

Revocable IBE Systems with Almost Constant-size Key Update

Le Su, Hoon Wei Lim, San Ling, Huaxiong Wang

Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
lsu1@e.ntu.edu.sg, {hoonwei, lingsan, hxwang}@ntu.edu.sg

Abstract

Identity-based encryption (IBE) has been regarded as an attractive alternative to more conventional certificate-based public key systems. It has recently attracted not only considerable research from the academic community, but also interest from the industry and standardization bodies. However, while key revocation is a fundamental requirement to any public key systems, not much work has been done in the identity-based setting. In this paper, we continue the study of *revocable* IBE (RIBE) initiated by Boldyreva, Goyal, and Kumar. Their proposal of a selective secure RIBE scheme, and a subsequent construction by Libert and Vergnaud in a stronger adaptive security model are based on a binary tree approach, such that their key update size is *logarithmic* in the number of users. We ask the question of whether or not the key update size could be further reduced by using a cryptographic accumulator. We show that, indeed, the key update material can be made constant with some small amount of auxiliary information, through a novel combination of the Lewko and Waters IBE scheme and the Camenisch, Kohlweiss, and Soriente pairing-based dynamic accumulator.

1 Introduction

It is sometimes necessary to remove keying material from use prior to the end of its normal cryptoperiod (or key lifetime) for reasons that include key compromise, removal of an entity from an organization, and so on. This process is known as *key revocation* and is used to explicitly revoke a symmetric key or the public key of a key pair, although the private key associated with the public key is also revoked [5]. Public key revocation in a conventional, certificate-based public key infrastructure (PKI) has been well studied and understood. A widely deployed revocation mechanism is through the use of a Certificate Revocation List (CRL) [22]. Alternatively, an Internet protocol called the Online Certificate Status Protocol (OCSP) is used to check if a certificate has been revoked [28].

In this paper, we study public key revocation in an *identity-based encryption* (IBE) system. The idea of using an identity (or identifier) as a public key was originally conceived by Shamir [34], and subsequently realized by Cocks [15] using quadratic residues, and Boneh and Franklin [10] using pairings on elliptic curves. One very appealing property of IBE is that it alleviates cumbersome certificate management in a traditional PKI. To securely send a message to an intended receiver, the sender no longer needs to look up for the public key certificate associated with the receiver, but simply encrypts the message directly using a common set

of public system parameters and the receiver’s identifier, such as email address. Over the past decade, pairing-based IBE has not only received considerable attention from academic researchers, but also attracted commercial interest from Mitsubishi, Noretech, Trend Micro, Voltage Security, and Gemplus [18], for example. Moreover, identity-based cryptographic techniques using pairings are currently undergoing standardization through the IEEE 1363.3 and the IETF S/MIME working groups. However, very few studies, for example [7, 14, 27, 33], have been devoted to key revocation thus far.

1.1 Motivation

Unlike a certificate-based public key, which is simply a random-looking string, a public key in the IBE setting is a user’s identity. This hinders “explicit” revocation of an identity-based public key using conventional revocation mechanisms. Instead, one typically adopts a more “implicit” approach by periodically updating the corresponding private key after a pre-defined validity period, while letting the old private key expire automatically and keeping the public key (identity) unchanged. One trivial way of achieving this is by encoding a current time period into an identity during encryption. This forces a decryptor to regularly obtain her private key (corresponding to the current time period) from a key authority [10]. However, such an approach does not scale well because the key authority has to generate new keys for all the remaining non-revoked users at the beginning of each time period. Further, distribution of private keys requires establishment of secure channels between the key authority and the users. This may not be always feasible for every user.

A more desirable approach is to let the key authority broadcast some *public* information, from which the users can perform key update themselves without interacting with the key authority. Clearly, we must ensure that the broadcast information is useful only to non-revoked users, but meaningless to those who have been revoked. Hanaoka et al. [21] proposed one of the first IBE schemes that supports a *non-interactive* key revocation approach. However, their scheme requires each user to possess a special tamper-resistant hardware device that stores a secret helper key used for key update—a requirement that is likely to hinder practical deployment of the scheme.

Subsequently, Boldyreva et al. [7] proposed a scheme that obviates the need for special devices and significantly reduces the complexity of key update information from linear to *logarithmic* in the number of users. They cleverly combined fuzzy IBE (FIBE) [32] with binary tree structure, which has previously been used to improve the efficiency of certificate revocation in a traditional PKI [1, 29]. By making use of the concept of FIBE, Boldyreva et al. gave a construction they called revocable IBE (RIBE), in which a message is encrypted under two attributes, namely identity id and time t . Correspondingly, the associated decryption key comprises two components, of which the identity part is fixed (also called a long-term private key), while the time part is updated after each time period (or epoch). In order to revoke a user, the key authority simply stops issuing key update for that user in the next time period. Without the latest key update, a revoked user will no longer be able to decrypt any ciphertext generated beyond the current (expiring) time period. As with [1, 29], a binary tree can then be used to more efficiently (logarithmically) represent all the remaining non-revoked users than simply listing all the revoked or non-revoked users. In Boldyreva et al.’s RIBE scheme, each user’s id is assigned to a leaf node in the binary tree and her long-term private key is generated according to the key material on each node along the path from the user’s leaf node to the root. To decrypt a message encrypted under id and t , the user needs an updated

decryption key (associated with t) that can be derived from the key material associated with any one of the nodes along the path from her *id* leaf node to the root. Hence, if a user has been revoked, such key material will not be made available in the key update broadcast by the key authority.

However, Boldyreva et al.’s scheme was proven secure in a *selective* security model, which is widely accepted as a weaker model in comparison with an *adaptive* security model. The former requires the adversary to announce the target identity and time at the beginning of a security game simulated in the model, while the latter has no such restriction. Nevertheless, Libert and Vergnaud [27] showed that adaptive security is possible. They proposed an RIBE scheme which has key update size that is also logarithmic using a similar binary tree technique, while proving their scheme to be adaptively secure. However, they achieved this at the expense of increasing the size of public parameters from constant to linear in the number of users.

The goal of this paper is then to study whether or not we could further reduce the key update size while retaining the adaptive security requirement. We give an affirmative answer and provide a concrete construction which relies on only constant-size of key update material along with some auxiliary information, through a novel approach that combines IBE with the concept of a cryptographic *accumulator*.

1.2 Our Approach

The key component in our approach that enables efficient key revocation and update is a pairing-based cryptographic accumulator by Camenisch et al. [13]. An accumulator, originally introduced by Benaloh and de Mare [6] as an alternative to digital signatures for secure decentralized and distributed protocols, is an algorithm that “compresses” a large set of values into a single, short value with the following two basic properties:

- For each accumulated value, it is possible to compute a *witness* that can be used to prove that a given value was indeed incorporated into the accumulator;
- Whenever a value is added or removed from the accumulator, all witnesses correspond to all the remaining values in the updated accumulator need to be re-computed as well.

The accumulator proposed by Camenisch et al. [13] is designed to address the problem of revocation of *anonymous credentials*—to efficiently prove that a hidden value has been accumulated. By making use of techniques from broadcast encryption developed by Boneh et al. [11], Camenisch et al.’s accumulator has a very nice property that allows update of witnesses to be performed very efficiently. Only one multiplication is required for addition or deletion of a value from the accumulator. Further, update of witnesses can be delegated to an untrusted entities without compromising the security of an anonymous credential system.

In this work, we take a different approach from that of [13] when considering public key revocation in the IBE setting. We combine Lewko and Waters’ IBE scheme [25] with Camenisch et al.’s accumulator [13] in a particular way. Conventionally, an accumulator is used for revocation of credentials or keys in an anonymous authentication system that is based on concepts such as, group signatures or anonymous e-cash. Also, a witness is independent of the authentication system, in the sense that it is not directly used for authentication, but rather is typically used to convince a verifier that a user has or has not been revoked through a zero-knowledge proof. On the other hand, in our approach, we integrate an accumulator with a public key encryption scheme, such that the accumulator is associated with ciphertexts, while witnesses are associated with decryption keys. Particularly, our encryption algorithm

takes as input a message, an identity, and an up-to-date accumulator for current time period t , such that a target recipient is able to decrypt the resulting ciphertext using an up-to-date witness. That is, the decryption would succeed only if the recipient has not already been revoked at time t . Here, a user’s decryption key comprises an identity-based key and a witness. During decryption, both the witness and the ciphertext component containing the accumulator are required to cancel out a blinding factor of the message.

Our approach of combining the IBE scheme of [25] and the accumulator of [13] requires a careful treatment. As described, one of the basic properties of an accumulator is that a witness can be used to prove that an associated value has been accumulated. Translating this into our design of RIBE, it turns out that a *collusion attack* is possible if a decryption key comprising a witness and an identity-based key is formed in a naïve manner. This is because a revoked user can collude with a non-revoked user, such that a valid witness (of the non-revoked user) can be used by the revoked user (together with her own identity-based key) to decrypt ciphertexts that she no longer has authorized access. To address this, for each user, we introduce a new secret component¹ that is associated with the accumulator and a witness, such that the secret component must be used to cancel out the blinding factor. We then bind the secret component to the user’s identity-based key. Since the identity-based key is randomized for each user, two users will no longer be able to collude to share one of their witnesses to perform decryption.

Our key update method (to be performed by the key authority) through an accumulator differs in two aspects from that of existing RIBE schemes [7, 27] which make use of a binary tree. First, we simply update an accumulator according to the updated revocation list at a new time period t' , generate a new witness associated with t' , and create a signature over the updated accumulator. (We note that we also add t' into the accumulator to ensure that the accumulated value for each time period is always distinct even if there is no change in the revocation list between two successive time periods.) However, in the binary tree method, they first need to identify the minimal set of nodes (in the tree) for which key update needs to be published so that only non-revoked users are able to decrypt ciphertexts generated at time t' . For each node in the identified set, they then generate some key material required to update a decryption key. Second, in terms of communication overhead, our key update material comprises just a (short) accumulator, a witness, and a signature. Hence, the complexity of the size of our key update improves significantly from $O(\log(n))$ in the binary tree approach for n users to $O(1)$ with some relatively small amount of bookkeeping information by using accumulator. (We provide further details on the efficiency of our scheme in Section 3.4.)

In our security analysis, we adopt the Waters dual system encryption methodology [35]. As with that of [7, 27], we consider two types of adversaries: Type I adversaries that are *not* allowed to request for the private key of a target identity throughout the entire security game; and Type II adversaries that are allowed to make a query on the private key of a target identity, provided that the queried identity must subsequently be revoked before the challenge time. However, we show that our accumulator-based approach has simpler and tighter security proofs than those of a binary-tree method. To simulate a security game in the latter setting, extra care needs to be taken in order to appropriately answer any private key query that is associated with a node in the tree. Particularly, to achieve adaptive security, the simulator has to guess the position of the target identity-time pair in the tree beforehand.

¹Coincidentally, the secret component we adopt here is also used as a user private key in Boneh et al.’s broadcast encryption scheme [11].

This causes some loss of reduction in their security proofs. Such concerns do not exist in our proofs.

In this paper, we also show how our accumulator-based revocation technique can be improved and extended in several ways. First, we show how our RIBE system can be extended to handle any arbitrary number of users. Second, we sketch an RIBE scheme that achieves forward-security, in the sense that compromise of a decryption key at time t would not leak any useful information about other decryption keys for other times $t'_i \neq t$. Moreover, we describe how our technique can be used to construct a revocable ABE (RABE) scheme. These are elaborated in Section 4.

1.3 Other Related Work

After the work by Boldyreva et al. [7] and Libert and Vergnaud [27], there have been proposals on various instances of functional encryption (generalization of IBE) schemes that support revocation, such as *revocable attribute-based encryption* (RABE) and *revocable predicate encryption* (RPE) [2, 3, 19]. We note that the revocation method in the schemes of [3, 19] is different from that of existing RIBE schemes. In the RABE schemes, the users themselves are the ones who enforce key revocation instead of the key authority. This is known as *sender-local revocation* and is achieved by taking as input a revocation list during encryption. A receiver's private key can decrypt a ciphertext only if her identity has not been included in the revocation list. This way, the users are not required to perform any private key update as with that in [7]. In [2], a key revocation system combining both approaches from [7] and [3] was proposed.

Moreover, there exist proposals on revocable IBE schemes with *mediators* [4, 9, 17, 26]. Here, a mediator is a semi-trusted authority that helps users to decrypt ciphertexts. If a user has been revoked, the mediator simply stops decrypting for the user. Such an approach, while interesting, does not seem to be satisfactory as it requires interactions between the mediator and the users for decryption of each ciphertext.

Recently, Chen et al. [14] proposed an RIBE scheme based on *lattices* under a similar security model as [7]. Also, Seo and Emura [33] gave a pairing-based RIBE construction and proved that it is secure under a new security model, which considers not only exposure of long-term private keys, but also *exposure of decryption keys*² (associated to each time period).

1.4 Outline

The paper is organized as follow: Section 2 introduces some background on bilinear maps and assumptions used in our security analysis. It also describes the definitions for RIBE and other cryptographic primitives required in our construction. In Section 3, we present our RIBE construction and its security proofs. In Section 4, we discuss and sketch some extensions to our scheme. We conclude and highlight some open problems in Section 5.

²In the security model of [33], an adversary is allowed to make decryption key queries, in addition to the conventional private key queries allowed in [7].

2 Preliminaries

2.1 Composite Order Bilinear Groups

Composite order bilinear groups, originally introduced in [12], are defined by a group generator \mathcal{G} which takes as input a security parameter λ and outputs the description of a bilinear map G . This includes (N, G, G_T, e) , where $N = p_1 p_2 p_3$ and p_1, p_2, p_3 are distinct primes, G and G_T are cyclic groups of order N , and $e : G \times G \rightarrow G_T$ is a bilinear map such that:

- (Bilinear) $\forall g, h \in G, a, b \in \mathbb{Z}_N, e(g^a, h^b) = e(g, h)^{ab}$
- (Non-degenerate) $\exists g \in G$ such that $e(g, g)$ has order N in G_T .

In addition to the above properties, we also require that the group operations in G and G_T , together with the bilinear map e , are polynomial time computable with respect to the security parameter λ . We also assume that the group descriptions of G and G_T include the group generators. For ease of exposition, we let $G_{p_1}, G_{p_2}, G_{p_3}$ denote the subgroups of order p_1, p_2 and p_3 in G respectively. We also note the orthogonality property of our bilinear map: that is, $e(h_i, h_j)$ is the identity element in G_T whenever $h_i \in G_{p_i}$ and $h_j \in G_{p_j}$ for $i \neq j$. This property of the three subgroups will be a principal tool in our construction and proofs.

2.2 Complexity Assumptions

We now state four complexity assumptions on which our construction and security proofs rely. The first three assumptions are the same those used in [25]. They are static (not dependent on the number of queries made by an adversary) and can be proved using the theorem introduced by Katz, Sakai and Waters [23]. The fourth assumption is called the Oracle Bilinear Diffie Hellman Exponent (OBDHE) assumption, which was used in [30] to prove the security of a broadcast encryption scheme. It is a modified version of the standard decisional BDHE problem such that it provides the adversary with an additional query oracle. In the assumptions below, we let $G_{p_i p_j}$ denote the subgroup of order $p_i p_j$ in G .

Assumption 1. (*Subgroup decisional problem for 3 primes*) *Given a group generator \mathcal{G} , we define the following distribution:*

$$\begin{aligned} \mathbb{G} &= (N = p_1 p_2 p_3, G, G_T, e) \xleftarrow{R} \mathcal{G}, \\ g &\xleftarrow{R} G_{p_1}, X_3 \xleftarrow{R} G_{p_3}, \\ D &= (\mathbb{G}, g, X_3), \\ T_1 &\xleftarrow{R} G_{p_1 p_2}, T_2 \xleftarrow{R} G_{p_1}. \end{aligned}$$

We define the advantage of an algorithm \mathcal{A} in breaking Assumption 1 to be:

$$\text{Adv}_{\mathcal{G}, \mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(D, T_1) = 1] - \Pr[\mathcal{A}(D, T_2) = 1]|.$$

Definition 1. *We say that \mathcal{G} satisfies Assumption 1 if $\text{Adv}_{\mathcal{G}, \mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm \mathcal{A} .*

Assumption 2. *Given a group generator \mathcal{G} , we define the following distribution:*

$$\begin{aligned}
\mathbb{G} &= (N = p_1 p_2 p_3, G, G_T, e) \stackrel{R}{\leftarrow} \mathcal{G}, \\
g, X_1 &\stackrel{R}{\leftarrow} G_{p_1}, X_2, Y_2 \stackrel{R}{\leftarrow} G_{p_2}, X_3, Y_3 \stackrel{R}{\leftarrow} G_{p_3}, \\
D &= (\mathbb{G}, g, X_1 X_2, X_3, Y_2 Y_3), \\
T_1 &\stackrel{R}{\leftarrow} G, T_2 \stackrel{R}{\leftarrow} G_{p_1 p_3}.
\end{aligned}$$

We define the advantage of an algorithm \mathcal{A} in breaking Assumption 2 to be:

$$\text{Adv}_{2\mathcal{G},\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(D, T_1) = 1] - \Pr[\mathcal{A}(D, T_2) = 1]|.$$

Definition 2. We say that \mathcal{G} satisfies Assumption 2 if $\text{Adv}_{2\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm \mathcal{A} .

Assumption 3. Given a group generator \mathcal{G} , we define the following distribution:

$$\begin{aligned}
\mathbb{G} &= (N = p_1 p_2 p_3, G, G_T, e) \stackrel{R}{\leftarrow} \mathcal{G}, \alpha, s \stackrel{R}{\leftarrow} \mathbb{Z}_N, \\
g &\stackrel{R}{\leftarrow} G_{p_1}, X_2, Y_2, Z_2 \stackrel{R}{\leftarrow} G_{p_2}, X_3 \stackrel{R}{\leftarrow} G_{p_3}, \\
D &= (\mathbb{G}, g, g^\alpha X_2, X_3, g^s Y_2, Z_2), \\
T_1 &= e(g, g)^{\alpha s}, T_2 \stackrel{R}{\leftarrow} G_T.
\end{aligned}$$

We define the advantage of an algorithm \mathcal{A} in breaking Assumption 3 to be:

$$\text{Adv}_{3\mathcal{G},\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(D, T_1) = 1] - \Pr[\mathcal{A}(D, T_2) = 1]|.$$

Definition 3. We say that \mathcal{G} satisfies Assumption 3 if $\text{Adv}_{3\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm \mathcal{A} .

We now define the Diffie-Hellman computation oracle required for the OBDHE assumption.

Definition 4. The Diffie Hellman computation oracle $\mathcal{O}_{g,e}^{DH}$ takes as inputs $u, v \in \mathbb{G}$ and outputs $w \in \mathbb{G}$ such that $e(u, v) = e(g, w)$.

We let prime p_1 be the group order of G and define the OBDHE assumption as follow:

Assumption 4. (Oracle Bilinear Diffie-Hellman Exponent) Given a group generator \mathcal{G} , we define the following distribution:

$$\begin{aligned}
\mathbb{G} &= (G, G_T, e) \stackrel{R}{\leftarrow} \mathcal{G}, \alpha \stackrel{R}{\leftarrow} \mathbb{Z}_N, \\
g, f &\stackrel{R}{\leftarrow} G, \\
D &= (\mathbb{G}, f, g, g^\alpha, g^{\alpha^2}, \dots, g^{\alpha^\ell}, g^{\alpha^{\ell+2}}, \dots, g^{\alpha^{2\ell}}) \\
T_1 &= e(g^{\alpha^{\ell+1}}, f), T_2 \stackrel{R}{\leftarrow} G_T.
\end{aligned}$$

We define the advantage of an algorithm \mathcal{A} in breaking Assumption 4 to be:

$$\text{Adv}_{4\mathcal{G},\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(D, T_1) = 1] - \Pr[\mathcal{A}(D, T_2) = 1]|$$

given that \mathcal{A} has access to the $\mathcal{O}_{g,e}^{DH}$ oracle.

Definition 5. We say that \mathcal{G} satisfies Assumption 4 if $\text{Adv}_{4\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm \mathcal{A} .

2.3 Revocable Identity Based Encryption

We are now ready to define RIBE [7]. Let \mathcal{M} denote a message space, \mathcal{I} denote an identity space, and \mathcal{T} denote a time space. Assume that the sizes of $\mathcal{M}, \mathcal{I}, \mathcal{T}$ are all polynomial in the security parameter. Each algorithm within RIBE is run by either one of three types of parties—key authority, sender or receiver. The key authority maintains a revocation list RL and state ST . An algorithm is called stateful if it updates RL or ST . We treat time as discrete as opposed to continuous.

Definition 6 (RIBE). *An identity-based encryption scheme with efficient revocation or simply revocable IBE (RIBE) scheme has seven PPT algorithms as follows:*

- $\text{Setup}(1^k, n) \rightarrow (\text{PP}, \text{MK}, \text{RL}, \text{ST})$ The setup algorithm takes as input a security parameter k and a maximal number of users n . It outputs a public parameters PP , a master key MK , a revocation list RL (initially empty), and a state ST . (This is run by the key authority.)
- $\text{PriKeyGen}(\text{PP}, \text{MK}, id, \text{ST}) \rightarrow (\text{SK}_{id}, \text{ST})$ The private key generation algorithm takes as input the public parameters PP , the master key MK , an identity $id \in \mathcal{I}$, and the state ST . It outputs a private key SK_{id} and an updated state ST . (This is stateful and run by the key authority.)
- $\text{KeyUpd}(\text{PP}, \text{MK}, t, \text{RL}, \text{ST}) \rightarrow \text{KU}_t$ The key update algorithm takes as input the public parameters PP , the master key MK , a key update time $t \in \mathcal{T}$, the revocation list RL , and the state ST . It outputs a key update KU_t . (This is run by the key authority.)
- $\text{DecKeyGen}(\text{SK}_{id}, \text{KU}_t) \rightarrow \text{DK}_{id,t}$ The decryption key generation takes as input a private key SK_{id} and key update KU_t . It outputs a decryption key $\text{DK}_{id,t}$ or a special symbol \perp indicating that id was revoked. (This is run by the receiver.)
- $\text{Enc}(\text{PP}, id, t, M) \rightarrow \text{CT}_{id,t}$ The encryption algorithm takes as input the public parameters PP , an identity id , an encryption time t , and a message $M \in \mathcal{M}$. It outputs a ciphertext $\text{CT}_{id,t}$. (This is run by the sender. For simplicity and without loss of generality, we assume that id, t are efficiently computable from $\text{CT}_{id,t}$.)
- $\text{Dec}(\text{PP}, \text{DK}_{id,t}, \text{CT}_{id,t}) \rightarrow M$ The decryption algorithm takes as input the public parameters PP , a decryption key $\text{DK}_{id,t}$, and a ciphertext $\text{CT}_{id,t}$. It outputs a message M . (This is deterministic and run by the receiver.)
- $\text{KeyRev}(id, t, \text{RL}, \text{ST}) \rightarrow \text{RL}$ The key revocation algorithm takes as input an identity to be revoked id , a revocation time t , the revocation list RL , and the state ST . It outputs an updated revocation list RL . (This is stateful and run by the key authority.)

The consistency condition requires that for all $k \in \mathbb{N}$ and polynomials (in k) n , all PP and MK output by setup algorithm Setup , all $M \in \mathcal{M}, id \in \mathcal{I}, t \in \mathcal{T}$ and all possible valid states ST and revocation lists RL , we then have $\text{Dec}(\text{PP}, \text{DK}_{id,t}, \text{CT}_{id,t}) = M$ with probability 1 if identity id was not revoked before or at time t .

Next, we define the security of RIBE in the form of a security game played between an adversary and a challenger.

- **Setup:** It is run to generate some public parameters PP, a master key MK, a revocation list RL (initially empty), and a state ST. Then PP is given to \mathcal{A} .
- **Query:** \mathcal{A} may adaptively make a polynomial number of queries of the following oracles (which share state information):
 - The private key generation oracle $\text{PriKeyGen}(\cdot)$ takes as input an identity id and runs $\text{PriKeyGen}(\text{PP}, \text{MK}, id, \text{ST})$ to return a private key SK_{id} .
 - The key update generation oracle $\text{KeyUpd}(\cdot)$ takes as input time t and runs $\text{KeyUpd}(\text{PP}, \text{MK}, t, \text{RL}, \text{ST})$ to return key update KU_t .
 - The revocation oracle $\text{KeyRev}(\cdot, \cdot)$ takes as input an identity id and time t , and runs $\text{KeyRev}(id, t, \text{RL}, \text{ST})$ to update RL.
- **Challenge:** \mathcal{A} outputs the target ID-time pair (id^*, t^*) and two messages M_0, M_1 . The challenger flips a random bit d and returns the output of $\text{Enc}(\text{PP}, id^*, t^*, M_d)$ to \mathcal{A} . After that, the adversary may continue to make queries to the oracles as with in the Query phase.
- **Guess:** At the end of the game, the adversary outputs a bit d' , and succeeds if $d' = d$.

The following restrictions must always hold:

1. $M_0, M_1 \in \mathcal{M}$ and $|M_0| = |M_1|$.
2. $\text{KeyUpd}(\cdot)$ and $\text{KeyRev}(\cdot, \cdot)$ can be queried on a time which is greater than or equal to all the previously queried times, i.e., the adversary is allowed to query only in non-decreasing order of time. Also, the oracle $\text{KeyRev}(\cdot, \cdot)$ cannot be queried at time t if $\text{KeyUpd}(\cdot)$ was queried on t .
3. If $\text{PriKeyGen}(\cdot)$ was queried on identity id^* , then $\text{KeyRev}(\cdot, \cdot)$ must be queried on (id^*, t) for some $t \leq t^*$.

If the adversary's output d' equals to d , we set $\text{return} = 1$, otherwise $\text{return} = 0$. We define the adversary's advantage as

$$\text{Adv}_{\mathcal{A}}^{\text{RIBE}}(\lambda) := |\text{Pr}[\text{return} = 1] - \frac{1}{2}|.$$

An RIBE scheme is adaptive-ID secure if for all PPT adversaries \mathcal{A} the function $\text{Adv}_{\mathcal{A}}^{\text{RIBE}}(\lambda)$ is negligible.

3 Our Construction

3.1 Intuition

Our RIBE scheme is based on the Lewko and Waters IBE scheme [25]. In our scheme, however, the decryption key of each user has two components: one is fixed (long-term) and is associated with her identity id ; while the other is updated at the beginning of each time period (epoch) and corresponds to t . Particularly, the key component associated with id is

essentially a normal identity-based key (in the IBE setting) combined with a secret value³ that is associated with an accumulator, while the key component associated with t is a witness in the context of an accumulator.

All non-revoked users' identities are captured through an accumulator. At the beginning of each time period t , the key authority adds t to the accumulator and generates the corresponding witness (i.e., the values contained in the up-to-date accumulator are current time period and the identities of all legitimate users under this period). The key authority then broadcasts the updated accumulator and witness (with respect to t) to all users, who will then update their respective existing witnesses. We note here that the integrity of the accumulator is protected through a standard signature. To encrypt a message intended for id at time t , the encryptor makes use of the updated accumulator as part of the ciphertext. To decrypt a ciphertext, on the other hand, the decryptor must possess the correct identity-based key associated with id and updated witness corresponds to t . To revoke a user, the key authority simply removes the identity of the user from the accumulator. A revoked user would not be able to update his witness and therefore, would not be able to decrypt any ciphertext generated beyond the current epoch.

3.2 Construction

In addition to the Lewko and Waters IBE scheme [25], our RIBE construction makes use of two other building blocks: Camenisch et al.'s accumulator [13], and any standard public key signatures scheme PKS with three algorithms: the key generation algorithm PKSGen, the signing algorithm PKSSig and the verification algorithm PKSVer.

Let ϕ denote a one-to-one map from a string (id or t) to an index i . Our RIBE construction is described as follows:

- **Setup**($1^k, n$) \rightarrow (PP, MK, RL, ST $_{\emptyset}$) The setup algorithm first chooses a bilinear group G of order $N = p_1 p_2 p_3$ (with 3 distinct primes), random exponents $\alpha, \gamma \in \mathbb{Z}_N$, and random group elements $u, g, h \in G_{p_1}$. It also computes $e(g, g)^\alpha$, where $e : G \times G \rightarrow G_T$ is a bilinear map.

From the parameters $\langle N, G, G_T, e, g \rangle$, the algorithm performs the following steps:

1. run the PKSGen algorithm to generate a private-public key pair (sk, pk) ;
2. calculate $z = e(g, g)^{\gamma^{n+1}} \in G_T$ and $P_i = g^{(\gamma^i)} \in G_{p_1}$ for $i = 1, 2, \dots, n, n + 2, \dots, 2n$, where γ is randomly chosen from \mathbb{Z}_N ;
3. choose a random $\beta \in \mathbb{Z}_N$ and compute $g^\beta \in G_{p_1}$.

Let U be the bookkeeping information of all the elements that have ever been added into the accumulator (but not necessarily contained in the current accumulator), and at the point of system setup, $U = \emptyset$. The Setup algorithm then sets the accumulator $AC_{\emptyset} = 1$ and state $ST_{\emptyset} = \{U, P_1, \dots, P_n, P_{n+2}, \dots, P_{2n}\}$. The revocation list RL is initially empty.

The public parameters PP are $\langle N, u, g, h, g^\beta, e(g, g)^\alpha, z, pk, AC_{\emptyset}, P_1, \dots, P_n, P_{n+2}, \dots, P_{2n} \rangle$. The master secret key MK is $\langle \alpha, \beta, \gamma, sk \rangle$ and a generator of G_{p_3} .

³As described in Section 1.2, this is needed to circumvent a possible collusion attack.

- $\text{PriKeyGen}(\text{PP}, \text{MK}, id, \text{ST}_U) \rightarrow (\text{SK}_{id}, \text{ST}_{U \cup \{i\}})$ Let V denotes the bookkeeping information of the values that have *currently* been accumulated (so V is a subset of U). Given $i = \phi(id) \in [n]$, the private key generation algorithm performs the following steps:

1. compute $w_i = \prod_{j \in V, j \neq i} P_{n+1-j+i}$;
2. update the accumulator and state such that

$$\begin{aligned} \text{AC}_{V \cup \{i\}} &= \text{AC}_V \cdot P_{n+1-i} \text{ and} \\ \text{ST}_{U \cup \{i\}} &= \{U \cup \{i\}, P_1, \dots, P_n, P_{n+2}, \dots, P_{2n}\}. \end{aligned}$$

The PriKeyGen algorithm then chooses a random $r \in \mathbb{Z}_N$, and random elements $R_3, R'_3 \in G_{p_3}$. The private key SK_{id} is then:

$$\langle K_1 = g^r R_3, K_2 = g^{\alpha(u^{id}h)^r P_i^\beta R'_3}, K_3 = w_i \rangle.$$

The PriKeyGen algorithm also prepares a set V_w , which denotes the values contained in the accumulator when a witness w_i was created (so V_w is fixed for each user and it is also a subset of U). This set V_w is given to the user along with his private key SK_{id} . (We give a simple example in Appendix A illustrating how sets V and V_w are derived and updated.)

- $\text{KeyUpd}(\text{PP}, \text{MK}, t, \text{RL}, \text{ST}_U) \rightarrow \text{KU}_t$ At the start of each new time period t , the key update algorithm first updates the accumulator by performing the following steps:

1. remove $l' = \phi(t')$ associated with the just expired time period t' from V ;
2. remove all $i = \phi(id)$ that corresponds to t' in RL from V ;
3. update the accumulator, that is $\text{AC}_V = \prod_{i' \in V} P_{n+1-i'}$ for all i' in the updated V .

The KeyUpd algorithm then adds the new time period $l = \phi(t) \in [n]$ following the same steps as before (in PriKeyGen) to obtain the latest accumulator $\text{AC}_{V \cup \{l\}}$. It then generates a signature σ_l on $\text{AC}_{V \cup \{l\}}$. The algorithm also prepares a set ΔV , which contains a list of recently joined and revoked users' identities within the last (just expired) epoch. Then the KeyUpd algorithm broadcasts $\text{KU}_t = \langle \text{AC}_{V \cup \{l\}}, \sigma_l, w_l \rangle$, together with the set ΔV , to all users.

- $\text{DecKeyGen}(\text{SK}_{id}, \text{KU}_t) \rightarrow \text{DK}_{id,t}$ The decryption key generation algorithm first checks if:

1. $i = \phi(id), l = \phi(t) \in V$;
2. σ_l is a valid signature associated with AC_V using the PKSVer algorithm and pk ;
3. $e(P_l, \text{AC}_V) / e(g, w_l) = z$ to ensure the correctness of AC_V .

We set a Boolean flag denoted by DecKeyChk to 0 if *any* of the above three checks fails. If all the three conditions are satisfied, we set $\text{DecKeyChk} = 1$. If $\text{DecKeyChk} = 0$, then the DecKeyGen algorithm outputs a special symbol \perp . Otherwise, DecKeyGen replaces the existing accumulator with an up-to-date one. It then updates the witness and computes the decryption key as follows:

1. if $i \in V$ and $V \cup V_w \subset U$, compute

$$w'_i = w_i \cdot \frac{\prod_{j \in V \setminus V_w} P_{n+1-j+i}}{\prod_{j \in V_w \setminus V} P_{n+1-j+i}};$$

2. otherwise, output \perp .

Set the decryption key $\text{DK}_{id,t}$ to be

$$\langle K_1 = g^r R_3, K_2 = g^\alpha (u^{id} h)^r P_i^\beta R'_3, K_3 = w'_i \rangle.$$

- $\text{Enc}(\text{PP}, M, id, \text{AC}_V) \rightarrow \text{CT}_{id,t}$ Given a message M and an up-to-date accumulator AC_V containing current time t , the encryption algorithm chooses $s \in \mathbb{Z}_N$ randomly, and set the ciphertext $\text{CT}_{id,t}$ to be

$$C = M \frac{e(g, g)^{\alpha s}}{z^s}, C_0 = g^s, C_1 = (u^{id} h)^s, C_2 = (g^\beta \text{AC}_V)^s.$$

- $\text{Dec}(\text{PP}, \text{DK}_{id,t}, \text{CT}_{id,t}) \rightarrow M$ The decryption algorithm computes

$$\frac{e(C_0, K_2 K_3)}{e(C_1, K_1) e(P_{\phi(id)}, C_2)} = \frac{e(g, g)^{\alpha s}}{z^s}.$$

The message M can be recovered by dividing C by the computed term.

- $\text{KeyRev}(id, t, \text{RL}, \text{ST}_U) \rightarrow \text{RL}$ The key revocation algorithm adds (id, t) to the revocation list RL if $i = \phi(id) \in \text{ST}_U$.

CORRECTNESS. We now verify that the decryption algorithm works correctly. First we notice that a correct accumulator is always in the form of $\text{AC}_V = \prod_{j \in V} P_{n+1-j}$, and the witness w_i for each $i \in V$ always has a value $w_i = \prod_{j \in V, j \neq i} P_{n+1-j+i}$. Hence, the following equation always holds:

$$\frac{e(P_i, \text{AC}_V)}{e(g, w_i)} = \frac{e(g, g)^{\sum_{j \in V} (\gamma^{n+1-j+i})}}{e(g, g)^{\sum_{j \in V, j \neq i} (\gamma^{n+1-j+i})}} = e(g, g)^{(\gamma^{n+1})} = z.$$

Thus we have

$$\begin{aligned} & \frac{e(C_0, K_2 K_3)}{e(C_1, K_1) e(P_{\phi(id)}, C_2)} \\ &= \frac{e(g^s, g^\alpha (u^{id} h)^r P_{\phi(id)}^\beta R'_3 \cdot w_{\phi(id)})}{e((u^{id} h)^s, g^r R_3) e(P_{\phi(id)}, (g^\beta \text{AC}_V)^s)} \\ &= \frac{e(g^s, g^\alpha (u^{id} h)^r R'_3)}{e((u^{id} h)^s, g^r R_3)} \cdot \frac{e(g^s, P_{\phi(id)}^\beta w_{\phi(id)})}{e(P_{\phi(id)}, (g^\beta \text{AC}_V)^s)} \\ &= \frac{e(g, g)^{\alpha s} e(g, u^{id} h)^{rs}}{e(u^{id} h, g)^{rs}} \cdot \frac{e(g, P_{\phi(id)}^\beta)^s e(g, w_{\phi(id)})^s}{e(P_{\phi(id)}, g)^{\beta s} e(P_{\phi(id)}, \text{AC}_V)^s} \\ &= \frac{e(g, g)^{\alpha s}}{z^s}. \end{aligned}$$

REMARK. In comparison with the Lewko and Waters IBE scheme, our identity-based private key component K_2 has an additional secret value P_i^β . Moreover, our ciphertexts are different in two aspects: the blinding factor of our ciphertext component C has an additional term z^{-s} , and we have an additional ciphertext component C_2 .

It is also worth stressing again that our scheme enforces key revocation through decryption (at the recipient), that is, a ciphertext recipient can decrypt properly only if he uses an updated decryption key. An encryptor may or may not know the set V (which is associated with a revocation list),⁴ and hence, may not always know if a target recipient has been revoked.

3.3 Security Analysis

3.3.1 Overview

In our security analysis, we consider the following two types of adversaries:

- Type I adversaries that never make a private key query on the target identity id^* at any time throughout the game.
- Type II adversaries that are allowed to make a private key query on the target identity id^* at some point of the game, provided that the queried identity must subsequently be revoked before the challenge time t^* .

We then prove the security of our RIBE scheme by adopting the dual system encryption technique by Waters [35]. In our proofs, private keys and ciphertexts take two forms: normal or semi-functional. A normal private key could decrypt a ciphertext, which in turn, is either normal or semi-functional; while a semi-functional private key can only decrypt a normal ciphertext. When using a semi-functional key to decrypt a semi-functional ciphertext, the decryption will fail. We then use a hybrid argument through a sequence of games to prove the security of our scheme. We first change the challenge ciphertext to semi-functional, then gradually change the private keys into semi-functional one by one. At the very last step, we change the semi-functional ciphertext into an encryption of a random message, in which the adversary has no advantage at all. Particularly, we prove that neither Type I nor Type II adversary learns any useful information about the chosen message from the challenge ciphertext, even when they are provided some information on the associated blinding factor.

3.3.2 Security proofs

We first define the semi-functional ciphertexts and semi-functional keys that will be used in our proofs.

Semi-functional Ciphertext Let g_2 denote a generator of the subgroup G_{p_2} . A semi-functional ciphertext is then created as follows: first, generate a normal ciphertext C', C_0, C_1, C_2 using the encryption algorithm; then choose random exponents $x, z_c, z_d \in \mathbb{Z}_N$ and set C to be C' , C_0 to be $C'_0 g_2^x$, C_1 to be $C'_1 g_2^{xz_c}$, and C_2 to be $C'_2 g_2^{z_d}$. The resulting tuple (C, C_0, C_1, C_2) is a semi-functional ciphertext.

⁴Recall that in IBE, anyone can encrypt to an identity using the appropriate public parameters (even without having a decryption key).

Semi-functional Key We create semi-functional key as follows: first generate a set of normal key K'_1, K'_2, K'_3 with the key generation algorithm; then choose random exponents $\delta, z_k \in \mathbb{Z}_N$ and set K_1 to be $K'_1 g_2^\delta$, K_2 to be $K'_2 g_2^{\delta z_k}$, and $K_3 = K'_3$ is kept unchanged. The resulting tuple (K_1, K_2, K_3) is a semi-functional key.

If a semi-functional key is used to decrypt a semi-functional ciphertext, an additional factor $e(g_2, g_2)^{x\delta(z_k - z_c)}$ will hinder the decryption. If $z_c = z_k$, the decryption will still work, in this case we call the key *nominally* semi-functional: it is in the semi-functional form but still allows decryption.

Our proofs rely on Assumptions 1, 2, 3, and 4 defined in Section 2.2, and a hybrid argument through a sequence of games: **Game_{real}**, **Game_{restricted}**, **Game_k**, and **Game_{final}** as defined in [25], for $0 \leq k \leq q$ where q denotes the number of key queries the attacker makes. However, we split **Game_{final}** into two games, corresponding to two types of adversaries, as follows:

Game_{final_1} is the same as $Game_q$ except that the challenge ciphertext is a semi-functional encryption of a random message, embedded with an instance of the hard problem defined by Assumption 3.

Game_{final_2} is the same as $Game_{final_1}$, except that here, we embed a instance of the hard problem defined by Assumption 4.

At the outset of the game, the simulator \mathcal{B} flips a coin $coin \xleftarrow{R} \{0, 1\}$ as the guess for the type of adversary it will face: 0 for Type I and 1 for Type II. The game then proceed from $Game_{real}$ to $Game_q$. After $Game_q$, if \mathcal{B} faces a Type I adversary, it will proceed to $Game_{final_1}$, otherwise, it will proceed to $Game_{final_2}$. In what follows, we prove that these games are indistinguishable from the attacker's viewpoint.

Lemma 1. *Suppose there exists an algorithm \mathcal{A} such that $Game_{real}Adv_{\mathcal{A}} - Game_{restricted}Adv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm \mathcal{B} with advantage $\frac{\epsilon}{2}$ in breaking Assumption 2.*

Proof. Given $g, X_1 X_2, X_3, Y_2 Y_3$, \mathcal{B} can simulate $Game_{real}$ with \mathcal{A} . With probability ϵ , \mathcal{A} produces identities id and id^* such that $id \neq id^*$ modulo N and p_2 divides $id - id^*$. If \mathcal{A} fails to do this, \mathcal{B} will simply guess randomly. \mathcal{B} uses these identities to produce a nontrivial factor of N by computing $a = \gcd(id - id^*, N)$. We let $b = \frac{N}{a}$. Consider the following three cases:

1. one of a, b is p_1 , and the other is $p_2 p_3$
2. one of a, b is p_2 , and the other is $p_1 p_3$
3. one of a, b is p_3 , and the other is $p_1 p_2$.

\mathcal{B} can determine if case 1 has occurred by testing if either of $(Y_2 Y_3)^a$ or $(Y_2 Y_3)^b$ is the identity element. If this happens, we will suppose that $a = p_1$ and $b = p_2 p_3$ without loss of generality. \mathcal{B} can then learn whether T has a G_{p_2} component or not by testing if $e(T^a, X_1 X_2)$ is the identity element. If it is not, then T has a G_{p_2} component.

\mathcal{B} can determine if case 2 has occurred by testing if either of $(X_1 X_2)^a$ or $(X_1 X_2)^b$ is the identity element. Assuming that \mathcal{B} has already ruled out case 1 and neither of these is the identity element, then case 2 has occurred. \mathcal{B} can learn which of a, b is equal to $p_1 p_3$ by testing which of g^a, g^b is the identity. We assume without loss of generality that $a = p_2$ and

$b = p_1 p_3$. Then \mathcal{B} can learn whether T has a G_{p_2} component or not by testing if T^b is the identity element. If it is not, then T has a G_{p_2} component.

\mathcal{B} can determine that case 3 has occurred when the tests for case 1 and 2 fail. It can learn which of a, b is equal to p_3 by testing which of X_3^a, X_3^b is the identity. We assume without loss of generality that $a = p_3$. \mathcal{B} can learn whether T has a G_{p_2} component or not by testing whether $e(T^a, Y_2 Y_3)$ is the identity. If it is not, then T has a G_{p_2} component. \square

Lemma 2. *Suppose there exists an algorithm \mathcal{A} such that $\text{Game}_{\text{restrictedAdv}_{\mathcal{A}}} - \text{Game}_0 \text{Adv}_{\mathcal{A}} = \epsilon$. Then we can build an algorithm \mathcal{B} with advantage ϵ in breaking Assumption 1.*

Proof. \mathcal{B} receives g, X_3, T and simulates either $\text{Game}_{\text{restricted}}$ or Game_0 with \mathcal{A} . \mathcal{B} sets the public parameters as follows: it chooses random exponents $a, b, \alpha, \beta, \gamma \in \mathbb{Z}_N$ and sets $g = g, u = g^a, h = g^b$; computes $z = e(g, g)^{\gamma^{n+1}}$, $P_i = g^{\gamma^i}$ for $i \in [2n] \setminus \{n+1\}$, and sets $\text{AC}_\emptyset = 1$; and generates a public-private key pair (sk, pk) using PKSGen . \mathcal{B} then forwards the public parameters $\langle N, u, g, h, g^\beta, e(g, g)^\alpha, z, pk, \text{AC}_\emptyset, P_i \rangle$ to \mathcal{A} .

Whenever \mathcal{B} is asked to provide a private key for an identity id_i , it chooses random exponents $r_i, t_i, y_i \in \mathbb{Z}_N$. Using the set V which contains the current accumulated values, \mathcal{B} sets $K_1 = g^{r_i} X_3^{t_i}$, $K_2 = g^\alpha (u^{id_i} h)^{r_i} P_i^\beta X_3^{y_i}$, and $K_3 = \prod_{j \in V, j \neq i} P_{n+1-j+i}$.

If \mathcal{B} is asked for a key update, it calculates the new accumulator (with the current sets V and U) as $\text{AC} = \prod_{j \in V} P_{n+1-j}$ and creates a signature σ_j for the accumulator using key sk . \mathcal{B} then publishes $\langle \text{AC}_V, \sigma_j, w_j \rangle$.

At the challenge phase, \mathcal{A} sends \mathcal{B} two messages, M_0 and M_1 , along with a challenge identity-time pair (id^*, t^*) . \mathcal{B} first checks the Boolean flag DecKeyChk using an up-to-date accumulator associated with t^* and the given target identity-time pair. If $\text{DecKeyChk} = 0$, then \mathcal{B} just guesses randomly which group the value T belongs to; otherwise \mathcal{B} chooses $d \in \{0, 1\}$ randomly and sets the challenge ciphertext as:

$$C = M_d \frac{e(g, T)^\alpha}{e(g, T)^{\gamma^{n+1}}}, \quad C_0 = T, \quad C_1 = T^{a \cdot id^* + b}, \quad C_2 = T^{\beta + \sum_{j \in V} \gamma^{n+1-j}}.$$

First note this implicitly sets g^s to be equal to the G_{p_1} part of T , and hence the value z^s is computed as $e(g, T)^{\gamma^{n+1}}$. We note that a normal C_2 component is in the form of $(g^\beta \text{AC}_V)^s = (g^\beta \prod_{j \in V} g^{\gamma^{n+1-j}})^s = (g^\beta g^{\sum_{j \in V} \gamma^{n+1-j}})^s = (g^s)^{\beta + \sum_{j \in V} \gamma^{n+1-j}}$. Thus, if $T \in G_{p_1 p_2}$, then the ciphertext is semi-functional with $z_c = a \cdot id^* + b$ and $z_d = \beta + \sum_{j \in V} \gamma^{n+1-j}$. Here the values z_c and z_d modulo p_2 are independent of the values of a, b, β, γ modulo p_1 , so they are properly distributed. If $T \in G_{p_1}$, this is a normal ciphertext. Hence \mathcal{B} can use the output of \mathcal{A} to distinguish T . \square

Lemma 3. *Suppose there exists an algorithm \mathcal{A} such that $\text{Game}_{k-1} \text{Adv}_{\mathcal{A}} - \text{Game}_k \text{Adv}_{\mathcal{A}} = \epsilon$. Then we can build an algorithm \mathcal{B} with advantage ϵ in breaking Assumption 2.*

Proof. \mathcal{B} first receives $g, X_1 X_2, X_3, Y_2 Y_3, T$. It sets the public parameters as with those of Lemma 2.

If \mathcal{A} requests for the i^{th} key for id_i where $i < k$, \mathcal{B} creates a semi-functional key by choosing random exponents $r_i, t_i, y_i \in \mathbb{Z}_N$ and setting $K_1 = g^{r_i} (Y_2 Y_3)^{t_i}$, $K_2 = g^\alpha (u^{id_i} h)^{r_i} P_i^\beta (Y_2 Y_3)^{y_i}$, $K_3 = \prod_{j \in V, j \neq i} P_{n+1-j+i}$. Note that this is a properly distributed semi-functional key with $g_2^\delta = Y_2^{t_i}$. As with Lemma 2, the values of t_i and y_i modulo p_2 and modulo p_3 are uncorrelated by the Chinese Remainder Theorem. To handle private key queries for $i > k$, \mathcal{B}

generates normal keys using random exponents $r_i, t_i, y_i \in \mathbb{Z}_N$ and setting: $K_1 = g^{r_i} X_3^{t_i}$, $K_2 = g^\alpha (u^{id_i} h)^{r_i} P_i^\beta X_3^{y_i}$, $K_3 = \prod_{j \in V, j \neq i} P_{n+1-j+i}$. To answer the k^{th} private key query, \mathcal{B} simply sets $z_k = a \cdot id_k + b$, chooses a random exponent $t_k \in \mathbb{Z}_N$, and sets: $K_1 = T$, $K_2 = g^\alpha T^{z_k} P_i^\beta X_3^{t_k}$, $K_3 = \prod_{j \in V, j \neq i} P_{n+1-j+i}$.

At the challenge phase, \mathcal{A} sends \mathcal{B} two messages, M_0 and M_1 , together a challenge identity-time pair (id^*, t^*) . \mathcal{B} first uses the current accumulator associated with t^* and the challenge pair to check the value of DecKeyChk. If DecKeyChk = 0 then \mathcal{B} just guesses randomly, otherwise \mathcal{B} chooses $d \in \{0, 1\}$ randomly and sets the ciphertext as:

$$C = M_d \frac{e(g, X_1 X_2)^\alpha}{e(g, X_1 X_2)^{\gamma^{n+1}}}, C_0 = X_1 X_2, C_1 = (X_1 X_2)^{a \cdot id^* + b}, C_2 = (X_1 X_2)^{\beta + \sum_{j \in V} \gamma^{n+1-j}}.$$

In this lemma, the argument that \mathcal{B} could only make a nominally semi-functional key k is similar to that of [25, Lemma 7]. If $T \in G_{p_1 p_3}$, then \mathcal{B} has properly simulated $Game_{k-1}$. If $T \in G$, then \mathcal{B} has properly simulated $Game_k$. From the output of \mathcal{A} , \mathcal{B} could distinguish the possibilities for T . \square

Lemma 4. *Suppose there exists an algorithm \mathcal{A} such that $Game_q Adv_{\mathcal{A}} - Game_{final-1} Adv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm \mathcal{B} with advantage ϵ in breaking Assumption 3.*

Proof. \mathcal{B} first receives $g, g^\alpha, X_3, g^s Y_2, Z_2, T$. It sets public parameters as with those of Lemma 2 except that it sets $e(g, g)^\alpha$ as $e(g^\alpha X_2, g)$ instead of choosing α randomly. Additionally, \mathcal{B} computes $z^s = e(g, g^s Y_2)^{\gamma^{n+1}}$ and passes it to \mathcal{A} .

When \mathcal{A} requests for the private key for id_i , \mathcal{B} generates a semi-functional key, chooses random exponents $c_i, r_i, t_i, x_i, y_i \in \mathbb{Z}_N$, and sets: $K_1 = g^{r_i} Z_2^{x_i} X_3^{t_i}$, $K_2 = g^\alpha X_2 (u^{id_i} h)^{r_i} P_i^\beta Z_2^{c_i} X_3^{y_i}$, $K_3 = \prod_{j \in V, j \neq i} P_{n+1-j+i}$.

At the challenge phase, \mathcal{A} sends \mathcal{B} two messages, M_0 and M_1 , together with a challenge identity-time pair (id^*, t^*) . \mathcal{B} first checks the value of DecKeyChk. If DecKeyChk = 0, then \mathcal{B} just guesses randomly, otherwise \mathcal{B} chooses $d \in \{0, 1\}$ randomly and sets the ciphertext as:

$$C = M_d \frac{T}{e(g, g^s Y_2)^{\gamma^{n+1}}}, C_0 = g^s Y_2, C_1 = (g^s Y_2)^{a \cdot id^* + b}, C_2 = (g^s Y_2)^{\beta + \sum_{j \in V} \gamma^{n+1-j}}.$$

This implicitly sets $z_c = a \cdot id^* + b$. Note that although $u = g^a, h = g^b$ and $z_c = a \cdot id^* + b$, since u, h are elements of G_{p_1} (meaning modulo p_1) and z_c is modulo p_2 , their values are independent from each other.

Given the value $z^s = e(g, g^s Y_2)^{\gamma^{n+1}}$, the remaining task of \mathcal{A} is simply to distinguish the possibilities of T . If $T = e(g, g)^{\alpha s}$, then this is a properly generated semi-functional ciphertext with message M_d . If T is a random element of G_T , then the ciphertext is a semi-functional one with a random message. Thus \mathcal{B} could use the output of \mathcal{A} to distinguish between the possibilities of T . \square

Lemma 5. *Suppose there exists an algorithm \mathcal{A} such that $Game_q Adv_{\mathcal{A}} - Game_{final-2} Adv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm \mathcal{B} with advantage ϵ in breaking the OBDHE assumption.*

Proof. \mathcal{B} first receives an instance of the OBDHE problem:

$$g, f, \{P_i\}_{i \in \{1, \dots, 2n\} \setminus \{n+1\}}, v, T$$

where $f = g^s$ for some random $s \in \mathbb{Z}_N$, $P_i = g^{\gamma^i}$, $v = g^\beta$, together with oracle access to $\mathcal{O}_{g,e}^{DH}$ as specified by the assumption. \mathcal{B} is required to distinguish if $T = e(g^{\gamma^{n+1}}, f)$ or T is a random element of G_T . \mathcal{B} then chooses random exponents $a, b, \alpha \in \mathbb{Z}_N$ and sets $g = g, u = g^a, h = g^b$. It can compute $z = e(g, g)^{\gamma^{n+1}}$ using a pair (P_{n+1-i}, P_i) for any i and calculate $e(P_{n+1-i}, P_i)$. It sets $\text{AC}_\emptyset = 1$. Moreover, \mathcal{B} generates a public-private key pair (sk, pk) using PKSGen and forwards the public parameters $\langle N, g, u, h, g^\beta, e(g, g)^\alpha, z, pk, \text{AC}_\emptyset, P_i \rangle$ to \mathcal{A} . Additionally, \mathcal{B} computes and returns $e(g, g)^{\alpha s} = e(g, f)^\alpha$ to \mathcal{A} .

When \mathcal{A} requests for the private key for id_i , \mathcal{B} generates a semi-functional key by choosing random exponents $c_i, r_i, t_i, x_i, y_i \in \mathbb{Z}_N$, random $Z_2 \in G_{p_2}$, $X_3 \in G_{p_3}$. \mathcal{B} first computes $K_1 = g^{r_i} Z_2^{x_i} X_3^{t_i}$ and $K_3 = \prod_{j \in V, j \neq i} P_{n+1-j+i}$. To provide the key component K_2 for id_i , \mathcal{B} queries the oracle $\mathcal{O}_{g,e}^{DH}(P_i, v)$. Since $v = g^\beta$, the oracle's output equals to P_i^β . Then \mathcal{B} sets $K_2 = g^\alpha (u^{id_i} h)^{r_i} P_i^\beta Z_2^{c_i} X_3^{y_i}$.

At the challenge phase, \mathcal{A} sends \mathcal{B} two messages, M_0 and M_1 , together with a challenge identity-time pair (id^*, t^*) . \mathcal{B} first checks the value of DecKeyChk . If $\text{DecKeyChk} = 0$ then \mathcal{B} just guesses randomly, otherwise \mathcal{B} chooses $d \in \{0, 1\}$ and a group G_{p_2} element Y_2 randomly, and sets parts of the ciphertext as:

$$C = M_d \frac{e(g, f)^\alpha}{T}, C_0 = fY_2, C_1 = (fY_2)^{a \cdot id^* + b}.$$

To transform the ciphertext component C_2 into semi-functional, \mathcal{B} computes $\omega = v \prod_{j \in V} P_{n+1-j}$ and makes an oracle query $\mathcal{O}_{g,e}^{DH}(\omega, f)$. Upon receiving the output $h' = (g^\beta \prod_{j \in V} P_{n+1-j})^s = (g^\beta \text{AC}_V)^s$, \mathcal{B} chooses a random $u_i \in \mathbb{Z}_N$ and sets $C_2 = h' Y_2^{u_i}$.

Given the value $e(g, g)^{\alpha s} = e(g, f)^\alpha$, the remaining task of \mathcal{A} is simply to distinguish the possibilities of T . If $T = e(g^{\gamma^{n+1}}, h) = e(g, g)^{\gamma^{n+1} s} = z^s$, then this is a properly generated semi-functional ciphertext with message M_d . If T is a random element of G_T , then the ciphertext is a semi-functional one with a random message. Thus \mathcal{B} could use the output of \mathcal{A} to distinguish between the possibilities of T . \square

Theorem 1. *If Assumptions 1, 2, 3, and 4 hold, then our RIBE scheme is secure.*

The proof for Theorem 1 follows from Lemmas 1 to 5.

3.4 Efficiency

We now compare the efficiency of our construction against existing pairing-based RIBE schemes. This is illustrated in Table 1. We let \tilde{n} denote the number of users in the system, \hat{n} denote the size of identity space representing \tilde{n} many users, r denote the number of revoked users, and $r' = |\Delta V|$, where ΔV is as defined in Section 3.2. Also, we let PP denote public parameters, DK denote decryption key, CT denote ciphertext, KUp denote key update, Dec denote decryption, SM denote security model, and Group denote the underlying bilinear group. The sizes for PP, DK, CT, and KUp are measured in the number of group elements; Dec is measured as the number of pairing operations; SM is either selective or adaptive and Group is either prime or composite.

From Table 1, we see that all adaptively secure RIBE schemes, including ours, have comparable DK and CT sizes, and the computational overhead of Dec. However, our scheme has a clear advantage of having constant size KUp.

Table 1: A comparison between existing and our RIBE schemes.

	PP size	DK size	CT size	KUp size	Dec	SM	Group
BGK [7]	$O(1)$	4	4	$O(\log(\hat{n}))$	4	select.	prime
LV [27]	$O(\hat{n})$	4	5	$O(\log(\hat{n}))$	3	adapt.	prime
SE [33]	$O(\hat{n})$	3	4	$O(\log(\hat{n}))$	3	adapt.	prime
Ours	$O(\tilde{n})$	3	4	$O(1)$	3	adapt.	composite

We note that during key update, the key authority of all the above considered schemes also broadcasts some auxiliary information with respect to non-revoked/revoked user identities. As shown in the analysis of the key update algorithm in [7], the binary tree approach is most advantageous when $r \leq \frac{\tilde{n}}{2}$, in which case the complexity of key update is $O(r \log(\frac{\tilde{n}}{r}))$. (For the case of $r > \frac{\tilde{n}}{2}$, we simply assume that the scheme can be “reset” to retain the efficiency of key update.) By considering only the case of $r \leq \frac{\tilde{n}}{2}$, we have $\frac{\tilde{n}}{r} \geq 2$ and $\log(\frac{\tilde{n}}{r}) \geq 1$, and thus we have $O(r \log(\frac{\tilde{n}}{r})) \geq O(r)$. In the BGK, LV, SE schemes, therefore, the auxiliary information required during key update has complexity of $O(r)$, while ours has complexity of $O(r')$. In reality, we have $r' < r$, or even $r' \ll r$. Consider a concrete example by letting $r' = |\Delta V| = 100$ and assuming a user identity is of 32-bits, our bookkeeping information during key update consumes only 400 bytes.

4 Extension

4.1 Supporting More Than n Users

Our scheme presented in Section 3.2 supports up to only n users. However, as shown by Phan et al. [30] in their dynamic broadcast encryption scheme, our scheme similarly can handle polynomially many more than n users (but still bounded) and remains secure under a generalization [16, 30] of the decisional bilinear Diffie-Hellman Exponent (BDHE) assumption [8] defined as follow:

Assumption 5. *Given the input $g^{P(x_1)}, \dots, g^{P(x_n)} \in G_{p_1}$ and $e(g, g)^{Q(x_1)}, \dots, e(g, g)^{Q(x_n)} \in G_T$ for random choices of $x_1, \dots, x_n \in \mathbb{Z}_N$, the generalized decisional BDHE (GBDHE) assumption says that it is hard to decide between $e(g, g)^{f(x_1)}, \dots, e(g, g)^{f(x_n)} \in G_T$ and a random $T' \in G_T$ if polynomial f is independent of polynomials P and Q .*

Recall that the decisional BDHE assumption, typically parameterized by n and denoted by n -BDHE, is an instance of the GBDHE assumption defined as follows:

Assumption 6. *Given the input $g, h, \{g_k = g^{\alpha^k}\}_{k \in \{1, \dots, 2n\} \setminus \{n+1\}}$ for random $g, h \in G_{p_1}$ and $\alpha \in \mathbb{Z}_N$, it is hard to decide between $e(g_{n+1}, h)$ and a random $T' \in G_T$.*

Hence, let m be the new upper bound of the number of supported users, the security of our RIBE supporting more than n users (where the next user is numbered $n + 2$) can be proved by considering the following assumption:

Assumption 7. *Given the input h and $\{g_k = g^{\alpha^k}\}$ for $k \in \{n+1-m, \dots, n+1+m\} \setminus \{n+1\}$ for random $g, h \in G_{p_1}$ and $\alpha \in \mathbb{Z}_N$, it is hard to decide between $e(g_{n+1}, h)$ and a random $T' \in G_T$.*

Note that Assumption 7 is equivalent to the following assumption:

Assumption 8. *Given the input h and $\{g_k = g^{\alpha^k}\}$ for $k \in \{1, \dots, 2m\} \setminus \{m\}$ for random $g, h \in G_{p_1}$ and $\alpha \in \mathbb{Z}_N$, it is hard to decide between $e(g_m, h)$ and a random $T' \in G_T$.*

We have, therefore, $m \geq n+2$ being the new upper bound, since Assumption 8 is comparable to the m -BDHE assumption, which in fact, is also an instance of the GBDHE assumption.

We stress that our system setup and ciphertext size are independent of the upper bound of the number of supported users. Moreover, when $m \geq n+2$, the private keys of new users can be generated without requiring any update to other existing users' private keys. We assume that the additional new public parameters g_k (for $k \geq n+2$) can be distributed (one-off) to all users as part of key update information broadcast by the key authority or system update imposed by the system owner.

4.2 Forward-secure Decryption Keys

Our security model, as with that of Boldyreva et al.'s [7], considers exposure of only long-term private keys, but not decryption keys. Given a private key SK_{id} for a user with identity id , an adversary attacking our RIBE scheme can easily compute the corresponding decryption key $\text{DK}_{id,t}$ associated with time t from SK_{id} and key update KU_t . If a private key query is made on a challenge identity id^* , we simply revoke id^* such that DK_{id^*,t^*} for challenge time t^* is no longer useful to the adversary. However, if the adversary is also allowed access to the decryption key of any user, such as that in a security model recently considered by Seo and Emura [33], we then require the decryption keys to be forward-secure. Otherwise, given a decryption key $\text{DK}_{id^*,t}$ for $t \neq t^*$ and without making a private key query on the challenge id^* (implying id^* has still not been revoked at t^*), the adversary may be able to deduce the decryption key DK_{id^*,t^*} from $\text{DK}_{id^*,t}$ and KU_{t^*} . Consequently, it will be trivial for the adversary to win the security game.

Recall that in our current RIBE scheme, a decryption key comprises

$$\langle K_1 = g^r R_3, K_2 = g^\alpha (u^{id} h)^r P_{\phi(id)}^\beta R'_3, K_3 = w_{\phi(id)} \rangle$$

where K_1, K_2 are fixed and K_3 is periodically updated. However, since K_3 can, in principle, be computed by anyone who has access to the public parameters, exposure of a decryption key $\text{DK}_{id^*,t}$ for time $t \neq t^*$ can also lead to exposure of a decryption key $\text{DK}_{id^*,t'}$ for any t' , including $t' = t^*$. Clearly, the adversary can then trivially learn the challenge message.

Nevertheless, our RIBE scheme can naturally be extended to achieve forward-secure decryption keys using a 2-level Lewko and Waters HIBE scheme [25]. Particularly, we let level 1 keys be users' long-term private keys (associated with identities), and let level 2 keys be decryption keys (associated with times). For each time period, a user "delegates" a new, fully randomized decryption key with her long-term private key. As shown in [33], re-randomization of decryption keys is sufficient to achieve the forward-secure property. In what follows, we sketch a modified version of our RIBE scheme that achieves forward-secure decryption keys.

- $\text{Setup}(1^k, n) \rightarrow (\text{PP}, \text{MK}, \text{RL}, \text{ST}_\emptyset)$ As before. However, we require two random group elements $u_1, u_2 \in G_{p_1}$, instead of just u .
- $\text{PriKeyGen}(\text{PP}, \text{MK}, id, \text{ST}_U) \rightarrow (\text{SK}_{id}, \text{ST}_{U \cup \{i\}})$ As before. However, the private key SK_{id} now has an additional E_2 component:

$$K_1 = g^r R_3, K_2 = g^\alpha (u_1^{id} h)^r P_{\phi(id)}^\beta R'_3, K_3 = w_{\phi(id)}, E_2 = u_2^r R'_3$$

- $\text{KeyUpd}(\text{PP}, \text{MK}, t, \text{RL}, \text{ST}_U) \rightarrow \text{KU}_t$ As before.
- $\text{DecKeyGen}(\text{SK}_{id}, \text{KU}_t) \rightarrow \text{DK}_{id,t}$ As before. In addition, the algorithm chooses a random $r' \in \mathbb{Z}_N$ and random elements \hat{R}_3, \hat{R}'_3 of G_{p_3} . The decryption key is then set to be:

$$\begin{aligned} K'_1 &= K_1 g^{r'} \hat{R}_3 = g^{r+r'} \tilde{R}_3, \\ K'_2 &= K_2 (u_1^{id} h)^{r'} (E_2)^t u_2^{r't} \hat{R}'_3 = g^\alpha (u_1^{id} u_2^t h)^{r+r'} P_{\phi(id)}^\beta \tilde{R}'_3, \\ K_3 &= w'_{\phi(id)}. \end{aligned}$$

- $\text{Enc}(\text{PP}, M, id, \text{AC}_V) \rightarrow \text{CT}_{id,t}$ As before, except a small modification to C_1 :

$$C = M \frac{e(g, g)^{\alpha s}}{z^s}, \quad C_0 = g^s, \quad C_1 = (u_1^{id} u_2^t h)^s, \quad C_2 = (g^\beta \text{AC}_V)^s.$$

- $\text{Dec}(\text{PP}, \text{DK}_{id,t}, \text{CT}_{id,t}) \rightarrow M$ As before.
- $\text{KeyRev}(id, t, \text{RL}, \text{ST}_U) \rightarrow \text{RL}$ As before.

The security proof of our modified RIBE scheme can be obtained by combining techniques used for our original scheme and those of [25]. The details of the proof will be provided in a full version of this paper.

4.3 Revocable Attribute-based Encryption

Attribute-based encryption (ABE) is a generalization of IBE and Fuzzy IBE. In this subsection, we consider a variant of ABE called key-policy ABE (KP-ABE) [20], in which a message is encrypted under a set of descriptive attributes (as compared to just a single identity in the normal IBE setting), and a private key is associated with an access policy that specifies which kind of ciphertexts this particular private key is able to decrypt. A private key could decrypt a ciphertext that is associated with an attribute set only if the attribute set satisfies the access policy associated with the key.

Boldyreva et al. [7] sketched a construction of revocable KP-ABE using the binary tree method. Subsequently Sahai et al. [31] extended their idea and gave a complete construction with a security proof. Similarly, our accumulator-based revocation technique can be extended to the KP-ABE setting. Intuitively, we rely on an accumulator to capture all valid (non-revoked) attributes such that they can be represented with a single group element. Without loss of generality, we assume that an attribute can be a user identity. This way, we can revoke not only a common attribute shared among multiple users, but also a unique identity attribute to revoke a user. We now sketch a construction of revocable KP-ABE that is based on Lewko et al.'s adaptively secure KP-ABE scheme [24].

Let U, V, V_w be sets as defined before. Let n be the total number of attributes in the system. For simplicity, we assume that all attributes i are elements in \mathbb{Z}_N and we let S denote the set of attributes under which a message will be encrypted.

- $\text{Setup}(1^k, n) \rightarrow (\text{PP}, \text{MK}, \text{RL}, \text{ST}_\emptyset)$ As with that of our RIBE, except that it runs the setup algorithm of the scheme of [24]. The public parameters PP are $\langle N, g, g^\beta, e(g, g)^\alpha, z, pk, T_i, \text{AC}_\emptyset, P_1, \dots, P_n, P_{n+2}, \dots, P_{2n} \rangle$. The master secret key MK is $\langle \alpha, \beta, \gamma, sk \rangle$ and a generator X_3 of G_{p_3} .

- $\text{Enc}(\text{PP}, M, S, \text{AC}_V) \rightarrow \text{CT}$ As with that of our RIBE, except that given an attribute set S , the ciphertext CT is set to be

$$C = M \frac{e(g, g)^{\alpha s}}{z^s}, C_0 = g^s, C_1 = (g^\beta \text{AC}_V)^s, C_i = T_i^s$$

for all $i \in S$.

- $\text{PriKeyGen}(\text{PP}, \text{MK}, (A, \rho)) \rightarrow \text{SK}$ For each row x appears in an access matrix A , there is a corresponding attribute i such that $i = \rho(x)$.⁵ The algorithm generates the witness for each $\rho(x) \in A$ and updates the accumulator and state as with that of our RIBE. To generate a private key, the algorithm chooses a random vector \mathbf{u} such that $\mathbf{1} \cdot \mathbf{u} = \alpha$ (here, $\mathbf{1}$ denotes the vector with the first entry equals to 1 and the rest are all 0's), and a vector \mathbf{v} such that the first entry is β while the other entries are all 0's. For each row A_x of A , it chooses a random $r_x \in \mathbb{Z}_N$, and random elements $W_x, V_x \in G_{p_3}$. The private key SK is then

$$K_x^1 = g^{A_x \cdot \mathbf{u}} T_{\rho(x)}^{r_x} P_{\rho(x)}^{A_x \cdot \mathbf{v}} W_x, K_x^2 = g^{r_x} V_x, K_x^3 = w_{\rho(x)}.$$

- $\text{KeyUpd}(\text{PP}, \text{MK}, \text{RL}, \text{ST}_U) \rightarrow \text{KU}$ As before, except that instead of adding/removing identities and times, the algorithm adds/removes attributes into/from the accumulator.
- $\text{DecKeyGen}(\text{SK}, \text{KU}) \rightarrow \text{DK}$ As before, but instead of updating the witness for just an identity, the algorithm updates the witness corresponding to each row of A .
- $\text{Dec}(\text{PP}, \text{DK}, \text{CT}) \rightarrow M$ If the attribute set S satisfies the access matrix A , the decryption algorithm computes constants ω_x such that $\sum_{\rho(x) \in S} \omega_x A_x = \mathbf{1}$. It then computes the blinding factor as

$$\prod_{\rho(x) \in S} \left(\frac{e(C_0, K_x^1)^{\omega_x}}{e(C_{\rho(x)}, K_x^2)^{\omega_x}} \cdot \frac{e(C_0, K_x^3)^{\omega_x A_x \cdot \mathbf{1}}}{e(P_{\rho(x)}, C_1)^{\omega_x A_x \cdot \mathbf{1}}} \right) = \frac{e(g, g)^{\alpha s}}{z^s}.$$

- $\text{KeyRev}(i, \text{RL}, \text{ST}_U) \rightarrow \text{RL}$ As before.

We can prove the security of the above revocable KP-ABE using similar techniques as those for our RIBE scheme and those used in [24]. Further details of the above scheme and its security proof will be shown in a full version of this paper.

We believe that similar techniques can be applied to obtain a revocable ciphertext-policy ABE (CP-ABE) scheme, another variant of ABE that reverses the properties of KP-ABE. That is, it encrypts a message with an access policy and generates a private key according to an attribute set.

5 Conclusions

In this paper, we proposed a very efficient and adaptively secure RIBE scheme based on an accumulator. Our scheme enjoys constant-size key update, a major improvement from all previous RIBE schemes.

⁵Here ρ is a map from the row A_x of A to an index i .

One immediate open problem would be to achieve adaptive security under more standard assumptions. Also, it would be interesting to investigate if our accumulator-based key update technique can be applied to revocable storage ABE proposed by Sahai et al. [31] and other variants of functional encryption.

Acknowledgment: The first author is supported by the A*STAR Graduate Scholarship. The other three authors are supported by the Singapore Ministry of Education Research Grant MOE2013-T2-1-041 and National Research Foundation of Singapore Research Grant NRF-CRP22007-03. The authors are thankful to the very useful comments from the anonymous reviewers.

References

- [1] William Aiello, Sachin Lodha, and Rafail Ostrovsky. Fast digital identity revocation. In *CRYPTO*, pages 137–152. Springer, 1998.
- [2] Nuttapong Attrapadung and Hideki Imai. Attribute-based encryption supporting direct/indirect revocation modes. In *IMA International Conference on Cryptography and Coding*, pages 278–300. Springer, 2009.
- [3] Nuttapong Attrapadung and Hideki Imai. Conjunctive broadcast and attribute-based encryption. In *Pairing*, pages 248–265. Springer, 2009.
- [4] Joonsang Baek and Yuliang Zheng. Identity-based threshold decryption. In *PKC*, page 262. Springer, 2004.
- [5] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. Recommendation for key management – part 1: General (revision 3). In *NIST Special Publication 800-57*, 2012.
- [6] Josh Benaloh and Michael de Mare. One-way accumulators: a decentralized alternative to digital signatures. In *EUROCRYPT*, pages 274–285. Springer, 1993.
- [7] Alexandra Boldyreva, Vipul Goyal, and Virendra Kumar. Identity-based encryption with efficient revocation. In *ACM Conference on Computer and Communications Security*, pages 417–426. ACM, 2008.
- [8] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT*, page 440. Springer, 2005.
- [9] Dan Boneh, Xuhua Ding, Gene Tsudik, and Chi Ming Wong. A method for fast revocation of public key certificates and security capabilities. In *USENIX*, page 22. USENIX Association, 2001.
- [10] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, page 213. Springer, 2001.
- [11] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO*, pages 258–275. Springer, 2005.

- [12] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Proceedings of the Second International Conference on Theory of Cryptography*, pages 325–341. Springer, 2005.
- [13] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *PKC*, pages 481–500. Springer-Verlag, 2009.
- [14] Jie Chen, Hoon Wei Lim, San Ling, Huaxiong Wang, and Khoa Nguyen. Revocable identity-based encryption from lattices. In *ACISP*, pages 390–403. Springer, 2012.
- [15] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 360–363. Springer, 2001.
- [16] Cécile Delerablée, Pascal Paillier, and David Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In *Pairing*, pages 39–59. Springer, 2007.
- [17] Xuhua Ding and Gene Tsudik. Simple identity-based cryptography with mediated RSA. In *CT-RSA*, pages 193–210. Springer, 2003.
- [18] Matt Franklin. An introduction to identity-based encryption. In *NIST Identity-based Encryption Workshop*, 2008.
- [19] Juan Manuel González-Nieto, Mark Manulis, and Dongdong Sun. Fully private revocable predicate encryption. In *ACISP*, pages 350–363. Springer, 2012.
- [20] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98. ACM, 2006.
- [21] Yumiko Hanaoka, Goichiro Hanaoka, Junji Shikata, and Hideki Imai. Identity-based hierarchical strongly key-insulated encryption and its application. In *ASIACRYPT*, pages 495–514. Springer, 2005.
- [22] Russell Housley, W Polk, Warwick Ford, and David Solo. Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile, 2002.
- [23] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162. Springer, 2008.
- [24] Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91. Springer, 2010.
- [25] Allison Lewko and Brent Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *TCC*, page 455. Springer, 2010.
- [26] Benoît Libert and Jean-Jacques Quisquater. Efficient revocation and threshold pairing based cryptosystems. In *Proceedings of the 22nd Annual Symposium on Principles of Distributed Computing*, pages 163–171. ACM, 2003.

- [27] Benoît Libert and Damien Vergnaud. Adaptive-ID secure revocable identity-based encryption. In *CT-RSA*, pages 1–15. Springer, 2009.
- [28] Michael Myers, Rich Ankney, Ambarish Malpani, Slava Galperin, and Carlisle Adams. X.509 internet public key infrastructure online certificate status protocol-OCSP. Technical report, 1999.
- [29] Moni Naor and Kobbi Nissim. Certificate revocation and certificate update. In *USENIX*, page 17. USENIX Association, 1998.
- [30] Duong-Hieu Phan, David Pointcheval, Siamak F Shahandashti, and Mario Strefer. Adaptive CCA broadcast encryption with constant-size secret keys and ciphertexts. In *ACISP*, pages 308–321. Springer, 2012.
- [31] Amit Sahai, Hakan Seyalioglu, and Brent Waters. Dynamic credentials and ciphertext delegation for attribute-based encryption. In *CRYPTO*, pages 199–217. Springer, 2012.
- [32] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473. Springer, 2005.
- [33] Jae Hong Seo and Keita Emura. Revocable identity-based encryption revisited: Security model and construction. In *PKC*, pages 216–234. Springer, 2013.
- [34] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, page 47. Springer, 1984.
- [35] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO*, pages 619–636. Springer, 2009.

A Example

In this section, we provide a simple but illustrative example to show how the sets V, V_w, U are constructed, how a user witness is generated, and how to perform key revocation and update. For ease of exposition, we omit the private key generation process and abuse some of the notations.

Scenario: Let’s assume that we currently have id_1, id_2, id_3 as the legitimate users in the system under current time period t . For simplicity and without loss of generality, let $\phi(id_i) = i$ for $i = 1, 2, 3$ and $\phi(t) = 4$. We then have $V = \{1, 2, 3, 4\}$ and $AC_V = \prod_{j \in V} P_{n+1-j} = P_n P_{n-1} P_{n-2} P_{n-3}$. Also, we have $U = V$.

Add user: Now assume that user id_5 , where $\phi(id_5) = 5$, joins the system within the same time period t . The key authority (KA) first updates the set V as $V = \{1, 2, 3, 4\} \cup \{5\}$, then issues a witness $w_5 = \prod_{j \in V, j \neq 5} P_{n+1-j+5} = P_{n+5} P_{n+4} P_{n+3} P_{n+2}$ and a unique set $V_w = \{1, 2, 3, 4, 5\}$ to user id_5 . Moreover, the KA updates the accumulator $AC_V = P_n P_{n-1} P_{n-2} P_{n-3} P_{n-4}$ according to the most recent set V . Also it updates U as $U = \{1, 2, 3, 4\} \cup \{5\}$. Note that no key update information will be broadcast to other system users since the time period t has not elapsed yet.

Key update by KA: Assume that user id_2 has been revoked before the expiry of time period t . At the beginning of a subsequent new time period t' , where $\phi(t') = 6$, the KA performs the following steps:

1. Remove $\{2, 4\}$ from V (recall 2 corresponds to user id_2 and 4 corresponds to time period t) and add $\{6\}$ into V , obtaining $V = \{1, 3, 5, 6\}$; then compute a new accumulator $AC_V = P_n P_{n-2} P_{n-4} P_{n-5}$, and update the set U as $U = \{1, 2, 3, 4, 5\} \cup \{6\}$. (Recall U is the set containing all elements that have ever been added into the accumulator.) The KA also sets ΔV to be $\{2, 4, 5, 6\}$.
2. Calculate the witness for time period t' as $w_6 = P_{n+6} P_{n+4} P_{n+2}$.
3. Generate a signature $\sigma_{t'}$ on $AC_{V''}$, broadcast $KU_{t'} = \langle AC_{V''}, \sigma_{t'}, w_6 \rangle$ together with the set ΔV .

Key update by user: During key update, each user first verifies the integrity and authenticity of the accumulator, as described in the scheme. Then, the user derives the latest set V and update his witness. For example, for user id_5 , he first obtains the set $V = \{1, 3, 5, 6\}$ (this could easily be calculated from the broadcast set ΔV and his own set V_w); using his own set $V_w = \{1, 2, 3, 4, 5\}$, he then derives $V \setminus V_w = \{6\}$ and $V_w \setminus V = \{2, 4\}$, and updates w_5 as described in the scheme to obtain the latest witness used for decryption.

Encrypt & decrypt: The encryption and decryption algorithms are straightforward.

From the above example, one can easily verify that an accumulator is always in the form of $AC_V = \prod_{j \in V} P_{n+1-j}$ and a witness for value i has the value $w_i = \prod_{j \in V, j \neq i} P_{n+1-j+i}$.