ID, $\mathcal{C}$ performs the following:
  1. It first checks whether $ID \in S^*$ or not. If $ID \in S^*$, $\mathcal{C}$ will assign ID to a random undefined leaf $\eta^* \in L_k^*$ and saves (ID, η^*) to ST. Furthermore, $\mathcal{C}$ chooses m random tags: $ktag_1, \cdots, ktag_m$ from $\mathbb{Z}_p$, and stores these tags for ID. Otherwise, $ID \notin S^*$, $\mathcal{C}$ assigns ID to a random undefined leaf η outside of L_k^* from BT and saves (ID, η) to ST. $\mathcal{C}$ can transfer ID to the oracle: $\text{KeyGen}_{\text{IBBE}}(\cdot)$ and gets the private key $\{K_1, K_2, K_3, (K_{4,i}, K_{5,i} ktag_i)_{i=1}^m\}$. $\mathcal{C}$ also stores these given tags for ID. We denote N_k^* as all the nodes from the root node to the leaf nodes which are assigned to identities queried in the challenge set S^*: $N_k^* = \bigcup_{i=1}^k Path(BT, \eta_i^*)$.
  2. For each node $\theta \in Path(BT, \eta)$, $\mathcal{C}$ can retrieve H_θ if it was defined. Otherwise, it chooses $H_\theta = (H_{1,\theta}, H_{2,\theta}) \xleftarrow{R} \mathbb{G}_2$ and stores H_θ in the node θ. Note that θ can be further divided into the following two types according to N_k^*:
     - **Case** $\theta \notin N_k^*$: In this case, $ID \notin S^*$ and $\mathcal{C}$ has gotten a private key of ID from the oracle $\text{KeyGen}_{\text{IBBE}}(\cdot)$: $\{K_1, K_2, K_3, (K_{4,i}, K_{5,i} ktag_i)_{i=1}^m\}$. $\mathcal{C}$ further chooses $r_\theta \xleftarrow{R} \mathbb{Z}_p$, and computes:
       $K_{1,\theta} = K_1\cdot(g_2^{w_1})^{r_\theta}\cdot H_{1,\theta}, K_{2,\theta} = K_2\cdot(g_2^{w_2})^{r_\theta}\cdot H_{2,\theta}, K_{3,\theta} = K_3\cdot g_2^{r_\theta}$,
       $K_{4,i,\theta} = K_{4,i}\cdot g_2^{r_\theta(u_{1,i}-(ID)^i\cdot u_{1,0}+ktag_i\cdot w_1)}$,
       $K_{5,i,\theta} = K_{5,i}\cdot g_2^{r_\theta(u_{2,i}-(ID)^i\cdot u_{2,0}+ktag_i\cdot w_2)}$.
       REMARK. Here we implicitly set $H_{1,\theta} := g_2^{\alpha_1}H_{1,\theta}, H_{2,\theta} := g_2^{\alpha_2}H_{2,\theta}$, $r_\theta := r + r_\theta$, where r denotes the internal randomness of K_3.

- **Case** $\theta \in N_k^*$: $\mathcal{C}$ can retrieve the tags $ktag_1, \cdots, ktag_m$ corresponding to ID. Then it chooses $r_\theta \xleftarrow{R} \mathbb{Z}_p$, and computes:
  $K_{1,\theta} = H_{1,\theta} \cdot (g_2^{w_1})^{r_\theta}, K_{2,\theta} = H_{2,\theta} \cdot (g_2^{w_2})^{r_\theta}, K_{3,\theta} = g_2^{r_\theta}$;
  $K_{4,i,\theta} = g_2^{r_\theta(u_{1,i}-(ID)^i \cdot u_{1,0}+ktag_i \cdot w_1)}, K_{5,i,\theta} = g_2^{r_\theta(u_{2,i}-(ID)^i \cdot u_{2,0}+ktag_i \cdot w_2)}$.

3. Finally, it stores and outputs the private key SK_{ID} to $\mathcal{A}_k$ with:
   $SK_{ID} = \{K_{1,\theta}, K_{2,\theta}, K_{3,\theta}, (K_{4,i,\theta}, K_{5,i,\theta}, ktag_i)_{i=1}^m\}_{\theta \in Path(BT,\eta)}$.

– If this is an update key query for the time T, $\mathcal{C}$ first runs $\text{KUNode}(BT, RL, T)$ algorithm with the current revocation list RL and time T. For each node $\theta \in \text{KUNode}(BT, RL, T)$, $\mathcal{C}$ can retrieve H_θ if it was defined. Otherwise, it chooses $H_\theta = (H_{1,\theta}, H_{2,\theta}) \xleftarrow{R} \mathbb{G}_2$ and stores H_θ in the node θ. Then it chooses $s_\theta \xleftarrow{R} \mathbb{Z}_p$, checks whether $\theta \in N_k^*$, and computes:

- **Case** $\theta \in N_k^*$: $KU_{1,\theta} = (g_2^{z_1}(g_2^{\hat{z}_1})^T)^{s_\theta} H_{1,\theta}^{-1}(g_2^{\frac{1}{\beta}})^{-\frac{z_1'+T\hat{z}_1'}{T-T^*}}$,
  $KU_{2,\theta} = (g_2^{z_2}(g_2^{\hat{z}_2})^T)^{s_\theta} H_{2,\theta}^{-1}(g_2^{\frac{1}{\beta}})^{-\frac{z_2'+T\hat{z}_2'}{T-T^*}}$, $KU_{3,\theta} = g_2^{s_\theta}(g_2^{\frac{1}{\beta}})^{-\frac{1}{T-T^*}}$.
  REMARK. In this case, we implicitly set $s_\theta := s_\theta - \frac{1}{\beta(T-T^*)}$. We further emphasize that for $T = T^*$, there will be no node θ such that $\theta \in \text{KUNode}(BT, RL, T) \cap N_k^*$, as the corresponding SK_{ID} with $ID \in S^*$ must be revoked before T^* according to the restriction.
- **Case** $\theta \notin N_k^*$:
  $KU_{1,\theta} = (g_2^{z_1}(g_2^{\hat{z}_1})^T)^{s_\theta} H_{1,\theta}^{-1}, KU_{2,\theta} = (g_2^{z_2}(g_2^{\hat{z}_2})^T)^{s_\theta} H_{2,\theta}^{-1}, KU_{3,\theta} = g_2^{s_\theta}$.

Finally, $\mathcal{C}$ stores and outputs the update key KU_T to $\mathcal{A}_k$ with:
$KU_T = \{KU_{1,\theta}, KU_{2,\theta}, KU_{3,\theta}\}_{\theta \in \text{KUNode}(BT,RL,T)}$.

– For a decryption key query with an identity ID and time T, if ID is not revoked before T (otherwise, $\mathcal{C}$ can output $\perp$), $\mathcal{C}$ can generate the corresponding decryption key $DK_{ID,T}$ in the following way, regardless of the strategy taken by $\mathcal{A}$:

- **Case** $ID \in S^*$: $\mathcal{C}$ selects random exponents $s, r, ktag_1, \cdots, ktag_m \xleftarrow{R} \mathbb{Z}_p$ and creates the decryption $DK_{ID,T}$ as:
  $DK_1 = (g_2^{z_1}(g_2^{\hat{z}_1})^T)^s (g_2^{w_1})^r (g_2^{\frac{1}{\beta}})^{-\frac{z_1'+T\hat{z}_1'}{T-T^*}}$,
  $DK_2 = (g_2^{z_2}(g_2^{\hat{z}_2})^T)^s (g_2^{w_2})^r (g_2^{\frac{1}{\beta}})^{-\frac{z_2'+T\hat{z}_2'}{T-T^*}}$, $DK_3 = g_2^r, DK_3' = g_2^s$,
  $DK_{4,i} = g_2^{r(u_{1,i}-(ID)^i \cdot u_{1,0}+ktag_i \cdot w_1)}, DK_{5,i} = g_2^{r(u_{2,i}-(ID)^i \cdot u_{2,0}+ktag_i \cdot w_2)}$.
- **Case** $ID \notin S^*$: In this case, $\mathcal{C}$ transfers ID to the oracle: $\text{KeyGen}_{\text{IBBE}}(\cdot)$, and gets the private key $\{K_1, K_2, K_3, (K_{4,i}, K_{5,i}, ktag_i)_{i=1}^m\}$. $\mathcal{C}$ further chooses $r, s \xleftarrow{R} \mathbb{Z}_p$, and computes:
  $DK_1 = K_1(g_2^{z_1}(g_2^{\hat{z}_1})^T)^s (g_2^{w_1})^r, DK_2 = K_2(g_2^{z_2}(g_2^{\hat{z}_2})^T)^s (g_2^{w_2})^r$,
  $DK_3 = K_3 g_2^r$, $DK_3' = g_2^s$;
  $DK_{4,i} = K_{4,i} g_2^{r(u_{1,i}-(ID)^i \cdot u_{1,0}+ktag_i \cdot w_1)}$,
  $DK_{5,i} = K_{5,i} g_2^{r(u_{2,i}-(ID)^i \cdot u_{2,0}+ktag_i \cdot w_2)}$.

REMARK. Note that T will never equal to T^* in the case of $ID \in S^*$ according to the restriction in the security model of Section 2.4.
Finally, $\mathcal{C}$ stores and outputs the decryption key $DK_{ID,T}$ to $\mathcal{A}$ with:
$DK_{ID,T} = \{DK_1, DK_2, DK_3, DK_3', (DK_{4,i}, DK_{5,i}, ktag_i)_{i=1}^m\}$.

– If this is a revocation key query for an identity to be revoked ID and a revocation time T, then $\mathcal{C}$ updates the revocation list RL by running Revoke(ID, T, RL, ST) algorithm.

Challenge Cihphertexts: Once $\mathcal{A}_k$ decides that the **Key Extraction Phase 1** is over, $\mathcal{C}$ sends the challenge privileged set $S^* = \{ID_1^*, ID_2^*, ..., ID_n^*\}$ to the challenger in the IND-CPA game of the IBBE scheme $\prod_{\text{IBBE}}$ and gets (Hdr^*, K^*) with $Hdr^* = (C_1^*, C_2^*, C_3^*, ctag^*)$. Note that (Hdr^*, K_0^*) is obtained from the challenger of $\prod_{\text{IBBE}}$ by running *Encap* algorithm with $(Hdr^*, K_0^*) =$ Encap(S^*), and K_1^* is a random element from the key space $\mathcal{K}$ of $\prod_{\text{IBBE}}$. It is $\mathcal{C}$'s task to decide $K^* = K_0^*$ or $K^* = K_1^*$. $\mathcal{C}$ sets $C_1 = C_1^*, C_2 = C_2^*, C_3 = (C_1^*)^{z_1' + T^* \hat{z}_1'} \cdot (C_2^*)^{z_2' + T^* \hat{z}_2'}, C_4 = C_3^*, ctag = ctag^*$ and sends $(Hdr = (C_1, C_2, C_3, C_4, ctag), K^*)$ to $\mathcal{A}$ as the challenge header and session key pair.

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

Guess: Finally, $\mathcal{A}$ outputs a guess $\beta' \in \{0, 1\}$, and $\mathcal{C}$ will transfer it to the challenger in the IND-CPA game of the IBBE scheme $\prod_{\text{IBBE}}$.

Now we show that the simulation is correct. That is, the distribution of all the above transcriptions between $\mathcal{A}_k$ and $\mathcal{C}$ is identical to the real experiment from the viewpoint of $\mathcal{A}_k$. Firstly, the public parameters PP is correct as the exponents $z_1', z_2', \hat{z}_1', \hat{z}_2' \in \mathbb{Z}_p$ are randomly chosen. Secondly, we show that the private keys are correct. For each node $\theta \in Path(BT, \eta)$, it can be easily verified that the private keys are of the same distribution in the case of $\theta \in N_k^*$. In the case of $\theta \notin N_k^*$, the private key for $(K_{1,\theta}, K_{2,\theta})$ is also correctly distributed from the setting $H'_{1,\theta} = g_2^{\alpha_1} \cdot H_{1,\theta}, H'_{2,\theta} = g_2^{\alpha_2} \cdot H_{2,\theta}, r'_\theta = r + r_\theta$ as

$K_{1,\theta} = K_1 \cdot (g_2^{w_1})^{r_\theta} \cdot H_{1,\theta} = (g_2^{\alpha_1} \cdot H_{1,\theta}) \cdot (g_2^{w_1})^{r + r_\theta} = H'_{1,\theta} \cdot (g_2^{w_1})^{r'_\theta}$,

$K_{2,\theta} = K_2 \cdot (g_2^{w_2})^{r_\theta} \cdot H_{2,\theta} = (g_2^{\alpha_1} \cdot H_{1,\theta}) \cdot (g_2^{w_2})^{r + r_\theta} = H'_{2,\theta} \cdot (g_2^{w_2})^{r'_\theta}$.

Thirdly, we show that the update key is correct. In case of $\theta \in N_k^*$, we have that a time related update key is correctly distributed from the setting $s'_\theta = s_\theta - \frac{1}{\beta(T - T^*)}$ as it holds that:

$$KU_{1,\theta} = (g_2^{z_1}(g_2^{\hat{z}_1})^T)^{s_\theta} H_{1,\theta}^{-1} (g_2^{\frac{1}{\beta}})^{-\frac{z_1' + T\hat{z}_1'}{T - T^*}} = (g_2^{z_1' - T^*\beta\alpha_1} (g_2^{\hat{z}_1'} g_2^{\beta\alpha_1})^T)^{s_\theta} g_2^{-\frac{z_1' + T\hat{z}_1'}{\beta(T - T^*)}} H_{1,\theta}^{-1}$$

$$= g_2^{\alpha_1} (g_2^{z_1' + T\hat{z}_1'} g_2^{(T - T^*)\beta\alpha_1})^{s_\theta} g_2^{-\frac{z_1' + T\hat{z}_1'}{\beta(T - T^*)}} \cdot g_2^{-\alpha_1} \cdot H_{1,\theta}^{-1}$$

$$= g_2^{\alpha_1} (g_2^{z_1' + T\hat{z}_1'} g_2^{(T - T^*)\beta\alpha_1})^{s_\theta} \cdot (g_2^{\alpha_1 \beta(T - T^*) + z_1' + T\hat{z}_1'})^{-\frac{1}{\beta(T - T^*)}} \cdot H_{1,\theta}^{-1}$$

$$= g_2^{\alpha_1} (g_2^{z_1' + T\hat{z}_1'} g_2^{(T - T^*)\beta\alpha_1})^{s_\theta - \frac{1}{\beta(T - T^*)}} \cdot H_{1,\theta}^{-1}$$

$$= g_2^{\alpha_1} (g_2^{z_1' + T\hat{z}_1'} g_2^{(T - T^*)\beta\alpha_1})^{s'_\theta} \cdot H_{1,\theta}^{-1}$$

$$= g_2^{\alpha_1} (g_2^{z_1} (g_2^{\hat{z}_1})^T)^{s'_\theta} \cdot H_{1,\theta}^{-1},$$

$$KU_{2,\theta} = (g_2^{z_2}(g_2^{\hat{z}_2})^T)^{s_\theta} H_{2,\theta}^{-1} (g_2^{\frac{1}{\beta}})^{-\frac{z_2' + T\hat{z}_2'}{T - T^*}} = (g_2^{z_2' - T^*\beta\alpha_2} (g_2^{\hat{z}_2'} g_2^{\beta\alpha_2})^T)^{s_\theta} g_2^{-\frac{z_2' + T\hat{z}_2'}{\beta(T - T^*)}} H_{2,\theta}^{-1}$$

$$= g_2^{\alpha_2} (g_2^{z_2' + T\hat{z}_2'} g_2^{(T - T^*)\beta\alpha_2})^{s_\theta} g_2^{-\frac{z_2' + T\hat{z}_2'}{\beta(T - T^*)}} \cdot g_2^{-\alpha_2} \cdot H_{2,\theta}^{-1}$$

$= g_2^{\alpha_2}(g_2^{z'_2+T\hat{z}'_2} g_2^{(T-T^*)\beta\alpha_2})^{s'_\theta} \cdot H_{2,\theta}^{-1} = g_2^{\alpha_2}(g_2^{z_2}(g_2^{\hat{z}_2})^T)^{s'_\theta} \cdot H_{2,\theta}^{-1}$,

$KU_{3,\theta} = g_2^{s_\theta}(g_2^{\frac{1}{\beta}})^{-\frac{1}{T-T^*}} = g_2^{s_\theta - \frac{1}{\beta(T-T^*)}} = g_2^{s'_\theta}$.

In case of $\theta \notin N_k^*$, the update key is correctly distributed from the setting $H'_{1,\theta} = g_2^{\alpha_1} H_{1,\theta}, H'_{2,\theta} = g_2^{\alpha_2} H_{2,\theta}$, $s'_\theta = s_\theta$ as:

$KU_{1,\theta} = (g_2^{z_1}(g_2^{\hat{z}_1})^T)^{s_\theta} H_{1,\theta}^{-1} = g_2^{\alpha_1}(g_2^{z_1}(g_2^{\hat{z}_1})^T)^{s_\theta} g_2^{-\alpha_1} H_{1,\theta}^{-1} = g_2^{\alpha_1}(g_2^{z_1}(g_2^{\hat{z}_1})^T)^{s_\theta}(H'_{1,\theta})^{-1}$,

$KU_{2,\theta} = (g_2^{z_2}(g_2^{\hat{z}_2})^T)^{s_\theta} H_{2,\theta}^{-1} = g_2^{\alpha_2}(g_2^{z_2}(g_2^{\hat{z}_2})^T)^{s_\theta}(H'_{2,\theta})^{-1}$, $KU_{3,\theta} = g_2^{s_\theta} = g_2^{s'_\theta}$.

Fourthly, we show that the decryption key is correct. As we have proved before, both the private key SK_{ID} and the update key KU_T are correctly distributed, the resulting decryption key must be correctly distributed by running the $\mathrm{DKGen}(SK_{ID}, KU_T)$ algorithm. Furthermore, if ID is not revoked before T, we can prove that the decryption key for $(DK_{1,\theta}, DK_{2,\theta})$ is also correctly distributed directly with $SK_{ID} = \{K_{1,\theta}, K_{2,\theta}, K_{3,\theta}, (K_{4,i,\theta}, K_{5,i,\theta}, ktag_{i,\theta})_{i=1}^m\}_{\theta \in Path(BT,\eta)}$ and $KU_T = \{KU_{1,\theta}, KU_{2,\theta}, KU_{3,\theta}\}_{\theta \in \mathrm{KUNode}(BT,RL,T)}$.

In the case of $ID \in S^*$, for the time $T \neq T^*$, there must exist a node $\theta \in \mathrm{KUNode}(BT, RL, T) \cap N_n^*$ with $KU_{1,\theta} = (g_2^{z_1}(g_2^{\hat{z}_1})^T)^{s_\theta} H_{1,\theta}^{-1}(g_2^{\frac{1}{\beta}})^{-\frac{z'_1+T\hat{z}'_1}{T-T^*}} = g_2^{\alpha_1}(g_2^{z_1}(g_2^{\hat{z}_1})^T)^{s'_\theta} \cdot H_{1,\theta}^{-1}$ and $K_{1,\theta} = H_{1,\theta} \cdot (g_2^{w_1})^{r_\theta}$. According to the $\mathrm{DKGen}(SK_{ID}, KU_T)$ algorithm:

$K_{1,\theta} \cdot KU_{1,\theta} = H_{1,\theta} \cdot (g_2^{w_1})^{r_\theta} \cdot g_2^{\alpha_1}(g_2^{z_1}(g_2^{\hat{z}_1})^T)^{s'_\theta} \cdot H_{1,\theta}^{-1} = g_2^{\alpha_1} \cdot (g_2^{w_1})^{r_\theta} \cdot (g_2^{z_1}(g_2^{\hat{z}_1})^T)^{s'_\theta}$,

$K_{2,\theta} \cdot KU_{2,\theta} = H_{2,\theta} \cdot (g_2^{w_2})^{r_\theta} \cdot g_2^{\alpha_2}(g_2^{z_2}(g_2^{\hat{z}_2})^T)^{s'_\theta} \cdot H_{2,\theta}^{-1} = g_2^{\alpha_2} \cdot (g_2^{w_2})^{r_\theta} \cdot (g_2^{z_2}(g_2^{\hat{z}_2})^T)^{s'_\theta}$.

Thus, $DK_1 = K_{1,\theta} \cdot KU_{1,\theta} \cdot g_2^{r'_\theta \cdot w_1} g_2^{s'_\theta \cdot (z_1 + T \cdot \hat{z}_1)}$ and $DK_2 = K_{2,\theta} \cdot KU_{2,\theta} \cdot g_2^{r'_\theta \cdot w_2} g_2^{s'_\theta \cdot (z_2 + T \cdot \hat{z}_2)}$ have the correct distribution.

In the case of $ID \notin S^*$, as $K_{1,\theta} = K_1 \cdot (g_2^{w_1})^{r_\theta} \cdot H_{1,\theta}, K_{2,\theta} = K_2 \cdot (g_2^{w_2})^{r_\theta} \cdot H_{2,\theta}$ and $KU_{1,\theta} = (g_2^{z_1}(g_2^{\hat{z}_1})^T)^{s_\theta} H_{1,\theta}^{-1}, KU_{2,\theta} = (g_2^{z_2}(g_2^{\hat{z}_2})^T)^{s_\theta} H_{2,\theta}^{-1}$, it is easy to check:

$K_{1,\theta} \cdot KU_{1,\theta} = K_1 \cdot (g_2^{w_1})^{r_\theta} \cdot H_{1,\theta} \cdot (g_2^{z_1}(g_2^{\hat{z}_1})^T)^{s_\theta} H_{1,\theta}^{-1} = K_1 \cdot (g_2^{w_1})^{r_\theta} \cdot (g_2^{z_1}(g_2^{\hat{z}_1})^T)^{s_\theta}$,

$K_{2,\theta} \cdot KU_{2,\theta} = K_2 \cdot (g_2^{w_2})^{r_\theta} \cdot H_{1,\theta}(g_2^{z_2}(g_2^{\hat{z}_2})^T)^{s_\theta} H_{1,\theta}^{-1} = K_2 \cdot (g_2^{w_2})^{r_\theta} \cdot (g_2^{z_2}(g_2^{\hat{z}_2})^T)^{s_\theta}$.

Thus, DK_1 and DK_2 also have the correct distribution.

Finally, we show that the challenge ciphertext is correct. For the challenge session key $K_0^* = g_T^s$, the challenge header $Hdr^* = (C_1^*, C_2^*, C_3^*, ctag^*)$ that $\mathcal{C}$ receives with a privileged set $S^* = \{ID_1^*, ID_2^*, ..., ID_n^*\}$ is of the following distribution: $C_1^* = g_1^s, C_2^* = (g_1^b)^s, C_3^* = (W_1^{ctag} \cdot \prod_{i=0}^n (g_1^{u_i})^{y_i})^s$. Thus, for the same privileged set S^*, the challenge header $Hdr = (C_1, C_2, C_3, C_4, ctag)$ that $\mathcal{A}$ is given from $\mathcal{C}$ is also well formed since:

$C_3 = (Z_1 \cdot \hat{Z}_1^{T^*})^s = (g_1^{z'_1}(g_1^b)^{z'_2}(g_1^{\beta\alpha})^{-T^*} \cdot (g_1^{\hat{z}'_1}(g_1^b)^{\hat{z}'_2}(g_1^{\beta\alpha}))^{T^*})^s$

$= (g_1^{z'_1 + \hat{z}'_1 \cdot T^*}(g_1^b)^{z'_2 + \hat{z}'_2 \cdot T^*})^s = (C_1^*)^{z'_1 + \hat{z}'_1 \cdot T^*} \cdot (C_2^*)^{z'_2 + \hat{z}'_2 \cdot T^*}$.

This completes the proof of Lemma 4. □

Theorem 4. *The resulting revocable IBE scheme $\prod_{RIBE}$ is adaptively secure under chosen plaintext attacks for $m = 1$, if the basic IBBE scheme $\prod_{IBBE}$ described in Section 3.1 is adaptively secure.*

Proof. If there exists an adversary $\mathcal{A}$ that attacks the above revocable IBE scheme $\prod_{\mathrm{RIBE}}$ with a non-negligible advantage, we will construct a PPT algorithm $\mathcal{C}$ to break the adaptive security of the IBE scheme $\prod_{\mathrm{IBE}}$ for $m = 1$

described in Section 3.1. In the following proof, we will omit some detailed discussion due to page limitation. Especially, we focus on the part that are different from the proof of Theorem 3.

At the beginning, $\mathcal{C}$ receives public parameters of the IBE scheme $\prod_{\text{IBE}}$: $(g_1, g_1^b, \boldsymbol{U}_1 = g_1^{\boldsymbol{u}_1 + b\boldsymbol{u}_2}, W_1 = g_1^{w_1 + bw_2}, g_T, g_2, g_2^{\boldsymbol{u}_1}, g_2^{\boldsymbol{u}_2}, g_2^{w_1}, g_2^{w_2}, g_1^{\beta\alpha}, g_2^{\beta\alpha_1}, g_2^{\beta\alpha_2}, g_2^{1/\beta})$ with $\boldsymbol{u}_1 = (u_{1,0}, u_{1,1}), \boldsymbol{u}_2 = (u_{2,0}, u_{2,1})$ for $m = 1$. During the process, $\mathcal{C}$ can access to the secret key generation oracle $\text{KeyGen}_{\text{IBE}}(\cdot)$, which can receive a private key of $SK_{ID} = \{K_1, K_2, K_3, K_4, K_5, ktag\}$ with: $K_1 = g_2^{\alpha_1} \cdot (g_2^{w_1})^r, K_2 = g_2^{\alpha_2} \cdot (g_2^{w_2})^r, K_3 = g_2^r$, $K_4 = ((g_2^{w_1})^{ktag} g_2^{u_{1,1}} (g_2^{-u_{1,0}})^{ID})^r, K_5 = ((g_2^{w_2})^{ktag} g_2^{u_{2,1}} (g_2^{-u_{2,0}})^{ID})^r$.

As the adversary $\mathcal{A}$ won't declare the target identity ID^* and time period T^* at the initial phase for the adaptive security model, $\mathcal{C}$ should first guess the right T^* that $\mathcal{A}$ submits the target identity in the challenge phase, which holds with probability $1/|\mathcal{T}|$ for polynomial-size $\mathcal{T}$.

Furthermore, for the challenge ID^*, $\mathcal{C}$ should guess the exact index of queries i^* that $\mathcal{A}$ issues ID^* to the $SKGen$ or $DKGen$ oracles for the first time. More precisely, $i^* \in \{1, 2, \cdots, q_1\}$ denotes that $\mathcal{A}$ first issues ID^* to $\mathcal{C}$ at the i^*-th identity for the private key query or the decryption key query in the **Key Extraction Phase 1**, where q_1 is the maximum number of private key queries and the decryption key queries before the challenge phase. $i^* = q_1 + 1$ denotes that $\mathcal{A}$ does not query any private key or decryption for ID^* before the challenge phase, but it can issue a private key query or decryption key query for ID^* in the **Key Extraction Phase 2**. $\mathcal{C}$ makes a random guess $i^* \in \{1, 2, \cdots, q_1, q_1 + 1\}$ for the adversary $\mathcal{A}$. Similar as in [33], the adversary $\mathcal{A}$ can be divided into the following two types: $\mathcal{A}$ is a Type-a adversary if $i^* \in \{1, 2, \cdots, q_1\}$; and $\mathcal{A}$ is a Type-b adversary if $i^* = q_1 + 1$. Note that $\mathcal{A}$ is a still a Type-b adversary even $\mathcal{A}$ has never queried ID^* for any private key or decryption key, in which case the target identity ID^* is already known by $\mathcal{C}$ in the challenge phase. In the rest of the proof, we assume that $\mathcal{C}$'s guess for i^* is right. Once $\mathcal{C}$ finds the guess is wrong, it terminates the simulation and outputs a random bit $\beta' \in \{0, 1\}$.

Setup: $\mathcal{C}$ first creates a binary tree BT with N leaves. It chooses a random leaf node η^* for a target identity ID^* in advance, that is, η^* is pre-assigned to ID^* that will be used in the challenge phase. $\mathcal{C}$ then chooses $z_1', z_2', \hat{z}_1', \hat{z}_2' \xleftarrow{R} \mathbb{Z}_p$ and computes: $Z_1 = g_1^{z_1 + b \cdot z_2} = g_1^{z_1'} (g_1^b)^{z_2'} / (g_1^{\beta\alpha})^{T^*}, \hat{Z}_1 = g_1^{\hat{z}_1 + b \cdot \hat{z}_2} = g_1^{\hat{z}_1'} (g_1^b)^{\hat{z}_2'} (g_1^{\beta\alpha})$, $g_2^{z_1} = g_2^{z_1'} / (g_2^{\beta\alpha_1})^{T^*}, g_2^{z_2} = g_2^{z_2'} / (g_2^{\beta\alpha_2})^{T^*}, g_2^{\hat{z}_1} = g_2^{\hat{z}_1'} g_2^{\beta\alpha_1}, g_2^{\hat{z}_2} = g_2^{\hat{z}_2'} g_2^{\beta\alpha_2}$. Finally, the public parameters PP of $\prod_{\text{RIBE}}$ is then sent to $\mathcal{A}$ with:
$PP = (g_1, g_1^b, \boldsymbol{U}_1, W_1, Z_1, \hat{Z}_1, g_T, g_2, g_2^{\boldsymbol{u}_1}, g_2^{\boldsymbol{u}_2}, g_2^{w_1}, g_2^{w_2}, g_2^{z_1}, g_2^{z_2}, g_2^{\hat{z}_1}, g_2^{\hat{z}_2})$.

Key Extraction Phase 1 for Type-a adversary: $\mathcal{A}$ is a Type-a adversary in the case of $i^* \leqslant q_1$. $\mathcal{C}$ will keep an integer i to count the number of queries from $\mathcal{A}$ for private key or decryption key up to the current time. Similar as in [38], we also classify type-a adversary more specifically: type-a-1 and type-a-0 adversary. $\mathcal{A}$ is a type-a-0 adversary if the private key of ID^* has been queried. Otherwise, $\mathcal{A}$ is a type-a-0 adversary if $\mathcal{A}$ has never queried the private key SK_{ID^*}. $\mathcal{C}$ interacts with $\mathcal{A}$ in the following steps:

– If this is a private key or decryption key query for an identity ID from a type-a-1 adversary $\mathcal{A}$, $\mathcal{C}$ performs as follows:
  - **Case** $i < i^*$: In this case, $ID \neq ID^*$, $\mathcal{C}$ will transfer ID to the oracle: $\text{KeyGen}_{\text{IBE}}(\cdot)$, and can get the private key $\{K_1, K_2, K_3, K_4, K_5, ktag\}$. $\mathcal{C}$ then assigns ID to a random leaf η from BT and stores ID in the leaf node η if ID is first issued to $\mathcal{C}$ for the private key or decryption key, otherwise, $\mathcal{C}$ uses the stored leaf node η for ID.
    * Private key query: If this is a private key query for an identity ID, for each node $\theta \in Path(BT, \eta)$, $\mathcal{C}$ can retrieve H_θ if it was defined. Otherwise, it chooses $H_\theta = (H_{1,\theta}, H_{2,\theta}) \xleftarrow{R} \mathbb{G}_2$ and stores H_θ in the node θ. $\mathcal{C}$ chooses r_θ randomly from $\mathbb{Z}_p$, and computes the private key according to θ:
      1. **If** $\theta \in Path(BT, \eta^*)$:
         $K_{1,\theta} = H_{1,\theta} \cdot (g_2^{w_1})^{r_\theta}, K_{2,\theta} = H_{2,\theta} \cdot (g_2^{w_2})^{r_\theta}$,
         $K_{3,\theta} = g_2^{r_\theta}$, $K_{4,\theta} = ((g_2^{w_1})^{ktag} \cdot g_2^{u_{1,1}} / (g_2^{u_{1,0}})^{ID})^{r_\theta}$,
         $K_{5,\theta} = ((g_2^{w_2})^{ktag} \cdot g_2^{u_{2,1}} / (g_2^{u_{2,0}})^{ID})^{r_\theta}$, $ktag := ktag$.
      2. **If** $\theta \notin Path(BT, \eta^*)$: $K_{1,\theta} = K_1 \cdot (g_2^{w_1})^{r_\theta} \cdot H_{1,\theta}, K_{2,\theta} = K_2 \cdot (g_2^{w_2})^{r_\theta} \cdot H_{2,\theta}$,
         $K_{3,\theta} = K_3 \cdot g_2^{r_\theta}$, $K_{4,\theta} = K_4 \cdot ((g_2^{w_1})^{ktag} \cdot g_2^{u_{1,1}} / (g_2^{u_{1,0}})^{ID})^{r_\theta}$,
         $K_{5,\theta} = K_5 \cdot ((g_2^{w_2})^{ktag} \cdot g_2^{u_{2,1}} / (g_2^{u_{2,0}})^{ID})^{r_\theta}$, $ktag := ktag$.
    * Decryption key query: If this is a decryption key query for an identity ID and time T, $\mathcal{C}$ selects random exponents $s, r \xleftarrow{R} \mathbb{Z}_p$ and creates the decryption $DK_{ID,T}$ as: $DK_1 = K_1 \cdot (g_2^{z_1}(g_2^{\hat{z}_1})^T)^s (g_2^{w_1})^r$, $DK_2 = K_2 \cdot (g_2^{z_2}(g_2^{\hat{z}_2})^T)^s (g_2^{w_2})^r$, $DK_3 = g_2^r, DK_3' = g_2^s$,
      $DK_4 = g_2^{r(u_{1,1} - (ID) \cdot u_{1,0} + ktag \cdot w_1)}, DK_5 = g_2^{r(u_{2,1} - (ID) \cdot u_{2,0} + ktag \cdot w_2)}$.
  - **Case** $i = i^*$: $\mathcal{C}$ identifies this identity ID as the target identity ID^* and stores ID^* in the leaf node η^*, which is pre-assigned in the **Setup** phase. $\mathcal{C}$ also assigns ID to a random tag: $ktag \in \mathbb{Z}_p$.
    * Private key query: If this is a private key query for an identity ID, for each node $\theta \in Path(BT, \eta^*)$, $\mathcal{C}$ can retrieve H_θ if it was defined. Otherwise, it chooses $H_\theta = (H_{1,\theta}, H_{2,\theta}) \xleftarrow{R} \mathbb{G}_2$ and stores H_θ in the node θ. $\mathcal{C}$ further chooses r_θ randomly from $\mathbb{Z}_p$, and computes:
      $K_{1,\theta} = H_{1,\theta} \cdot (g_2^{w_1})^{r_\theta}, K_{2,\theta} = H_{2,\theta} \cdot (g_2^{w_2})^{r_\theta}, K_{3,\theta} = g_2^{r_\theta}$, $ktag := ktag$,
      $K_{4,\theta} = ((g_2^{w_1})^{ktag} \cdot g_2^{u_{1,1}} / (g_2^{u_{1,0}})^{ID})^{r_\theta}, K_{5,\theta} = ((g_2^{w_2})^{ktag} \cdot g_2^{u_{2,1}} / (g_2^{u_{2,0}})^{ID})^{r_\theta}$.
    * Decryption key query: If this is a decryption key query for the identity $ID = ID^*$ and time $T \neq T^*$, $\mathcal{C}$ selects random exponents $s, r \xleftarrow{R} \mathbb{Z}_p$ and creates the decryption $DK_{ID,T}$ as:
      $DK_1 = (g_2^{z_1}(g_2^{\hat{z}_1})^T)^s (g_2^{w_1})^r (g_2^{\frac{1}{\beta}})^{-\frac{z_1' + T\hat{z}_1'}{T - T^*}}$,
      $DK_2 = (g_2^{z_2}(g_2^{\hat{z}_2})^T)^s (g_2^{w_2})^r (g_2^{\frac{1}{\beta}})^{-\frac{z_2' + T\hat{z}_2'}{T - T^*}}$, $DK_3 = g_2^r, DK_3' = g_2^s$,
      $DK_4 = g_2^{r(u_{1,1} - (ID) \cdot u_{1,0} + ktag \cdot w_1)}, DK_5 = g_2^{r(u_{2,1} - (ID) \cdot u_{2,0} + ktag \cdot w_2)}$.
  - **Case** $i > i^*$: $\mathcal{C}$ does the same process as in the case of $i < i^*$.

– If this is an update key query from a type-a-1 adversary $\mathcal{A}$, $\mathcal{C}$ performs as follows:, $\mathcal{C}$ first runs $\text{KUNode}(BT, RL, T)$ algorithm with the current revocation list RL and time T. For each node $\theta \in \text{KUNode}(BT, RL, T)$, $\mathcal{C}$ can

retrieve H_θ if it was defined. Otherwise, it chooses $H_\theta = (H_{1,\theta}, H_{2,\theta}) \xleftarrow{R} \mathbb{G}_2$ and stores H_θ in the node θ. Then it chooses $s_\theta \xleftarrow{R} \mathbb{Z}_p$, and computes:

- **If** $\theta \in Path(BT, \eta^*)$: $KU_{1,\theta} = (g_2^{z_1}(g_2^{\hat{z}_1})^T)^{s_\theta} \cdot H_{1,\theta}^{-1} \cdot (g_2^{\frac{1}{\beta}})^{-\frac{z'_1 + T\hat{z}'_1}{T-T^*}}$,
  $KU_{2,\theta} = (g_2^{z_2}(g_2^{\hat{z}_2})^T)^{s_\theta} \cdot H_{2,\theta}^{-1} \cdot (g_2^{\frac{1}{\beta}})^{-\frac{z'_2 + T\hat{z}'_2}{T-T^*}}$, $KU_{3,\theta} = g_2^{s_\theta} \cdot (g_2^{\frac{1}{\beta}})^{-\frac{1}{T-T^*}}$.
  REMARK. Note that ID^* must have been revoked before the time $T = T^*$, so there will be no θ such that $\theta \in Path(BT, \eta^*) \cap \text{KUNode}(BT, RL, T^*)$, as we restrict that the i^*-th identity query corresponding to ID^* is a private key query.
- **If** $\theta \notin Path(BT, \eta^*)$:
  $KU_{1,\theta} = (g_2^{z_1}(g_2^{\hat{z}_1})^T)^{s_\theta} H_{1,\theta}^{-1}, KU_{2,\theta} = (g_2^{z_2}(g_2^{\hat{z}_2})^T)^{s_\theta} H_{2,\theta}^{-1}, KU_{3,\theta} = g_2^{s_\theta}$.

Finally, $\mathcal{C}$ stores and outputs the update key KU_T to $\mathcal{A}$ with:
$KU_T = \{KU_{1,\theta}, KU_{2,\theta}, KU_{3,\theta}\}_{\theta \in \text{KUNode}(BT,RL,T)}$.

– If this is a private key or decryption key query for an identity ID from a type-a-0 adversary $\mathcal{A}$, $\mathcal{C}$ performs as follows:

- **Case** $i < i^*$: In this case, $ID \neq ID^*$, $\mathcal{C}$ will transfer ID to the oracle: $\text{KeyGen}_{\text{IBE}}(\cdot)$, and can get the private key $\{K_1, K_2, K_3, K_4, K_5, ktag\}$. $\mathcal{C}$ then assigns ID to a random leaf η from BT and stores ID in the leaf node η if ID is first issued to $\mathcal{C}$ for the private key or decryption key, otherwise, $\mathcal{C}$ uses the stored leaf node η for ID.
  * Private key query: If this is a private key query for an identity ID, for each node $\theta \in Path(BT, \eta)$, $\mathcal{C}$ can retrieve H_θ if it was defined. Otherwise, it chooses $H_\theta = (H_{1,\theta}, H_{2,\theta}) \xleftarrow{R} \mathbb{G}_2$ and stores H_θ in the node θ. $\mathcal{C}$ chooses r_θ randomly from $\mathbb{Z}_p$, and computes the private key as following:
    $K_{1,\theta} = K_1 \cdot (g_2^{w_1})^{r_\theta} \cdot H_{1,\theta}, K_{2,\theta} = K_2 \cdot (g_2^{w_2})^{r_\theta} \cdot H_{2,\theta}$,
    $K_{3,\theta} = K_3 \cdot g_2^{r_\theta}$, $K_{4,\theta} = K_4 \cdot ((g_2^{w_1})^{ktag} \cdot g_2^{u_{1,1}} / (g_2^{u_{1,0}})^{ID})^{r_\theta}$,
    $K_{5,\theta} = K_5 \cdot ((g_2^{w_2})^{ktag} \cdot g_2^{u_{2,1}} / (g_2^{u_{2,0}})^{ID})^{r_\theta}$, $ktag := ktag$.
  * Decryption key query: If this is a decryption key query for an identity ID and time T, $\mathcal{C}$ selects random exponents $s, r \xleftarrow{R} \mathbb{Z}_p$ and creates the decryption $DK_{ID,T}$ as: $DK_1 = K_1 \cdot (g_2^{z_1}(g_2^{\hat{z}_1})^T)^s (g_2^{w_1})^r$, $DK_2 = K_2 \cdot (g_2^{z_2}(g_2^{\hat{z}_2})^T)^s (g_2^{w_2})^r$, $DK_3 = g_2^r, DK'_3 = g_2^s$,
    $DK_4 = g_2^{r(u_{1,1} - (ID) \cdot u_{1,0} + ktag \cdot w_1)}, DK_5 = g_2^{r(u_{2,1} - (ID) \cdot u_{2,0} + ktag \cdot w_2)}$.
- **Case** $i = i^*$: $\mathcal{C}$ identifies this identity ID as the target identity ID^* and stores ID^* in the leaf node η^*, which is pre-assigned in the **Setup** phase. Note that there's no private key query for ID^* in the case of type-a-0 adversary. Given a decryption key query for the identity $ID = ID^*$ and time $T \neq T^*$, $\mathcal{C}$ selects random exponents $s, r, ktag \xleftarrow{R} \mathbb{Z}_p$ and creates the decryption $DK_{ID,T}$ as:
  $DK_1 = (g_2^{z_1}(g_2^{\hat{z}_1})^T)^s (g_2^{w_1})^r (g_2^{\frac{1}{\beta}})^{-\frac{z'_1 + T\hat{z}'_1}{T-T^*}}$,
  $DK_2 = (g_2^{z_2}(g_2^{\hat{z}_2})^T)^s (g_2^{w_2})^r (g_2^{\frac{1}{\beta}})^{-\frac{z'_2 + T\hat{z}'_2}{T-T^*}}$, $DK_3 = g_2^r, DK'_3 = g_2^s$,
  $DK_4 = g_2^{r(u_{1,1} - (ID) \cdot u_{1,0} + ktag \cdot w_1)}, DK_5 = g_2^{r(u_{2,1} - (ID) \cdot u_{2,0} + ktag \cdot w_2)}$.
- **Case** $i > i^*$: $\mathcal{C}$ does the same process as in the case of $i < i^*$.

– If this is an update key query from a type-a-0 adversary $\mathcal{A}$, $\mathcal{C}$ first runs KUNode(BT, RL, T) algorithm with the current revocation list RL and time T. For each node $\theta \in$ KUNode(BT, RL, T), $\mathcal{C}$ can retrieve H_θ if it was defined. Otherwise, it chooses $H_\theta = (H_{1,\theta}, H_{2,\theta}) \xleftarrow{R} \mathbb{G}_2$ and stores H_θ in the node θ. Then it chooses $s_\theta \xleftarrow{R} \mathbb{Z}_p$, and computes: $KU_{1,\theta} = (g_2^{z_1}(g_2^{\hat{z}_1})^T)^{s_\theta} H_{1,\theta}^{-1}, KU_{2,\theta} = (g_2^{z_2}(g_2^{\hat{z}_2})^T)^{s_\theta} H_{2,\theta}^{-1}, KU_{3,\theta} = g_2^{s_\theta}$.
Finally, $\mathcal{C}$ stores and outputs the update key KU_T to $\mathcal{A}$ with:
$KU_T = \{KU_{1,\theta}, KU_{2,\theta}, KU_{3,\theta}\}_{\theta \in \text{KUNode}(BT,RL,T)}$.
– If this is a revocation key query for an identity to be revoked ID and a revocation time T, then $\mathcal{C}$ updates the revocation list RL by running Revoke(ID, T, RL, ST) algorithm.

Key Extraction Phase 1 for Type-b adversary: In this case, $i^* = q_1 + 1$. For a Type-b adversary $\mathcal{A}$, there is no need for $\mathcal{C}$ to keep an integer i to count the number of queries from $\mathcal{A}$, as the target identity ID^* that $\mathcal{A}$ issues is only after the challenge phase, which is already known by $\mathcal{C}$. Similarly as the proof above, we have to classify Type-b adversary into Type-b-1 adversary and Type-b-0 adversary, depending on the private key of the challenge identity ID^* has been queried or not. $\mathcal{A}$ is said to be a type-b-1 adversary if the private key of ID^* has been queried (in key extraction phase 2). Otherwise, $\mathcal{A}$ is a type-b-0 adversary if $\mathcal{A}$ has never queried the private key SK_{ID^*}. For all the of queries from a Type-b adversary, including the privacy key, update key query, decryption key query and revocation key query, $\mathcal{C}$ acts almost identical to those in the key extraction phase 1 for a Type-a adversary, so they are omitted here.
Challenge: Now $\mathcal{A}$ sends the challenge identity ID^* and time T^* to $\mathcal{C}$. We assume that $\mathcal{C}$'s guess is right. If the guess is wrong, $\mathcal{C}$ terminates the simulation and outputs a random bit $\beta' \in \{0, 1\}$. $\mathcal{C}$ then sends the challenge identity ID^* to the challenger in the IND-CPA game of the IBE scheme $\prod_{\text{IBE}}$ and gets (Hdr^*, K^*) with $Hdr^* = (C_1^*, C_2^*, C_3^*, ctag^*)$. Note that (Hdr^*, K_0^*) is obtained from the challenger of $\prod_{\text{IBE}}$ by running $Encap$ algorithm with $(Hdr^*, K_0^*) =$ Encap(S^*), and K_1^* is a random element from the key space $\mathcal{K}$ of $\prod_{\text{IBE}}$. $\mathcal{C}$ sets $C_1 = C_1^*, C_2 = C_2^*, C_3 = (C_1^*)^{z_1' + T^* \hat{z}_1'} \cdot (C_2^*)^{z_2' + T^* \hat{z}_2'}, C_4 = C_3^*, ctag = ctag^*$ and sends $(Hdr = (C_1, C_2, C_3, C_4, ctag), K^*)$ to $\mathcal{A}$ as the challenge header and session key pair.
Key Extraction Phase 2: Same as **Key Extraction Phase 1**.
Guess: Finally, $\mathcal{A}$ outputs a guess $\beta' \in \{0, 1\}$, and $\mathcal{C}$ will transfer it to the challenger in the IND-CPA game of the IBE scheme $\prod_{\text{IBE}}$.

We note that during the simulation, $\mathcal{C}$ can access to the secret key generation oracle KeyGen$_{\text{IBE}}(\cdot)$ only for identities that $ID \neq ID^*$. $\mathcal{A}$ can query the private key for ID^* or the decryption key related to ID^* related to time $T \neq T^*$. In this case, $\mathcal{C}$ can also simulate the correct private key SK_{ID^*} or decryption key $DK_{ID^*,T}$, the distribution of which is identical to those in the real experiment. Furthermore, we can prove that the distribution of all transcriptions between $\mathcal{A}$ and $\mathcal{C}$ is same as those generated by real algorithm. The analysis is very similar to the proof of Theorem 3, and is omitted here.

This completes the proof of Theorem 4. □

6 Experimental and Evaluation

To demonstrate its practicality, we implement the proposed revocable IBBE scheme in Python 3.3.1 using the Charm 0.43 framework [3], a programming framework for cryptographic primitives. For the Type-3 pairings, we choose the default Miyaji-Nakabayashi-Takano elliptic curve group [26] with base field size 224 bits (MNT224) to establish our scheme, which can provide 96-bit security level [39]. All programs are running on a laptop with Intel® Core™ i3-4010U CPU@1.70GH and 4.0GB RAM using operating system 32-bit Ubuntu 13.04.

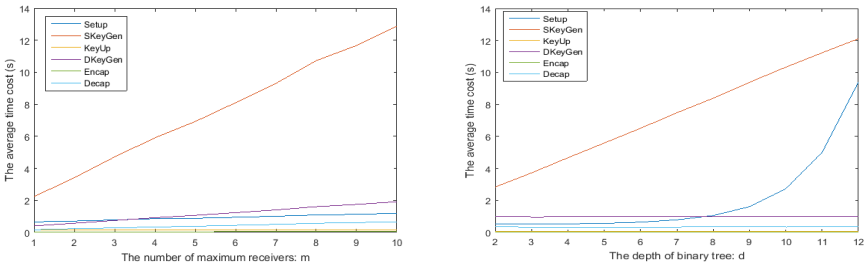

Fig. 2. Average time cost of all algorithms for different choices of m and d

Figure 2 and Figure 3 demonstrates the average time costs of all kinds of algorithms in our scheme. The data is measured by the benchmark tool provided by Charm. The average time cost is recorded after running each program using the MNT224 curve and other related parameter for 100 times. We perform the experiment in the following way: first setup the system with the maximum size of privileged set m and the total number of system users N, then generate a secret key, update key periodically via a public channel for the revocation list RL, generate the decryption key, encrypt a message given a privileged set S and decrypt the ciphertext.

Figure 2 (left) plots the influence of the maximum size of privileged set m on the efficiency of our scheme, where $m \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, the total number of system users is set to be $N = 64$, and there is no users is revoked (i.e., $RL = \emptyset$). For the **Encap** algorithm, the privileged identity is set as $S = \{\text{user1@email.com}, ..., \text{user}m\text{@email.com}\}$ for $m = 1, 2, ..., 10$. The user with identity user1@email.com that is not revoked can always correctly decrypt during this experiment. Note that for $m = 1$, this is an adaptively secure revocable IBE scheme, which is proven in Lemma 4. As we see in Figure 2(a), **SKGen** algorithm consumes the most computation costs. This is because $(3m+3)*(log_2 N)$ elements are needed for each private key, which will take $O(m*(log_2 N))$ exponentiations of $\mathbb{G}_2$. The operations on group $\mathbb{G}_2$ are more expensive than that of $\mathbb{G}_1$, about 11 times for exponentiations according to [1]. Note that the private key is generated only one time from PKG via a secure channel for each user in the system. For

the more frequent activity of **KeyUp** algorithm in the PKG side, the average time cost is bounded by just 0.14s.

Figure 2 (right) plots the time taken by the total number of system users $N \in \{2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8, 2^9, 2^{10}, 2^{11}, 2^{12}\}$, in which case the maximum size of privileged set for one encryption is set to be $m = 10$, and the revocation list $RL = \emptyset$. One can see that the time cost of the **Setup** algorithm grows exponentially in the depth d of the binary tree, which means the computation overhead is still linear in the number of system users N, where $N = 2^d$. The reason is that PKG should assign each identity into a random leaf node in the binary tree, and maintain the state information ST. It is worth mentioning that PKG can use a pseudorandom generator instead of storing the random values for each node in the binary tree, which is suggested in the Libert-Vergnaud scheme [25]. In terms of secret key generation algorithm **SKGen**, the compution overhead is linear in the depth of the binary tree, as the secret key is associated with the path from root to the leaf node. The compution overhead of the **DKGen**, **Decap** and **KeyUp** algorithms are all under 1 second, which are independent of the number of system users. Sepcifically, the average time cost of the **Encap** algorithm is just 31.2 ms, which makes our revocable IBBE scheme very efficient.

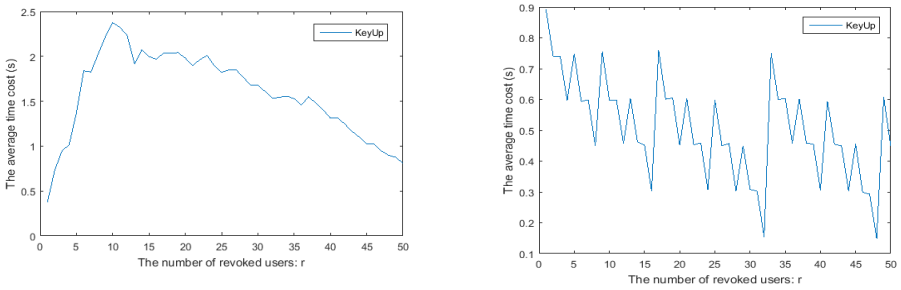

Fig. 3. Average time cost of KeyUp algorithm for different numbers of r

Figure 3 (left) demonstrates the time cost of **KeyUp** algorithm for different numbers of users to be revoked r from 1 to 50, where the total number of system users is set to be $N = 64$. Note that the random leaf node assignment technique [33] is used in our scheme. When a new user joins the system, it is assigned a random leaf node in the tree. In our implementation, each identity is pre-assigned a random leaf node via the built-in function `random.shuffle ()` in the system setup phase. The revocation list is $RL = \{\text{user1@email.com}, ..., \text{user}j\text{@email.com}\}$ for $j = 1, 2, ..., 50$. We can see that the time overhead of this **KeyUp** algorithm in all cases is upper bounded by 2.5s. More specifically, if each identity is assigned a leaf node in sequence without the random leaf node assignment technique [33], the average time costs of the **KeyUp** algorithm would be present in Figure 3 (right). We can say that $r = 32$ will have the least computation overhead, as there will be only one node in the `KUNode` algorithm of the **KeyUp** algorithm.

7 Conclusion

Providing an efficient revocation mechanism is necessary in the IBE setting and BE setting where a large number of users are involved, especially when considering practical deployments of these cryptosystems. It is more desirable that the sender does not need to know the revocation list, and only the receiver needs to check the revocation list of his credential to decrypt ciphertext. We further expand the study of scalable revocation methodology in the setting of IBBE, and then present a concrete instantiation of revocable IBBE scheme with DKER, which is motivated by a new revocable IBE scheme recently proposed in [38]. To build our revocable IBBE scheme, we first propose an adaptive IBBE scheme derived from [31]. Then we can construct a revocable IBBE scheme with a security reduction to the aforementioned IBBE scheme. The proposed scheme is very efficient both in terms of computation costs and communication overheads, as the ciphertext size is constant, independent of the number of recipients. Our scheme can withstand decryption key exposure, which is proved its semi-adaptive security under mild variants of the SXDH assumption. As a side contribution, we also present an adaptive secure revocable IBE scheme with DKER, which can be seen as a complementary of Watanabe et al.'s revocable IBE scheme [38].

Acknowledgment. Part of this work was done while Aijun Ge was visiting Institute for Advanced Study, Tsinghua University. The authors would like to thank Jianghong Wei and Jie Zhang for their helpful discussions on the `Charm` framework. We also thank anonymous reviewers of PKC 2019 for their insightful comments. The work is partially supported by the National Natural Science Foundation of China (No.61502529 and No.61502276) and the National Key Research and Development Program of China (No.2017YFA0303903).

References

1. Agrawal S., Chase M.: FAME: fast attribute-based message encryption. In *Proc. of the 24th ACM Conference on Computer and Communications Security* (*CCS 2017*), New York, NY, USA, pp.665-682. ACM (2017)
2. Attrapadung N. and Imai H.: Attribute-based encryption supporting direct/indirect revocation modes. In *IMA International Conference on Cryptography and Coding* (*IMCC 2009*), LNCS 5921, pp.278-300. Springer, Heidelberg (2009)
3. Akinyele J.A., Garman C., Miers I., et al.: Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3: 111-128, 2013
4. Boneh D., Boyen X.: Efficient selective-ID secure identity-based encryption without random oracles. In *Advances in Cryptology-EUROCRYPT 2004*, LNCS 3027, pp. 223-238. Springer, Heidelberg (2004)
5. Boneh D., Franklin M.: Identity-based encryption from the Weil pairing. In *Advances in Cryptology-CRYPTO 2001*, LNCS 2139, pp. 213-229. Springer, Heidelberg (2001)
6. Boldyreva A., Goyal V., Kumar G.: Identity-based encryption with efficient revocation. In *Proc. of the 15th ACM Conference on Computer and Communications Security* (*CCS 2008*), New York, NY, USA, pp.417-426. ACM (2008)

7. Chang D., Chauhan A., Kumar S., Sanadhya S.: Revocable identity-based encryption from codes with rank metric. In *Int. Conf. of Cryptographers' Track at the RSA (CT-RSA 2018)*, LNCS 10808, pp. 435-451. Springer, Heidelberg (2018)
8. Chen J., Lim H., Ling S., Su L., Wang H. Nguyen K.: Revocable identity-based encryption from lattices. In *Int. Conf. of Information Security and Privacy* (*ACISP 2012*), LNCS 7372, pp.390-403. Springer, Heidelberg (2012)
9. Chatterjee S., Menezes A.: On cryptographic protocols employing asymmetric pairings-the role of φ revisited. *Discrete Applied Mathematics*, 159(13): 1311-1322, 2011
10. Chen J., Wee H.: Semi-adaptive attribute-Based encryption and improved delegation for boolean formula. In *Int. Conf. of Security and Cryptography for Networks* (*SCN 2014*), LNCS 8642, pp.277-297. Springer, Heidelberg (2014)
11. Delerablée C.: Identity-based broadcast encryption with constant size ciphertexts and private keys. In *Advances in Cryptology -ASIACRYPT 2007*, *LNCS 4833*, pp.200-215. Springer, Heidelberg (2007)
12. Emura K., Seo J.H., Youn T.: Semi-generic transformation of revocable hierarchical identity-based encryption and its DBDH instantiation. *IEICE Trans*, 99(A(1)): 83-91, 2016
13. Fiat A., Naor M.: Broadcast encryption. In *Advances in Cryptology-CRYPTO 1993*, LNCS 773, pp. 480-491. Springer, Heidelberg (1994)
14. Galbraith S., Paterson K., Smart N.: Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16): 3113-3121, 2008
15. Gentry C., Waters B.: Adaptive security in broadcast encryption systems (with short ciphertexts). In *Advances in Cryptology-EUROCRYPT 2009*, *LNCS 5479*, pp. 171-188. Springer, Heidelberg (2009)
16. Jutla C., Roy A.: Shorter quasi-adaptive NIZK proofs for linear subspaces. In: Sako K. and Sarkar P. (Eds.) In *Advances in Cryptology -ASIACRYPT 2013*, *LNCS 8269*, pp.1-20. Springer, Heidelberg (2013)
17. Katsumata S., Matsuda T., Takayasu A.: Lattice-based revocable (hierarchical) identity-based encryption with decryption key exposure resistance. Cryptology ePrint Archive, Report 2018/420, 2018
18. Katz J., Sahai A., Waters B.: Predicate encryption supporting disjunctions, polynomial equations and inner products. In *Advances in Cryptology-EUROCRYPT 2008*, LNCS 4965, pp. 146-162. Springer, Heidelberg (2008)
19. Kogan N., Shavitt Y., Wool Avishai.: A Practical revocation scheme for broadcast encryption using smart cards. *ACM Transactions on Information and System Security*, 9(3): 325-351, 2006
20. Lee Kwangsu: Revocable hierarchical identity-based encryption with adaptive security. Cryptology ePrint Archive, Report 2016/749, 2016
21. Lee K., Lee D., Park J.: Efficient revocable identity-based encryption via subset difference methods. *Design Codes Cryptography* 85: 39-76, 2017
22. Lai J., Mu Y., Guo F., et al.: Full privacy-preserving and revocable ID-based broadcast encryption for data access control in smart city. *Pers Ubiquit Comput*, 21: 855-868, 2017.
23. Ling S., Nguyen K., Wang H., Zhang J.: Server-aided revocable predicate encryption: Formalization and lattice-based instantiation. CoRR, abs/1801.07844, 2018.
24. Lee K., Park S.: Revocable hierarchical identity-based encryption with shorter private keys and update keys. *Design Codes Cryptogrphy*, doi: 10.1007/s10623-017-0453-2, 2018

25. Libert B., Vergnaud D.: Adaptive-ID secure revocable identity-based encryption. In *Int. Conf. of Cryptographers' Track at the RSA* (*CT-RSA 2009*), *LNCS 5473*, pp.1-15. Springer, Heidelberg (2009)
26. Miyaji A., Nakabayashi M., Takano S.: Characterization of elliptic curve traces under FR-reduction. In *Int. Conf. of Information Security and Cryptology* (*ICISC 2000*), LNCS 2015, pp.90-108. Springer, Heidelberg (2000)
27. Nieto J., Manulis M., Sun D.: Fully private revocable predicate encryption. In *Int. Conf. of Information Security and Privacy* (*ACISP 2012*), LNCS 7372, pp.350-363. Springer, Heidelberg (2012)
28. Naor D., Naor M., Lotspiech J.: Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology-CRYPTO 2001*, LNCS 2139, pp. 41-62. Springer, Heidelberg (2001)
29. Nguyen K., Wang H., Zhang J.: Server-aided revocable identity-based encryption from lattices. In *Int. Conf. on Cryptology and Network Security* (*CANS 2016*), *LNCS 10052*, pp. 107-123. Springer, Heidelberg (2016)
30. Qin B., Deng R., Li Y., Liu S.: Server-aided revocable identity-based encryption. In *Proc. the 20th European Symposium on Research in Computer Security* (*ESORICS 2015*), LNCS 9326, pp.286-304. Springer, Heidelberg (2015)
31. Ramanna C. Somindu: More efficient construction for inner product encryption. In *Int. Conf. of Applied Cryptography and Network Security.* (*ACNS 2016*), LNCS, 9696, pp. 231-248. Springer, Heidelberg (2016) The full version, http://eprint.iacr.org/2016/356
32. Susilo W., Chen R., Guo F., et al.: Recipient rovocable identity-based broadcast encryption, or how to revoke some recipient in IBBE without knowledge of the plaintext. In *Proc. of the 11th ACM on Asia Conference on Computer and Communications Security* (*AsiaCCS 2016*), Xi'an, China, pp.201-210. ACM (2016)
33. Seo J. Hong and Emura Keita: Revocable identity-based encryption revisited: Security model and construction. In *Proc. the 16th Int. Conf. on Public Key Cryptography* (*PKC 2013*), LNCS 7778, pp. 216-234. Springer, Heidelberg (2013)
34. Seo J. Hong and Emura Keita: Efficient delegation of key generation and revocation functionalities in Identity-Based encryption. In *Int. Conf. of Cryptographers' Track at the RSA (CT-RSA 2013)*, LNCS 7779, pp. 343-358. Springer, Heidelberg (2013)
35. Seo J. Hong and Emura Keita: Revocable hierarchical identity-based encryption: History-free update, security against insiders, and short ciphertexts. In *Int. Conf. of Cryptographers' Track at the RSA (CT-RSA 2015)*, LNCS 9048, pp. 106-123. Springer, Heidelberg (2015)
36. Seo J. Hong and Emura Keita: Adaptive-id secure revocable hierarchical identity-based encryption. In *Int. Conf. on Information and Computer Security* (*IWSEC 2015*), *LNCS 9241*, pp. 21-38. Springer, Heidelberg (2015)
37. Waters B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In *Advances in Cryptology-CRYPTO 2009*, LNCS 5677, pp. 619-636. Springer, Heidelberg (2009)
38. Watanabe Y., Emura K., Seo J.: New revocable IBE in prime order groups: adaptively secure, decryption key exposure resistant and with short public parameters. In *Int. Conf. of Cryptographers' Track at the RSA (CT-RSA 2017)*, LNCS 10159, pp. 432-449. Springer, Heidelberg (2017) The full version, http://eprint.iacr.org/2016/1094
39. Yang B., Yang K., Qin Y., Zhang Z., Feng D.: DAA-TZ: an efficient DAA scheme for mobile devices using ARM trustzone. In *Int. Conf. on Trust and Trustworthy Computing* (*Trust 2015*), *LNCS 9229*, pp.209-227. Springer, Heidelberg (2015)

A Proof of Lemma 3 in Section 3.2

Proof. Given a PPT adversary $\mathcal{A}$ achieving a non-negligible difference ε in advantage between Game_q and $\text{Game}_{\text{Final}}$, we will create a PPT algorithm $\mathcal{C}$ to break the ADDH1 assumption. Let $(g_1, g_1^{\mu}, g_1^{\alpha_2}, g_1^{\beta\alpha}, g_2, g_2^{\alpha}, g_2^{\beta\alpha}, g_2^{\beta\alpha_2}, g_2^{1/\beta}, Z = g_1^{\mu\alpha_2+\eta})$ be the instance of ADDH1 problem in $\mathcal{G}$ that $\mathcal{C}$ has to solve, i.e., to decide whether $\eta = 0$ or a random value in $\mathbb{Z}_p^*$. Note that in Game_q, all the user keys returned to $\mathcal{A}$ are semi-functional and so is the challenge header and session key. $\mathcal{C}$ will simulate either Game_q or $\text{Game}_{\text{Final}}$ with $\mathcal{A}$, depending on the value of η.

Setup: At the beginning, $\mathcal{C}$ chooses random exponents $\boldsymbol{u}_1 = (u_{1,0}, \cdots, u_{1,m})$, $\boldsymbol{u}_2 = (u_{2,0}, \cdots, u_{2,m}), w_1, w_2 \xleftarrow{R} \mathbb{Z}_p$ and $b \xleftarrow{R} \mathbb{Z}_p^*$, and sets the public parameters PP :

$$g_1 := g_1, g_1^b, \boldsymbol{U}_1 := g_1^{\boldsymbol{u}_1 + b\boldsymbol{u}_2}, W_1 := g_1^{w_1 + bw_2}, g_T := e(g_1, g_2^{\alpha}),$$
$$g_2 := g_2, g_2^{\boldsymbol{u}_1}, g_2^{\boldsymbol{u}_2}, g_2^{w_1}, g_2^{w_2}, g_1^{\beta\alpha}, g_2^{\beta\alpha_1} := g_2^{\beta\alpha}/(g_2^{\beta\alpha_2})^b, g_2^{\beta\alpha_2}, g_2^{1/\beta}.$$

Note that this implicitly sets $\alpha_1 := \alpha - b\alpha_2$, and the secret exponents (α_1, α_2) in MSK are not available to $\mathcal{C}$.

Key Extraction: When the adversary $\mathcal{A}$ requests a secret key extract query for an identity $ID \in \mathbb{Z}_p$, $\mathcal{C}$ creates a semi-functional key. It does this by choosing random exponents $r, \gamma', ktag_1, \cdots, ktag_m \xleftarrow{R} \mathbb{Z}_p$, which implicitly sets $\gamma := \gamma' + b\alpha_2$. The semi-functional key elements are computed as:
$K_1 = g_2^{\alpha_1}(g_2^{w_1})^r g_2^{\gamma} = g_2^{\alpha}(g_2^{w_1})^r g_2^{\gamma'}$, $K_2 = g_2^{\alpha_2}(g_2^{w_2})^r / g_2^{\gamma' b^{-1}} = (g_2^{w_2})^r / g_2^{\gamma' b^{-1}}$, $K_3 = g_2^r$.
For $i = 1, 2, ..., m$:
$K_{4,i} = ((g_2^{w_1})^{ktag_i} \cdot g_2^{u_{1,i}} / (g_2^{u_{1,0}})^{(ID)^i})^r$, $K_{5,i} = ((g_2^{w_2})^{ktag_i} \cdot g_2^{u_{2,i}} / (g_2^{u_{2,0}})^{(ID)^i})^r$.
This is a properly distributed semi-functional key, which can be easily verified.

Challenge: Once the public parameters PP and the keys for all key extraction queries are given, $\mathcal{A}$ provides a challenge privileged set $S^* = \{ID_1, ID_2, ..., ID_n\}$. $\mathcal{C}$ first computes the vector $\boldsymbol{y} = (y_0, y_1, \cdots, y_m)$ according to S^* as the coefficient from $P_{S^*}[Z] = \prod_{ID_j \in S^*}(Z - ID_j)$. It then picks randomly $s, ctag \in \mathbb{Z}_p$, and computes the challenge header $Hdr = (C_1, C_2, C_3, ctag)$ as follows:
$C_1 = g_1^s \cdot g_1^{\mu}, C_2 = g_1^{sb}, C_3 = (W_1^{ctag} \cdot \prod_{i=0}^{n}(g_1^{u_i})^{y_i})^s \cdot g_1^{\mu(\langle \boldsymbol{y}, \boldsymbol{u}_1 \rangle + ctag \cdot w_1)})$. In addition, the challenge session key K is set to be: $K = {g_T}^s \cdot e(g_1^{\mu}, g_2^{\alpha})/e(Z, g_2^b)$.

One can verify that the challenge header $Hdr = (C_1, C_2, C_3, ctag)$ has proper semi-functional forms. Furthermore, if $Z = g_1^{\mu\alpha_2}$ (i.e., $\eta = 0$), then K is a properly distributed semi-functional session key. In this case, $\mathcal{C}$ has properly simulated Game_q. If η is a random value in $\mathbb{Z}_p^*$, which means $Z = g_1^{\mu\alpha_2+\eta}$ is a random element in G_1, then K is uniformly distributed and is independent of all other components. In this case, $\mathcal{C}$ has properly simulated $\text{Game}_{\text{Final}}$.

Guess: Eventually, the adversary $\mathcal{A}$ will output a guess β' of β. The challenger $\mathcal{C}$ then outputs 0 to guess that $Z = g_1^{\mu\alpha_2}$ if $\beta' = \beta$; otherwise, it outputs 1 to indicate that $Z = g_1^{\mu\alpha_2+\eta}$ is a random element of G_1. Also, $\mathcal{C}$ simulates Game_q if $\eta = 0$ and $\text{Game}_{\text{Final}}$ if $\eta \in {}_R\mathbb{Z}_p^*$. Therefore, $\mathcal{C}$ can use $\mathcal{A}$'s output to distinguish $Z = g_1^{\mu\alpha_2}$ from random with the same advantage that $\mathcal{A}$ has in distinguishing Game_q from $\text{Game}_{\text{Final}}$.

This completes the proof of Lemma 3. □