7-2017

Mergeable and revocable identity-based encryption

Shengmin XU
Singapore Management University, smxu@smu.edu.sg

Guomin YANG

Yi MU

Willy SUSILO

Mergeable and Revocable Identity-Based Encryption

Shengmin Xu(✉), Guomin Yang(✉), Yi Mu(✉), and Willy Susilo(✉)

School of Computing and Information Technology,
Institute of Cybersecurity and Cryptology, University of Wollongong,
Wollongong, Australia
{sx914,gyang,ymu,wsusilo}@uow.edu.au

Abstract. Identity-based encryption (IBE) has been extensively studied and widely used in various applications since Boneh and Franklin proposed the first practical scheme based on pairing. In that seminal work, it has also been pointed out that providing an efficient revocation mechanism for IBE is essential. Hence, revocable identity-based encryption (RIBE) has been proposed in the literature to offer an efficient revocation mechanism. In contrast to revocation, another issue that will also occur in practice is to combine two or multiple IBE systems into one system, e.g., due to the merge of the departments or companies. However, this issue has not been formally studied in the literature and the naive solution of creating a completely new system is inefficient. In order to efficiently address this problem, in this paper we propose the notion of mergeable and revocable identity-based encryption (MRIBE). Our scheme provides the first solution to efficiently revoke users and merge multiple IBE systems into a single system. The proposed scheme also has several nice features: when two systems are merged, there is no secure channel needed for the purpose of updating user private keys; and the size of the user private key remains unchanged when multiple systems are merged. We also propose a new security model for MRIBE, which is an extension of the security model for RIBE, and prove that the proposed scheme is semantically secure without random oracles.

Keywords: Identity-based encryption · Revocation · Merging

1 Introduction

Public key encryption is the most basic primitive of public key cryptography. However, it suffers from the key distribution and management problem. To overcome this drawback, identity-based encryption (IBE) has been proposed, and it provides a new paradigm for public key encryption [2,3,5,20]. IBE uses the identity string (e.g. emails or IP addresses) of a user as the public key of that user. The sender using an IBE does not need to look up the public keys and the corresponding certificates of the receivers, because the identities together with common public parameters are sufficient for encryption. The private keys of all the users are generated by a private key generator (PKG) which is a fully trusted third party.

J. Pieprzyk and S. Suriadi (Eds.): ACISP 2017, Part I, LNCS 10342, pp. 147–167, 2017.
DOI: 10.1007/978-3-319-60055-0_8

Revocation is an essential requirement in a cryptographic system when a user's key is compromised and/or any misuse is noticed. In PKI, revocation is done via certificate revocation lists (CRLs). However, IBE cannot apply this approach since there is no certificate in the system. Boneh and Franklin provided the first practical IBE scheme, and they also proposed a revocation mechanism by appending the timestamp in each identity string, but the workload of updating the private key is linear to the size of the non-revoked users. To address this issues, some practical Revocable IBE (RIBE) schemes have been proposed [1, 11,15]. Boldyreva et al. [1] proposed the first practical RIBE scheme with the authority's periodic workload to be logarithmic in the number of users while keeping the scheme efficient in both encryption and decryption. However, their RIBE scheme limits the number of users in the system. To overcome this problem, they proposed a method to double the number of users in the system. However, the size of the private key for each user is also increased whenever the size of the system is increased. Some following works [11,15] have focused on improving the security from selective security to adaptive security.

RIBE schemes are useful in many applications such as email systems and data storage systems by disallowing unauthorised or revoked users to access encrypted sensitive information in those systems. However, in practice it is also possible that two or more IBE systems need to be merged due to various reasons. As an example, a university wants to merge two departments: information technology (IT) and computer science (CS). A naive approach to address this issue is creating a completely new system and re-generating the public parameter and private keys for all users. However, this approach is impractical since a new private key needs to be generated for all the users in the combined system and a secure channel needs to be established between each user and the PKG for key distribution.

In this paper, we propose a new notion called mergeable and revocable identity-based encryption (MRIBE) to solve the above problem. Our scheme inherits the advantage of RIBE schemes by allowing the authority (i.e., PKG) to efficiently revoke users. In addition, our scheme allows different systems to be merged into a single system while keeping the size of the user private key unchanged. Also, there is no secure channel needed for updating the user private keys during the merging process.

1.1 Related Work

The concept of identity-based cryptography was introduced by Shamir [19], but the first practical IBE scheme was proposed by Boneh and Franklin in 2001 [3] and the scheme is proved secure in the random oracle model. Since the random oracle model is an idealised model, Boneh and Boyen [2] proposed a selectively security IBE scheme without random oracles in 2004. One year later, Waters [20] proposed a new IBE scheme with adaptive security in the standard model, but the size of the public parameter depends on the length of the user identity. To reduce the size of the public parameter, Gentry [5] proposed another IBE scheme in the standard model, but its security is based on a non-standard assumption.

RIBE is an extension of IBE by providing an efficient revocation mechanism. The issue of revocation in IBE has been pointed out by Boneh and Franklin in their seminal work [3]. They suggested that users renew their private keys periodically by representing an identity as $\mathsf{ID}\|\mathsf{T}$ where ID is the real identity and T is the current time. However, such an approach is inefficient and not scalable because a secure channel between the PKG and each user needs to be established each time, and the workload of generating new private keys is linear in the number of non-revoked users in each revocation epoch. Hanaoka et al. [8] proposed an approach that the users periodically renew their private keys without interacting with the PKG but each user needs to posses a tamper-resistant hardware device. This assumption makes the solution rather impractical. Boldyreva et al. [1] introduced a scalable but selectively secure RIBE by utilizing several techniques including fuzzy identity-based encryption [13], secret sharing [19] and the tree-based revocation method proposed for broadcast encryption [4,7,10,12,21]. Libert and Vergnaud [11] proposed the first adaptively secure RIBE scheme. Seo and Emura [15] improved the security model in [1] to prevent decryption key exposure attacks. Lee et al. [9] proposed a RIBE scheme by utilizing subset difference (SD) method instead of the Complete Subtree (CS) method which is used in all previous works. Recently, the techniques used in RIBE have also been extended to achieve revocable hierarchical identity-based encryption (RHIBE) [14,16,17].

1.2 Our Contributions

In this work, we propose a new cryptographic notion named mergeable and revocable identity-based encryption (MRIBE), which is an extension of revocable identity-based encryption (RIBE). The proposed MRIBE scheme inherits all the nice properties of RIBE, in particular the property of allowing efficient revocation, and also allows multiple IBE systems to be merged into a single system, which makes it more versatile in handling the dynamics that could occur in real applications.

We also give a new security model for MRIBE by extending of the security model for RIBE and prove that the proposed scheme is semantically secure in the standard model. Our scheme is based on RIBE by introducing several new algorithms to handle the merging functionality. The proposal scheme also has some several nice features: there is no secure channel needed for key update during the merging process; and the size of user private key remains unchanged when multiple systems are merged, which makes the system scalable.

1.3 Paper Organization

Some preliminaries are introduced in the next section. In Sect. 3, we provide definitions for the MRIBE scheme and its security model. We then present our MRIBE construction in Sect. 4. The security proof of the proposed scheme is provided in Sect. 5. Finally, we summarize our result in Sect. 6.

2 Preliminaries

In this section, we introduce the notations used in this paper and review the definitions for bilinear map and pseudorandom function family. We also review some cryptographic primitives, including threshold secret sharing scheme, fuzzy identity-based encryption scheme, and revocable identity-based encryption scheme.

2.1 Notations

Let $\mathbb{N}$ denote the set of all natural numbers, and for $n \in \mathbb{N}$, we define $[n] := \{1, ..., n\}$. "$x \leftarrow y$" denotes that x is chosen uniformly at random from y if y is a finite set, x is output from y if y is a function or an algorithm, or y is assigned to x otherwise. If x and y are strings, then "$|x|$" denotes the bit-length of x, "$x\|y$" denotes the concatenation of x and y. For a finite set S, "$|S|$" denotes its size and $S[i]$ denotes the i-th value in the set S. If $\mathcal{A}$ is a probabilistic algorithm, then "$y \leftarrow \mathcal{A}(x; r)$" denotes that $\mathcal{A}$ computes y as output by taking x as input and using r as randomness, and we just write "$y \leftarrow \mathcal{A}(x)$" if we do not need to make the randomness used by $\mathcal{A}$ explicit. If furthermore $\mathcal{O}$ is a function or an algorithm, then "$\mathcal{A}^{\mathcal{O}}$" means that $\mathcal{A}$ has oracle access to $\mathcal{O}$. A function $\epsilon(k) : \mathbb{N} \rightarrow [0, 1]$ is said to be *negligible* if for all positive polynomials $p(k)$ and all sufficiently large $k \in \mathbb{N}$, we have $\epsilon(k) < 1/p(k)$. Throughout this paper, we use the character "k" to denote a security parameter.

2.2 Bilinear Map

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic multiplicative groups of prime order p and g be a generator of $\mathbb{G}$. The map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is said to be an admissible bilinear pairing if the following properties hold true.

1. Bilinearity: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneration: $e(g, g) \neq 1$.
3. Computability: it is efficient to compute $e(u, v)$ for any $u.v \in \mathbb{G}$.

We say that $(\mathbb{G}, \mathbb{G}_T)$ are bilinear map groups if there exists a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ as above.

2.3 Pseudorandom Function Family

Goldreich, Goldwasser and Micali [6] introduced approaches to constructing random functions in 1984. In this section, we review the definition of pseudorandom function and pseudorandom function family.

Definition 1 (Pseudorandom Function). *Let $k \in \mathbb{N}$ be a security parameter. A function family F is associated with $\{\mathsf{Seed}_k\}_{k\in\mathbb{N}}, \{\mathsf{Dom}_k\}_{k\in\mathbb{N}}$ and $\{\mathsf{Rng}_k\}_{k\in\mathbb{N}}$. Formally, for any $\sum \leftarrow \mathsf{Seed}_k, \mathcal{D} \leftarrow \sum$ and $\mathcal{R} \leftarrow \mathsf{Rng}_k$, $\mathsf{F}_\omega^{k,\sum,\mathcal{D},\mathcal{R}}$ defines a function which maps an element of $\mathcal{D}$ to an element of $\mathcal{R}$. That is, $\mathsf{F}_\omega^{k,\sum,\mathcal{D},\mathcal{R}}(\rho) \in \mathcal{R}$ for any $\rho \in \mathcal{D}$.*

Definition 2 (Pseudorandom Function Family). F *is a pseudorandom function family if* $\mathsf{F}_{\omega}^{k,\sum,\mathcal{D},\mathcal{R}}(\rho_i)$ *and* $RF(\rho_i)$ *computational indistinguishability for any* $\rho_i \in \mathcal{D}$ *adaptively chosen by any polynomial time distinguisher, where* RF *is a truly random function. That is, for any* $\rho \in \mathcal{D}, RF(\rho) \leftarrow \mathcal{R}$.

2.4 Threshold Secret Sharing Scheme

Shamir's secret sharing scheme [18] divides a secret s into n pieces $s_1, ..., s_n$ using a unique polynomial of degree $(t-1)$, any t out of n shares may be used to recover the secret. The details are shown as follow.

Choose a group $\mathbb{Z}_p$ and $p \geq n$. Each user u_i is associated with a public unique number $u_i \in \mathbb{Z}_p$ and the user set $\mathsf{U} = \{u_1, ..., u_n\}$. Choose a random $k-1$ degree polynomial $p(x) = s + \prod_{i=1}^{k-1} a_i x^i$ where $a_i \in \mathbb{Z}_p^*$. Each user in U obtains a share $s_i = p(u_i)$. When k users come together and form a set $\mathsf{J} \subseteq \mathsf{U}$, the secret s can be recovered by utilizing Lagrange coefficient and polynomial interpolation. For $x, i \in \mathbb{Z}$, set $\mathsf{J} \subset \mathbb{Z}$ the *Lagrange coefficient* $\Delta_{i,\mathsf{J}}(x)$ is defined as

$$\Delta_{i,\mathsf{J}}(x) = \prod_{j \in \mathsf{J}, j \neq i} \left(\frac{x-j}{i-j} \right).$$

To recover $p(x)$, we have following equation:

$$p(x) = \sum_{i \in \mathsf{J}} s_i \cdot \Delta_{i,\mathsf{J}}(x).$$

Hence, we can recover the secret key s by setting the element x is equal to 0:

$$s = p(0) = \sum_{i \in \mathsf{J}} s_i \cdot \Delta_{i,\mathsf{J}}(0).$$

2.5 Fuzzy Identity-Based Encryption Scheme

Sahai and Waters [13] proposed a new type of identity-based encryption scheme called fuzzy identity-based encryption. It is the first attribute-based encryption scheme. There are two schemes, one has to define the universe in the setup phase, and the other one has a large universe. In the large universe construction, it utilizes all elements of $\mathbb{Z}_p^*$ as the universe and defines the following function to cooperate Shamir's secret sharing scheme to recover the plaintext from the ciphertext with J attributes. For $x \in \mathbb{Z}; \mathsf{J} \subset \mathbb{Z}; g, h_1, ..., h_{|\mathsf{J}|} \in \mathbb{G}$, we define

$$H_{g,\mathsf{J},h_1,...,h_{|\mathsf{J}|}} \stackrel{\text{def}}{=} g^{x^{|\mathsf{J}|-1}} \prod_{i=1}^{|\mathsf{J}|} \left(h_i^{\Delta_{i,\mathsf{J}}(x)} \right)$$

The large universe construction can be used to build revocable identity-based encryption scheme as follows. The private key generation centre issues the private key for every user based on the identity ω and the time t in each revocation epoch for the non-revoked user. The ciphertext is encrypted under the identity ω and time t. So the revoked users cannot decrypt the ciphertext since they do not have valid time t component.

2.6 Identity-Based Encryption with Revocation Scheme

Boldyreva, Goyal and Kumar [1] points out that the revocation list can be implemented by the complete subtree method [12]. Since our proposed scheme is mergeable, we slightly modify their revocation scheme. Let x_c denote the children of node x. For the root node, it has 2 or more degrees since the mergence, which is differ to the previous work [1]. For the non-root node, it only has two children x_l and x_r. The detail of revocation algorithm is described as follows.

The function $\mathsf{KUNodes}$ takes three parameters as input, a binary tree T, revocation list rl and time t. It outputs a set of nodes, which is the minimal set of nodes in the binary tree T such that the non-revoked nodes have at least one ancestor or themselves in the set and none of revoked nodes in revocation list rl have any ancestor or themselves in the set. The function operates as follows. First it marks all the ancestors of revoked nodes as revoked into the set X, then output all the non-revoked children of revoked nodes in the set Y. Here is a formal specification.

$\mathsf{KUNodes}(\mathsf{T}, rl, t)$
$\mathsf{X}, \mathsf{Y} \leftarrow \emptyset$
$\forall (v_i, t_i) \in rl$ if $t_i \leq t$ then add $\mathsf{Path}(v_i)$ to X
$\forall x \in \mathsf{X}$ if $x_c \notin \mathsf{X}$ then add x_c to Y
If $\mathsf{Y} = \emptyset$ then add root to Y
Return Y

Our scheme is based on the binary tree except the root node has more than two children. After merging, our revocation tree improves the degree of the root node rather than the depth of the binary tree. Let N denote the number of user for each system and N_S denote the number of systems. Our tree structure keeps the unchanged size of depth $\log_2 N$. Thus the number of the private key for each user remains unchanged $\log_2 N$. After merging all N_S system, the new root node has $2N_S$ degrees. It improves the width N_S times compared to the original revocation tree. However, this tree structure does not reduce the efficient since N_S is not a significant number and we can re-build the whole system if N_S is too large.

3 Formal Definitions and Security Models

3.1 Syntax of Mergeable and Revocable IBE

We start with defining the general syntax of a Mergeable and Revocable IBE scheme. We recall and modify the definition of revocable IBE schemes as defined in [1]. Each algorithm is run by one of following parties - key authority, sender or receiver. The key authority maintains a revocation list rl and state st. We define parameter N_S as the number of system in our proposed scheme and use the Greek characters as the subscript to represent the instantiations of different systems in this section and following sections.

Definition 3 (Mergeable and Revocable IBE). *A mergeable and revocable identity-based encryption scheme* $\mathcal{MRIBE} = (\mathcal{S}, \mathcal{SK}, \mathcal{KU}, \mathcal{DK}, \mathcal{E}, \mathcal{D}, \mathcal{R}, \mathcal{MP}, \mathcal{MSK}, \mathcal{SKU})$ *is defined by ten algorithms and has associated message space* $\mathcal{M}$*, identity space* $\mathcal{I}$ *and time space* $\mathcal{T}$*. In what follows, we call an algorithm stateful only if it updates* rl *or* st*.*

The stateful ***Setup*** *algorithm* $\mathcal{S}$ *is run by the key authority. Given a security parameter* k*, a maximal number of users* N *and a number of systems* N_S*, it outputs one public parameters* pp *which shares to all* N_S *systems, and generates a public key* pk_i*, a master secret key* msk_i*, a revocation list* rl_i *and a state* st_i *for each system* ($i \in \{1, ..., N_S\}$)*.*

The stateful ***Private Key Generation*** *algorithm* $\mathcal{SK}$ *is run by the key authority. Given the information of the public key, the master secret key and the state* $(pk_i, msk_i, state_i)$ *in system* i *and an identity* $\omega \in \mathcal{I}$*, it outputs private key* $sk_{\omega,i}$ *and an updated state* st_i*.*

The ***Key Update*** *algorithm* $\mathcal{KU}$ *is run by the key authority. Given the information of the public key, the master secret key, the revocation list and state* $(pk_i, msk_i, rl_i, st_i)$ *in system* i *and a revocation epoch* $t \in \mathcal{T}$*, it outputs a key update* $ku_{t,i}$*.*

The ***Decryption Key Generation*** *algorithm* $\mathcal{DK}$ *is run by the receiver. Given a private key* $sk_{\omega,i}$ *and a key update* $ku_{t,i}$*, it outputs decryption key* $dk_{\omega,t,i}$ *or a special symbol* $\perp$ *indicating that* ω *was revoked.*

The ***Encryption*** *algorithm* $\mathcal{E}$ *is run by the sender. Given a public key* pk_i *in system* i *and an identity* $\omega \in \mathcal{I}$*, an encryption time* $t \in \mathcal{T}$ *and a message* $m \in \mathcal{M}$*, it outputs a ciphertext* c_i*. For simplicity and w.l.o.g. we assume that* ω *and* t *are efficiently computable from* c_i*.*

The ***Decryption*** *algorithm* $\mathcal{D}$ *is run by the receiver. Given a decryption key* $dk_{\omega,t,i}$ *and a ciphertext* c_i*, it outputs a message* $m \in \mathcal{M}$ *or a special symbol* $\perp$ *indicating that the ciphertext is invalid.*

The stateful ***Revocation*** *algorithm* $\mathcal{R}$ *is run by the key authority. Given an identity to be the revoked* $\omega \in \mathcal{I}$*, revocation list* rl_i *and state* st_i *in system* i *and revocation time* $t \in \mathcal{T}$ *it outputs updated revocation list* rl_i*.*

The stateful ***Merge Parameter*** *algorithm* $\mathcal{MP}$ *is run by the key authority. Given the public key* pk_α*, the master key* msk_α*, the revocation list* rl_α *and the state* st_α *in system* α*, the public key* pk_β*, the master key* msk_β*, the revocation list* rl_β *and the state* st_β *in system* β*, it outputs updated revocation list* rl_β *and state* st_β*.*

The ***Merge Private Key*** *algorithm* $\mathcal{MSK}$ *is run by the key authority. Given the public key* pk_α*, the master key* msk_α *and the state* st_α *in system* α*, the public key* pk_β*, the master key* mk_β *and the state* st_β *in system* β *and an identity* $\omega \in \mathcal{I}$*, it outputs a mergeable private key* $sk_{\omega,\alpha,\beta}$*.*

The ***Private Key Update*** *algorithm* $\mathcal{SKU}$ *is run by the receiver. Given the private key* $sk_{\omega,\alpha}$ *and the mergeable private key* $sk_{\omega,\alpha,\beta}$*, it outputs the private key* $sk_{\omega,\beta}$*.*

Correctness requires that, for any outputs of $\mathcal{S}$, any $m \in \mathcal{M}$, $\omega \in \mathcal{I}$ and $t \in \mathcal{T}$, all possible states and revocation lists, the following experiments return 1 with probability 1:

– The key authority generates all public parameters for N_S systems:

$$(pp, \{pk_i, msk_i, rl_i, st_i\}_{i\in\{1,...,N_S\}}) \leftarrow \mathcal{S}(k, N, N_S); \alpha, \beta \leftarrow \{1, ..., N_S\}.$$

– ω_1 is a valid user in system α:

$$(sk_{\omega_1,\alpha}, st_\alpha) \leftarrow \mathcal{SK}(pk_\alpha, msk_\alpha, st_\alpha, \omega_1); ku_{t,\alpha} \leftarrow \mathcal{KU}(pk_\alpha, msk_\alpha, rl_\alpha, st_\alpha, t);$$
$$dk_{\omega_1,t,\alpha} \leftarrow \mathcal{DK}(sk_{\omega_1,\alpha}, ku_{t,\alpha}); c_1 \leftarrow \mathcal{E}(pk_\alpha, \omega_1, t, m_1).$$
$$\text{If } \mathcal{D}(dk_{\omega_1,t,\alpha}, c_1) \neq m_1 \text{ then return 0; else return 1}$$

– ω_2 is a revoked user in system β:

$$(sk_{\omega_2,\beta}, st_\beta) \leftarrow \mathcal{SK}(pk_\beta, msk_\beta, st_\beta, \omega_2); rl_\beta \leftarrow \mathcal{R}(\omega_2, rl_\beta, st_\beta, t);$$
$$ku_{t,\beta} \leftarrow \mathcal{KU}(pk_\beta, msk_\beta, rl_\beta, st_\beta, t).$$
$$\text{If } \mathcal{DK}(sk_{\omega_2,\beta}, ku_{t,\beta}) \neq \perp \text{ then return 0; else return 1}$$

– ω_1 is a valid user in system α and merges to system β:

$$(rl_\beta, st_\beta) \leftarrow \mathcal{MP}(\{pk_i, msk_i, rl_i, st_i\}_{i\in\{\alpha,\beta\}});$$
$$sk_{\omega_1,\alpha,\beta} \leftarrow \mathcal{MSK}(\{pk_i, msk_i, st_i\}_{i\in\{\alpha,\beta\}}, \omega_1);$$
$$sk_{\omega_1,\beta} \leftarrow \mathcal{SKU}(sk_{\omega_1,\alpha}, sk_{\omega_1,\alpha,\beta});$$
$$dk_{\omega_1,t,\beta} \leftarrow \mathcal{DK}(sk_{\omega_1,\beta}, ku_{t,\beta}); c_3 \leftarrow \mathcal{E}(pk_\beta, \omega_1, t, m_3).$$
$$\text{If } \mathcal{D}(dk_{\omega_1,t,\beta}, c_3) = m_3 \text{ then return 1; else return 0.}$$

3.2 Security of Mergeable and Revocable IBE

We define the *selective-mergeable-and-revocable-ID* security for mergeable and revocable IBE scheme. Our security model is based on the model for *selective-revocable-ID security* defined in [1].

Definition 4 (sMRID security). *Let* $\mathcal{MRIBE} = (\mathcal{S}, \mathcal{SK}, \mathcal{KU}, \mathcal{DK}, \mathcal{E}, \mathcal{D}, \mathcal{R}, \mathcal{MP}, \mathcal{MSK}, \mathcal{SKU})$ *be a mergeable and revocable IBE scheme defined by the security parameter* k*, the maximum number of user* N *and the number of systems* N_S*. The adversary first outputs the challenging identity* ω^**, a challenging time* t^* *and a subscript of challenging public key* i^**, and also some state information it wants to preserve. Later it is given access to five oracles that correspond to the algorithms of the scheme. The* ***Private Key Generation Oracle*** $\mathcal{O}_{\mathcal{SK}}(\cdot,\cdot)$ *takes a public key* pk_i *and an identity* ω*, runs* $\mathcal{SK}(pk_i, msk_i, st_i, \omega)$ *to return the private key* $sk_{\omega,i}$*.*

The ***Revocation Oracle*** $\mathcal{O}_{\mathcal{R}}(\cdot,\cdot)$ *takes input an identity* ω *and a time* t *and runs* $\mathcal{R}(\omega, rl_i, st_i, t)$ *to return the updated revocation list* rl_i *else return* $\perp$ *if the identity* ω *does not exist in any system.*

The ***Key Update Oracle*** $\mathcal{O}_{\mathcal{KU}}(\cdot,\cdot)$ *takes input a public key* pk_i *and a time* t *and runs* $\mathcal{KU}(pk_i, msk_i, rl_i, st_i, t)$ *to return key update* $ku_{t,i}$.

The ***Merge Parameter Oracle*** $\mathcal{O}_{\mathcal{MP}}(\cdot,\cdot)$ *takes input public key* pk_α *and public key* pk_β *and runs* $\mathcal{MP}(\{pk_i, msk_i, rl_i, st_i\}_{i\in\{\alpha,\beta\}})$ *to return updated revocation list* rl_β *and state* st_β. *The parameters of system* α *are no longer valid.*

The ***Merge Private Key Oracle*** $\mathcal{O}_{\mathcal{MSK}}(\cdot,\cdot,\cdot)$ *takes input an identity* ω, *a public key* pk_α *and a public key* pk_β *and runs* $\mathcal{MSK}(\{pk_i, msk_i, st_i\}_{i\in\{\alpha,\beta\}}, \omega)$ *to return the mergeable private key* $sk_{\omega,\alpha,\beta}$ *else return* $\perp$ *if related* $\mathcal{MP}(\cdot,\cdot)$ *does not query.*

$$
\begin{aligned}
&\mathsf{Experiment}\ \mathsf{Exp}^{\mathrm{smrid-cpa}}_{\mathcal{MRIBE}}(k, N, N_S)\\
&\quad b \leftarrow \{0,1\}\\
&\quad (\omega^*, t^*, i^*, state) \leftarrow \mathcal{A}(k, N, N_S)\\
&\quad (pp, \{pk_i, msk_i, rl_i, st_i\}_{i\in\{1,\ldots,N_S\}}) \leftarrow \mathcal{S}(k, N, N_S)\\
&\quad (m_0, m_1, state) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{SK}},\mathcal{O}_{\mathcal{R}},\mathcal{O}_{\mathcal{KU}},\mathcal{O}_{\mathcal{MP}},\mathcal{O}_{\mathcal{MSK}}}(state)\\
&\quad c^* \leftarrow \mathcal{E}(pk_{i^*}, \omega^*, t^*, m_b)\\
&\quad d \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathcal{SK}},\mathcal{O}_{\mathcal{R}},\mathcal{O}_{\mathcal{KU}},\mathcal{O}_{\mathcal{MP}},\mathcal{O}_{\mathcal{MSK}}}(pk_{i^*}, c^*, state)\\
&\quad \text{If } b = d \text{ return 1 else return 0.}
\end{aligned}
$$

Note that the following conditions must always hold:

1. $m_0, m_1 \in \mathcal{M}$ and $|m_0| = |m_1|$.
2. If $\mathcal{O}_{\mathcal{SK}}(\cdot,\cdot)$ has been queried on message (pk, ω) then the identity ω has been initialized or merged in the system with the public key pk.
3. $\mathcal{O}_{\mathcal{KU}}(\cdot,\cdot)$ and $\mathcal{O}_{\mathcal{R}}(\cdot,\cdot)$ can be queried on time which is greater than or equal to the time of all previous queries in each system i.e. the adversary is allowed to query only in a non-decreasing order of time. Also, the oracle $\mathcal{O}_{\mathcal{R}}(\cdot,\cdot)$ cannot be queried on time t if $\mathcal{O}_{\mathcal{KU}}(\cdot,\cdot)$ was queried on t.
4. If $\mathcal{O}_{\mathcal{R}}(\cdot,\cdot)$ has been queried on (ω^*, t) for any $t \leq t^*$ then $\mathcal{O}_{\mathcal{SK}}(\cdot,\cdot)$ and $\mathcal{O}_{\mathcal{MSK}}(\cdot,\cdot,\cdot)$ can be queried on identity ω^* without constrain. Otherwise, $\mathcal{O}_{\mathcal{SK}}(\cdot,\cdot)$ and $\mathcal{O}_{\mathcal{MSK}}(\cdot,\cdot,\cdot)$ can be queried on identity ω^* but these queries cannot derive the secret key sk_{ω^*,i^*} in a trivial way. The details are described as follows.

The relationships between private key generation oracle $\mathcal{O}_{\mathcal{SK}}(\cdot,\cdot)$ and merge private key generation oracle $\mathcal{O}_{\mathcal{MSK}}(\cdot,\cdot,\cdot)$ has been described in Fig. 1. Suppose the challenging subscript i^* is γ (challenging public key is pk_γ) and the challenging identity ω^* is a non-revoked user, the adversary cannot obtain the secret key $sk_{\omega,\gamma}$ and then there are two situations to be considered.

- $\mathcal{O}_{\mathcal{SK}}(pk_\gamma, \omega^*)$ cannot be queried.
- Any queries are equivalent to $\mathcal{O}_{\mathcal{SK}}(pk_\gamma, \omega^*)$ cannot be queried, e.g. if $\mathcal{O}_{\mathcal{SK}}(pk_\alpha, \omega^*)$ and $\mathcal{O}_{\mathcal{MSK}}(\omega^*, pk_\alpha, pk_\beta)$ have been queried, $\mathcal{O}_{\mathcal{MSK}}(\omega^*, pk_\beta, pk_\gamma)$ cannot be queried since the former two queries are equivalent to the query $\mathcal{O}_{\mathcal{SK}}(pk_\beta, \omega^*)$ and it is trial to gain the private key sk_γ by continually querying $\mathcal{O}_{\mathcal{MSK}}(\omega^*, pk_\beta, pk_\gamma)$.

We use two database called $\mathcal{D}_{\mathcal{SK}}$ and $\mathcal{D}_{\mathcal{MSK}}$ to record the messages queried to the $\mathcal{O}_{\mathcal{SK}}(\omega^*, \cdot)$ and $\mathcal{O}_{\mathcal{MSK}}(\cdot, \cdot, \omega^*)$ oracles, respectively. We can decide if the adversary can recover the secret key sk_{ω^*,i^*} by checking the database $\mathcal{D}_{\mathcal{SK}}$ and $\mathcal{D}_{\mathcal{MSK}}$.

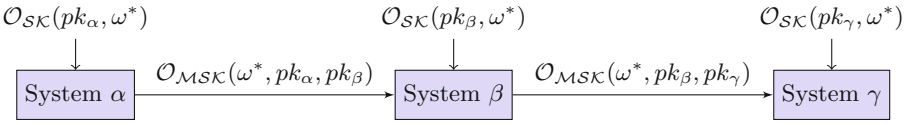

Fig. 1. Relationships of $\mathcal{O}_{\mathcal{SK}}(\cdot, \cdot)$ and $\mathcal{O}_{\mathcal{MSK}}(\cdot, \cdot, \cdot)$

We define the advantage of the adversary $\mathsf{Adv}^{smrid-cpa}_{\mathcal{MRIBE},\mathcal{A},N,N_S}(k)$ as

$$2 \cdot \Pr\left[\mathsf{Exp}^{smrid-cpa}_{\mathcal{MRIBE},\mathcal{A},N,N_S}(k) = 1\right] - 1$$

The scheme is said to be *sMRID-CPA secure* if the function $\mathsf{Adv}^{smrid-cpa}_{\mathcal{MRIBE},\mathcal{A},N,N_S}(k)$ is negligible in k for any efficient algorithm $\mathcal{A}$.

4 The Proposed Schemes

Setup $(pp, \{pk_i, msk_i, rl_i, st_i\}_{i\in\{1,...,N_S\}}) \leftarrow \mathcal{S}(k, N, N_S)$: given a security parameter $k \in \mathbb{N}$, a maximal number of users $N \in \mathbb{N}$ and a maximal number of systems $N_S \in \mathbb{N}$. The key authority defines the valid space of $\mathcal{M}, \mathcal{I}, \mathcal{T}$, define the pseudorandom function $\mathsf{F}_\sigma : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ as well as a complete binary tree T with at least N leaf nodes and does the following.

1. Select bilinear groups $(\mathbb{G}, p, g)$ as the public parameter pp which shares to all systems.
2. For $i \in \{1, ..., N_S\}$ do the following:
   (a) Randomly choose $(a_i, r_i) \leftarrow \mathbb{Z}_p^*$ and set $g_{1,i} \leftarrow g^{a_i}$ as well as randomly choose $g_{2,i}, h_{1,i}, h_{2,i}, h_{3,i} \leftarrow \mathbb{G}$.
   (b) Return public key $pk = (g_{1,i}, g_{2,i}, h_{1,i}, h_{2,i}, h_{3,i})$, master secret key $msk_i = (a_i, r_i)$, revocation list $rl_i = \emptyset$ and state $st_i = \mathsf{T}$.

Private Key Generation $(sk_{\omega,i}, st_i) \leftarrow \mathcal{SK}(pk_i, msk_i, st_i, \omega)$: given an identity $\omega \in \mathcal{I}$, a public key pk_i, a master key msk_i and a state st_i. The key authority generates private key $sk_{\omega,i}$ for the receiver with identity $\omega \in \mathcal{I}$ and the updated state st.

1. Choose an unassigned leaf v from T_i and associate it with $\omega \in \mathcal{I}$.
2. For all node $x \in \mathsf{Path}(v)$ do the following:
   (a) Retrieve a_x from T if it was defined. Otherwise, choose it at random $a_x \leftarrow \mathbb{Z}_p$ and store a_x at node x in $st_i = \mathsf{T}$.

(b) Generate random value $r_x \leftarrow \mathsf{F}_{r_i}(\omega \| x)$ bases on random value r_i in master secret key msk, the identity $\omega \in \mathcal{I}$ as well as the label value x and set

$$D_x \leftarrow g_{2,i}^{a_x \omega + a_i} H_{g_{2,i},\mathsf{J},h_{1,i},h_{2,i},h_{3,i}}(\omega)^{r_x}; d_x \leftarrow g^{r_x}.$$

3. Return private key $sk_{\omega_i} = \{(x, D_x, d_x)\}_{x \in \mathsf{Path}(v)}$ and the updated state $st_i = \mathsf{T}$.

Key Update $ku_{t,i} \leftarrow \mathcal{KU}(pk_i, msk_i, rl_i, st_i, t)$: given a public key pk_i, a master secret key msk_i, a key update time $t \in \mathcal{T}$, a revocation list rl_i and a state st_i. For all nodes $x \in \mathsf{KUNodes}(\mathsf{T}_i, rl_i, t)$.

1. Generate random value $r_x \leftarrow \mathbb{Z}_p$ and set

$$E_x \leftarrow g_{2,i}^{a_x t + a} H_{g_{2,i},\mathsf{J},h_{1,i},h_{2,i},h_{3,i}}(t)^{r_x}; e_x \leftarrow g^{r_x}.$$

2. Return key update $ku_t = \{(x, E_x, e_x)\}_{x \in \mathsf{KUNodes}(\mathsf{T}_i, rl_i, t)}$.

Decryption Key Generation $dk_{\omega,t,i} \leftarrow \mathcal{DK}(sk_{\omega,i}, ku_{t,i})$: given a private key $sk_{\omega,i}$ and a key update $ku_{t,i}$. The receiver generates the decryption key $dk_{\omega,t,i}$ as follows.

1. Parse $sk_{\omega,i}$ as $\{(j, D_j, d_j)\}_{j \in \vec{j}}$, $ku_{t,i}$ as $\{(k, E_k, e_k)\}_{k \in \vec{k}}$ for some set of nodes $\vec{j}, \vec{k}$. If there exists a pair (j, k) s.t. $j = k$, generate the decryption key

$$dk_{\omega,t,i} \leftarrow (D_j, E_k, d_j, e_k).$$

   Otherwise, set the decryption key $dk_{\omega,t,i}$ as $\perp$.
2. Return the decryption key $dk_{\omega,t,i}$.

Encryption $c_i \leftarrow \mathcal{E}(pk_i, \omega, t, m)$: given an identity $\omega \in \mathcal{I}$, a public key pk_i, an encryption time $t \in \mathcal{T}$ and a message $m \in \mathcal{M}$. The sender encrypts the message $m \in \mathcal{M}$ as follows.

1. Randomly choose $z \leftarrow \mathbb{Z}_p$ and generate ciphertext c_1, c_2, c_ω, c_t.

$$c_1 \leftarrow m \cdot e(g_{1,i}, g_{2,i})^z; c_2 \leftarrow g^z;$$
$$c_\omega \leftarrow H_{g_{2,i},\mathsf{J},h_{1,i},h_{2,i},h_{3,i}}(\omega)^z; c_t \leftarrow H_{g_{2,i},\mathsf{J},h_{1,i},h_{2,i},h_{3,i}}(t)^z.$$

2. Return ciphertext $c = (\omega, t, c_1, c_2, c_\omega, c_t)$.

Decryption $m \leftarrow \mathcal{D}(dk_{\omega,t,i}, c)$: given a decryption key $dk_{\omega,t,i}$ and a ciphertext c. The receiver decrypts the ciphertext c as follows:

1. Compute the message m:

$$m \leftarrow c_1 \left(\frac{e(d, c_\omega)}{e(D, c_2)} \right)^{\frac{t}{t-\omega}} \left(\frac{e(e, c_t)}{e(E, c_2)} \right)^{\frac{\omega}{\omega - t}}.$$

2. Return the message m.

Revocation $rl_i \leftarrow \mathcal{R}(\omega, rl_i, st_i, t)$: given an identity to be revoked $\omega \in \mathcal{I}$, a revocation list rl_i, a state st_i and a revocation time $t \in \mathcal{T}$. The key authority updates the revocation list rl_i as follows:

1. For all nodes v associated with identity $\omega \in \mathcal{I}$ add (v, t) to rl as follows:

$$rl_i \leftarrow rl_i \cup (v, t).$$

2. Return the updated revocation list rl_i.

Merge Parameter $(rl_\beta, st_\beta) \leftarrow \mathcal{MP}(\{pk_i, msk_i, rl_i, st_i\}_{i\in\{\alpha,\beta\}})$: given all the system parameters $\{pk_i, mk_i, rl_i, st_i\}_{i\in\{\alpha,\beta\}}$ from system α and system β. The key authority generates system parameters which bases on the system parameters in the system β as follows:

1. Update the revocation list rl_β by uniting two revocation lists rl_α and rl_β.

$$rl_\beta \leftarrow rl_\alpha \cup rl_\beta.$$

2. Update the state st_β as follows:
   (a) Let T_α denote the root node in the binary tree in the system α. Remove the T_α in the state st_α. The detail of the tree structure list in system α is in Fig. 2.

$$st_\alpha \leftarrow st_\alpha \setminus \mathsf{T}_\alpha.$$

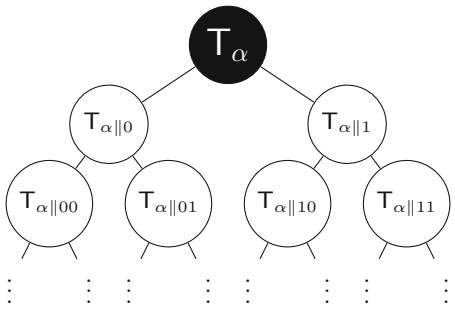

Fig. 2. Tree structure revocation list in System α

   (b) Update the state st_β by uniting it and st_α and the details in the Fig. 3. Note that the new tree is a binary tree except the root node has more than two children.

$$st_\beta \leftarrow st_\alpha \cup st_\beta.$$

3. Return the updated revocation list rl_β and the updated state st_β.

Merge Private Key $sk_{\omega,\alpha,\beta} \leftarrow \mathcal{MSK}(\{pk_i, msk_i, st_i\}_{i\in\{\alpha,\beta\}}, \omega)$: given all the system parameters $\{pk_i, mk_i, st_i\}_{i\in\{\alpha,\beta\}}$ from system α and system β and an identity $\omega \in \mathcal{I}$. The key authority generates the mergeable private key $sk_{\omega,\alpha,\beta}$.

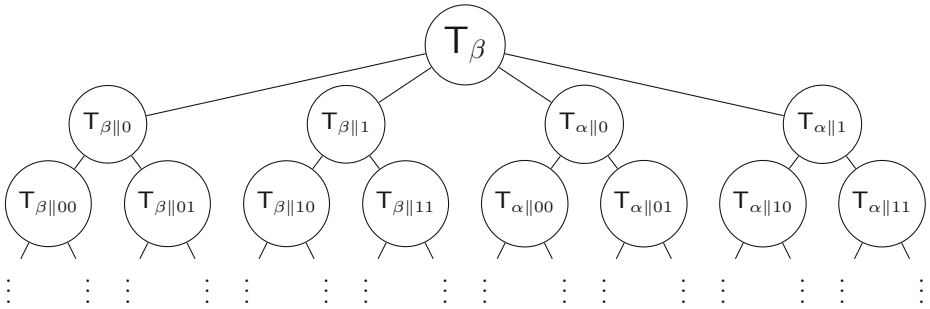

Fig. 3. Tree structure revocation list after merging α and β

1. Parse $pk_i = (g, g_{1,i}, g_{2,i}, h_{1,i}, h_{2,i}, h_{3,i})$ and $mk_i = (a_i, r_i)$ for $i = \alpha, \beta$.
2. Generate system parameters $a_{\alpha,\beta}, g_{1,\alpha,\beta}, g_{2,\alpha,\beta}, h_{1,\alpha,\beta}, h_{2,\alpha,\beta}, h_{3,\alpha,\beta}$ base on the system parameters from the system α and the system β.

$$a_{\alpha,\beta} = a_\beta - a_\alpha . g_{1,\alpha,\beta} = g_{1,\beta}/g_{1,\alpha}; g_{2,\alpha,\beta} = g_{2,\beta}/g_{2,\alpha};$$
$$h_{1,\alpha,\beta} = h_{1,\beta}/h_{1,\alpha}; h_{2,\alpha,\beta} = h_{2,\beta}/h_{2,\alpha}; h_{3,\alpha,\beta} = h_{3,\beta}/h_{3,\alpha}.$$

3. For all node $x \in \mathsf{Path}(v)$ do the following:
   (a) Compute random values $r_{x,\alpha}$, $r_{x,\beta}$ and $r_{x,\alpha,\beta}$.

$$r_{x,\alpha} = \mathsf{F}_{r_\alpha}(\omega \| x); r_{x,\beta} = \mathsf{F}_{r_\beta}(\omega \| x); r_{x,\alpha,\beta} = r_{x,\beta} - r_{x,\alpha}.$$

   (b) Let $a_{x,\alpha}$ and $a_{x,\beta}$ denote the value in node in system α and system β, respectively.
   If x is the root node in system α. Then compute

$$D_{x,\alpha,\beta} = g_{2,\alpha}^{a_{x,\beta}\omega - a_{x,\alpha}\omega + a_{\alpha,\beta}} \cdot g_{2,\alpha,\beta}^{a_{x,\beta}\omega + a_\beta} \cdot H_{g_{2,\alpha},\mathsf{J},h_{1,\alpha},h_{2,\alpha},h_{3,\alpha}}(\omega)^{r_{x,\alpha,\beta}} \cdot H_{g_{2,\alpha,\beta},\mathsf{J},h_{1,\alpha,\beta},h_{2,\alpha,\beta},h_{3,\alpha,\beta}}(\omega)^{r_{x,\beta}}.$$
$$d_{x,\alpha,\beta} = g^{r_{x,\alpha,\beta}}.$$

   Else, compute

$$D_{x,\alpha,\beta} = g_{2,\alpha}^{a_{x,\alpha,\beta}} \cdot g_{2,\alpha,\beta}^{a_{x,\alpha}\omega + a_\beta} \cdot H_{g_{2,\alpha},\mathsf{J},h_{1,\alpha},h_{2,\alpha},h_{3,\alpha}}(\omega)^{r_{x,\alpha,\beta}} \cdot H_{g_{2,\alpha,\beta},\mathsf{J},h_{1,\alpha,\beta},h_{2,\alpha,\beta},h_{3,\alpha,\beta}}(\omega)^{r_{x,\beta}}.$$
$$d_{x,\alpha,\beta} = g^{r_{x,\alpha,\beta}}.$$

   (c) Return $sk_{\omega,\alpha,\beta} = \{(x, D_{x,\alpha,\beta}, d_{x,\alpha,\beta})\}_{x \in \mathsf{Path}(v)}$.

Private Key Update $sk_{\omega,\beta} \leftarrow \mathcal{SKU}(sk_{\omega,\alpha}, sk_{\omega,\alpha,\beta})$: Parse $sk_{\omega,\alpha}$ as $\{(i, D_{i,\alpha}, d_{i,\alpha})\}_{i \in \vec{i}}$ and $\tilde{sk}_{\omega,\alpha,\beta}$ as $\{(j,\ D_{j,\alpha,\beta}, d_{j,\alpha,\beta}\}_{j \in \vec{j}}$ for some set of nodes $\vec{i}$ and $\vec{j}$. The receiver generates updated private key $sk_{\omega,\beta}$ as follows.

1. For $i \in \vec{i}$ do the following:

$$sk_{\omega,\beta} \leftarrow (i, D_{i,\alpha} \cdot D_{i,\alpha,\beta}, d_{i,\alpha} \cdot d_{i,\alpha,\beta})$$

2. Return the private key $sk_{\omega,\beta}$.

5 Security Proof

Definition 5 (Decisional Bilinear Diffie-Hellman). *Let $\mathcal{G}$ be a prime order bilinear group generator. The Decisional Bilinear Diffie-Hellman (DBDH) problem is said to be hard for $\mathcal{G}$ if for every efficient adversary $\mathcal{A}$ its advantage $\mathsf{Adv}^{dbdh}_{\mathcal{G},\mathcal{B}}(k)$ defined as*

$$\Pr[\mathsf{Exp}^{dbdh-real}_{\mathcal{G},\mathcal{A}}(k) = 1] - \Pr[\mathsf{Exp}^{dbdh-rand}_{\mathcal{G},\mathcal{A}}(k) = 1]$$

is a negligible function in k, and where the experiments are as follows:

$\mathsf{Experiment}\ \mathsf{Exp}^{dbdh-real}_{\mathcal{G},\mathcal{A}}(k)$	$\mathsf{Experiment}\ \mathsf{Exp}^{dbdh-rand}_{\mathcal{G},\mathcal{A}}(k)$
$(\mathcal{G}, p, g) \leftarrow \mathcal{G}(k); x, y, z \leftarrow \mathbb{Z}_p$	$(\mathcal{G}, p, g) \leftarrow \mathcal{G}(k); x, y, z, w \leftarrow \mathbb{Z}_p$
$X \leftarrow g^x; Y \leftarrow g^y; Z \leftarrow g^z;$	$X \leftarrow g^x; Y \leftarrow g^y; Z \leftarrow g^z;$
$W \leftarrow e(g,g)^{xyz}$	$W \leftarrow e(g,g)^w$
$d \leftarrow \mathcal{A}(k, \mathcal{G}, p, g, X, Y, Z, W)$	$d \leftarrow \mathcal{A}(k, \mathcal{G}, p, g, X, Y, Z, W)$
Return d	*Return* d

Theorem 1. *Let $\mathcal{G}$ be a prime order bilinear group generator and $\mathcal{MRIBE}$ be the associated mergeable and revocable identity-based encryption scheme proposed above. Then for any adversary $\mathcal{A}$ attacking sMRID security (defined in Sect. 3.2) of $\mathcal{MRIBE}$ with N users in each system and N_S systems, and making q_p private key generation queries, q_r revocation queries, q_k key update generation queries, q_m merge parameter queries and q_{mp} merge private key generation queries, there exists an adversary $\mathcal{B}$ solving DBDH problem for $\mathcal{G}$ such that*

$$\mathsf{Adv}^{smrid-cpa}_{\mathcal{MRIBE},\mathcal{A},N,N_S}(k) \leq 4 \cdot \mathsf{Adv}^{dbdh}_{\mathcal{G},\mathcal{B}}(k)$$

Please refer to Appendix A for the detailed proof.

6 Conclusions

In this paper, we proposed a new variant for Revocalbe Identity Based Encryption. Our proposed scheme allows not only efficient revocation but also efficient merging of multiple systems. Compared with all previous RIBE schemes, our construction does not incur any additional cost. Moreover, the size of the user private key remains unchanged when multiple systems are merged and there is no secure channel required for the purpose of key update during the merging process.

A Security Proof

Proof. The proof is similar to that of [1], except we meed to handle multiple systems and the mergeable algorithms. We construct an adversary $\mathcal{B}$ for the DBDH problem associated with $\mathcal{G}$. $\mathcal{B}$ gets $(k, \mathbb{G}, p, g, X, Y, Z, W)$ as input and it has to return a bit d. It is going to use $\mathcal{A}$. For answering oracles, we define the following four functions. For $i, j, l, r \in \mathbb{Z}_p, S = \{0, j\}$ define

$$F_1(g_2, h_1, h_2, h_3, i, l, r) \stackrel{\text{def}}{=} g_2^l H_{g_2,h_1,h_2,h_3}(i)^r, \quad F_2(r) \stackrel{\text{def}}{=} g^r,$$

$$F_3(g_1, g_2, i, j, l, r) \stackrel{\text{def}}{=} g_2^{l\Delta_{j,S}(i)} \left(g_1^{\frac{-f(i)}{i^2+u(i)}} \left(g_2^{i^2+u(i)} g^{f(i)} \right)^r \right)^{\Delta_{0,S}(i)},$$

$$F_4(g_1, g_2, i, r) \stackrel{\text{def}}{=} \left(g_1^{\frac{-1}{i^2+u(i)}} g^r \right)^{\Delta_{0,S}(i)}.$$

Setup: $\mathcal{B}$ receives the challenging message $(k, \mathbb{G}, p, g, X, Y, Z, W)$ and sets the system parameters as follows.

- $\mathcal{B}$ chooses the $N, N_S \in \mathbb{N}$ and sends the security parameter (k, N, N_S) to $\mathcal{A}$. $\mathcal{A}$ generates the challenging identity ω^*, the challenging time t^*, the subscript of challenging public key i^* and the *state* for some related information about (ω^*, t^*, i^*), then sends $(\omega^*, t^*, i^*, state)$ to $\mathcal{B}$.
- $\mathcal{B}$ chooses a random bit $b \leftarrow \{0, 1\}$ and initializes the database $\mathcal{D}, \mathcal{D}_{\mathcal{SK}}$, $\mathcal{D}_{\mathcal{MSK}} \leftarrow \emptyset$, where $\mathcal{D}$ is used to record the historical information of the challenging identity ω^*, and $\mathcal{D}_{\mathcal{SK}}, \mathcal{D}_{\mathcal{MSK}}$ records information of the challenging identity ω^* to verify whether to abort.
- $\mathcal{B}$ simulates the system parameters for all N_S systems. $\mathcal{B}$ sets public parameter $pp = (\mathbb{G}, p, g)$ and randomly picks a value $i_r \leftarrow \{1, 2, ..., N_S\}$, where the challenging identity ω^* is initialized in the system with public key pk_{i_r}. Then, $\mathcal{B}$ updates the database $\mathcal{D} \leftarrow (\vec{pk}_{\omega^*}, \omega^*)$, where $\vec{pk}_{\omega^*} \leftarrow \vec{pk}_{\omega^*} \cup \{pk_{i_r}\}$. $\forall j \in \{1, 2, ..., N_S\}$ then:
  1. Randomly pick and store $r_j, r_{1,j}, r_{2,j} \leftarrow \mathbb{Z}_p^*$ in the system j and generate the parameters $g_{1,j}$ and $g_{2,j}$.

$$g_{1,j} \leftarrow X^{r_{1,j}}, g_{2,j} \leftarrow Y^{r_{2,j}}.$$

  2. Pick random second-degree polynomials $f(x), u(x)$ with coefficients in $\mathbb{Z}_p$ s.t. $u(x) = -x^2$ for $x = \omega^*, t^*$, o.w. $u(x) \neq -x^2$. $\forall i = \{1, 2, 3\}$ then: set $h_{i,j} \leftarrow g_{2,j}^{u(i)} g^{f(i)}$.
  3. Set the public key $pk_j \leftarrow (g, g_{1,j}, g_{2,j}, h_{1,j}, h_{2,j}, h_{3,j})$.
- $\mathcal{B}$ sends the public parameter pp and public keys $\{pk_i\}_{i \in \{1,2,...,N_S\}}$ to $\mathcal{A}$.
- $\mathcal{B}$ simulates the revocation list and the binary tree. $\forall j = \{1, 2, ..., N_S\}$ then: let rl_j be an empty set and T_j be a binary tree with at least N leaf nodes. $\mathcal{B}$ picks a leaf node v^* from T_{i_r}, where the challenging identity ω^* is assigned to the leaf v^*, and chooses a random bit $rev \leftarrow \{0, 1\}$, where 0 means ω^* is a non-revoked user, otherwise, he is a revoked user.

$\mathcal{O}_{\mathcal{SK}}(pk_i, \omega)$: $\mathcal{A}$ issues up to q_p private key generation queries. $\mathcal{B}$ responds to a query on message (pk_i, ω) as follows.

- If $\omega = \omega^*$, $\mathcal{B}$ simulates the private key $sk_{\omega,i}$ for the challenging identity ω^*.
  1. If $rev = 0$, set $\mathcal{D}_{\mathcal{SK}} \leftarrow \mathcal{D}_{\mathcal{SK}} \cup \{pk_i\}$ and abort if $\mathcal{A}$ is able to obtain the secret key sk_{ω^*,i^*} by checking the transactions in database $\mathcal{D}_{\mathcal{SK}}$ and $\mathcal{D}_{\mathcal{MSK}}$ in Fig. 1.
  2. Else set $v \leftarrow v^*$. $\forall x \in \mathsf{Path}(v)$ then:
     (a) Set $r_x \leftarrow \mathsf{F}_{r_i}(\omega^* \| x)$, where $msk_i = (a_i, r_i)$.
     (b) If $\nexists l_x$ then randomly choose $l_x \leftarrow \mathbb{Z}_p$ and store l_x in node x.
     (c) Set (D_x, d_x) and update private key $sk_{\omega^*,i} \leftarrow sk_{\omega^*,i} \cup (x, D_x, d_x)$.

$$D_x \leftarrow F_1(g_{2,i}, h_{1,i}, h_{2,i}, h_{3,i}, \omega^*, l_x, r_x), d_x \leftarrow F_2(r_x).$$

- If $\omega \neq \omega^*$, $\mathcal{B}$ simulates the private key $sk_{\omega,i}$ for the identity ω. $\forall x \in \mathsf{Path}(v)$ then:
  1. Set $r_x \leftarrow \mathsf{F}_{r_i}(\omega \| x)$, where $msk_i = (a_i, r_i)$.
  2. If $\nexists l_x$ then randomly choose $l_x \leftarrow \mathbb{Z}_p$ and store l_x in node x.
  3. If $rev = 0$, set (D_x, d_x) and update private key $sk_{\omega,i} \leftarrow sk_{\omega,i} \cup (x, D_x, d_x)$.

$$D_x \leftarrow F_3(g_{1,i}, g_{2,i}, \omega, t^*, l_x, r_x), d_x \leftarrow F_4(g_{1,i}, g_{2,i}, \omega, r_x).$$

  4. If $rev = 1$, simulate the private key $sk_{\omega,i}$ depends on the $\mathsf{Path}(v)$ and $\mathsf{Path}(v^*)$.
     (a) $\forall x \in (\mathsf{Path}(v) \setminus \mathsf{Path}(v^*))$ then: set (D_x, d_x) and update private key $sk_{\omega,i} \leftarrow sk_{\omega,i} \cup (x, D_x, d_x)$.

$$D_x \leftarrow F_3(g_{1,i}, g_{2,i}, \omega, t^*, l_x, r_x), d_x \leftarrow F_4(g_{1,i}, g_{2,i}, \omega, r_x).$$

     (b) $\forall x \in (\mathsf{Path}(v) \cap \mathsf{Path}(v^*))$ then: set (D_x, d_x) and update private key $sk_{\omega,i} \leftarrow sk_{\omega,i} \cup (x, D_x, d_x)$.

$$D_x \leftarrow F_3(g_{1,i}, g_{2,i}, \omega, \omega^*, l_x, r_x), d_x \leftarrow F_4(g_{1,i}, g_{2,i}, \omega, r_x).$$

- Return the private key $sk_{\omega,i} = \{(x, D_x, d_x)\}_{x \in \mathsf{Path}(v)}$.

$\mathcal{O}_{\mathcal{R}}(\omega, t)$: $\mathcal{A}$ issues up to q_r revocation queries. $\mathcal{B}$ responds to a query on message (ω, t) as follows. If $(\cdot, \omega) \in \mathcal{D}$, for all leaf nodes v associated with identity ω add (v, t) to revocation list $rl_i \leftarrow rl_i \cup (v, t)$, then return rl_i else return $\perp$.

$\mathcal{O}_{\mathcal{KU}}(pk_i, t)$: $\mathcal{A}$ issues up to q_k key update generation queries. $\mathcal{B}$ responds to a query on message (pk_i, t) as follows.

- If $t \neq t^*$, $\mathcal{B}$ simulates the key update $ku_{t,i}$ for the system i.
  1. If $rev = 0$, $\forall x \in \mathsf{KUNodes}(\mathsf{T}, rl, t)$ then: $r_x \leftarrow \mathbb{Z}_p^*$, set E_x and d_x and update $ku_{t,i} \leftarrow ku_{t,i} \cup (x, E_x, e_x)$.

$$E_x \leftarrow F_3(g_{1,i}, g_{2,i}, t, t^*, l_x, r_x), e_x \leftarrow F_4(g_{1,i}, g_{2,i}, t, r_x).$$

2. If $rev = 1$, simulate the key ku_t depends on the $\mathsf{Path}(v)$ and $\mathsf{Path}(v^*)$.
   (a) $\forall x \in (\mathsf{KUNodes}(\mathsf{T}, rl, t) \setminus \mathsf{Path}(v^*))$ then: $r_x \leftarrow \mathbb{Z}_p^*$, set E_x and d_x and update $ku_{t,i} \leftarrow ku_{t,i} \cup (x, E_x, e_x)$.

$$E_x \leftarrow F_3(g_{1,i}, g_{2,i}, t, t^*, l_x, r_x), e_x \leftarrow F_4(g_{1,i}, g_{2,i}, t, r_x).$$

   (b) $\forall x \in (\mathsf{KNodes}(\mathsf{T}, rl, t) \cap \mathsf{Path}(v^*))$ then: $r_x \leftarrow \mathbb{Z}_p^*$, set E_x and d_x and update $ku_{t,i} \leftarrow ku_{t,i} \cup (x, E_x, e_x)$.

$$E_x \leftarrow F_3(g_{1,i}, g_{2,i}, t, \omega^*, l_x, r_x), e_x \leftarrow F_4(g_{1,i}, g_{2,i}, t, r_x).$$

– If $t = t^*$, $\mathcal{B}$ simulates the key update $ku_{t,i}$ in the challenging time t^* for the system i.
  1. If $rev = 1$ and $\forall t \leq t^*$ we have that $(\omega^*, t) \notin rl_{i^*}$ then abort since challenging identity ω^* must be revoked when $rev = 1$.
  2. Else, $\forall x \in \mathsf{KUNodes}(\mathsf{T}, rl, t)$ then: $r_x \leftarrow \mathbb{Z}_p^*$, set E_x and d_x and update $ku_{t,i} \leftarrow ku_{t,i} \cup (x, E_x, e_x)$.

$$E_x \leftarrow F_1(g_{2,i}, h_{1,i}, h_{2,i}, h_{3,i}, t^*, l_x, r_x), e_x \leftarrow F_2(r_x).$$

– Return the key update $ku_{t,i} = \{(x, E_x, e_x)\}_{x \in \mathsf{KUNodes}(\mathsf{T}, rl, t)}$.

$\mathcal{O}_{\mathcal{MP}}(pk_\alpha, pk_\beta)$: $\mathcal{A}$ issues up to q_m merge parameter generation queries. $\mathcal{B}$ responds to a query on message (pk_α, pk_β) by updating the revocation list rl_β, state st_β and the database $\mathcal{D}$ as follows.

– Update the revocation list and state $rl_\beta \leftarrow rl_\alpha \cup rl_\beta, st_\beta \leftarrow st_\beta \cup st_\alpha \setminus \mathsf{T}_\alpha$.
– If ω^* is involved in the system with pk_α, then updating the database $\mathcal{D}$. $\forall(\vec{pk}, \cdot) \in \mathcal{D}$ then set $len = |\vec{pk}|$, if $pk[len] = pk_\alpha$, $\vec{pk} \leftarrow \vec{pk} \cup pk_\beta$.
– Return the updated revocation list rl_β and state st_β.

$\mathcal{O}_{\mathcal{MSK}}(\omega, pk_\alpha, pk_\beta)$: $\mathcal{A}$ issues up to q_{mp} merge private key generation queries. $\mathcal{B}$ responds to a query on message $(\omega, pk_\alpha, pk_\beta)$.

– If $\omega = \omega^*$, $\mathcal{B}$ simulates the private key $sk_{\omega,\alpha,\beta}$ for challenging identity ω.
  1. If $rev = 0$, set $\mathcal{D}_{\mathcal{MSK}} \leftarrow \mathcal{D}_{\mathcal{MSK}} \cup \{(pk_\alpha, pk_\beta)\}$ and abort if $\mathcal{A}$ is able to obtain the secret key sk_{ω^*,i^*} by checking the transactions in database $\mathcal{D}_{\mathcal{SK}}$ and $\mathcal{D}_{\mathcal{MSK}}$ in Fig. 1.
  2. Else set $v \leftarrow v^*$. $\forall x \in \mathsf{Path}(v)$ then:
     (a) Set $r_{x,\alpha} \leftarrow G_{r_\alpha}(\omega \| x)$ and $r_{x,\beta} \leftarrow \mathsf{F}_{r_\beta}(\omega \| x)$.
     (b) Set $(D_{x,\alpha,\beta}, d_{x,\alpha,\beta})$ and update private key $sk_{\omega,\alpha,\beta} \leftarrow sk_{\omega,\alpha,\beta} \cup (x, D_{x,\alpha,\beta}, d_{x,\alpha,\beta})$, where the union symbol is used to combine the secret keys since this algorithm will return secret keys belong to $\mathsf{Path}(v)$.

$$D_{x,\alpha,\beta} = \frac{F_1(g_{2,\beta}, h_{1,\beta}, h_{2,\beta}, h_{3,\beta}, \omega, l_{x,\beta}, r_{x,\beta})}{F_1(g_{2,\alpha}, h_{1,\alpha}, h_{2,\alpha}, h_{3,\alpha}, \omega, l_{x,\alpha}, r_{x,\alpha})}, d_{x,\alpha,\beta} = \frac{F_2(r_{x,\beta})}{F_2(r_{x,\alpha})}.$$

– If $\omega \neq \omega^*$, $\mathcal{B}$ simulates the private key $sk_{\omega,\alpha,\beta}$ for the identity ω. $\forall x \in \mathsf{Path}(v)$ then:
  1. Set $r_{x,\alpha} \leftarrow G_{r_\alpha}(\omega \| x)$ and $r_{x,\beta} \leftarrow G_{r_\beta}(\omega \| x)$.
  2. If $rev = 0$, set (D_x, d_x) and update private key $sk_\omega \leftarrow sk_\omega \cup (x, D_x, d_x)$, where the union symbol is used for the same reason in previous section.

$$D_{x,\alpha,\beta} = \frac{F_3(g_{1,\beta}, g_{2,\beta}, \omega, t^*, l_{x,\beta}, r_{x,\beta})}{F_3(g_{1,\alpha}, g_{2,\alpha}, \omega, t^*, l_{x,\alpha}, r_{x,\alpha})}, d_{x,\alpha,\beta} = \frac{F_4(g_{1,\beta}, g_{2,\beta}, \omega, r_{x,\beta})}{F_4(g_{1,\alpha}, g_{2,\alpha}, \omega, r_{x,\alpha})}.$$

  3. If $rev = 1$, simulate the private key sk_ω depends on the $\mathsf{Path}(v)$ and $\mathsf{Path}(v^*)$.
     (a) $\forall x \in (\mathsf{Path}(v) \setminus \mathsf{Path}(v^*))$ then: set (D_x, d_x) and update private key $sk_\omega \leftarrow sk_\omega \cup (x, D_x, d_x)$.

$$D_{x,\alpha,\beta} = \frac{F_3(g_{1,\beta}, g_{2,\beta}, \omega, t^*, l_{x,\beta}, r_{x,\beta})}{F_3(g_{1,\alpha}, g_{2,\alpha}, \omega, t^*, l_{x,\alpha}, r_{x,\alpha})}, d_{x,\alpha,\beta} = \frac{F_4(g_{1,\beta}, g_{2,\beta}, \omega, r_{x,\beta})}{F_4(g_{1,\alpha}, g_{2,\alpha}, \omega, r_{x,\alpha})}.$$

     (b) $\forall x \in (\mathsf{Path}(v) \cap \mathsf{Path}(v^*))$ then: set (D_x, d_x) and update private key $sk_\omega \leftarrow sk_\omega \cup (x, D_x, d_x)$.

$$D_{x,\alpha,\beta} = \frac{F_3(g_{1,\beta}, g_{2,\beta}, \omega, \omega^*, l_{x,\beta}, r_{x,\beta})}{F_3(g_{1,\alpha}, g_{2,\alpha}, \omega, \omega^*, l_{x,\alpha}, r_{x,\alpha})}, d_{x,\alpha,\beta} = \frac{F_4(g_{1,\beta}, g_{2,\beta}, \omega, r_{x,\beta})}{F_4(g_{1,\alpha}, g_{2,\alpha}, \omega, r_{x,\alpha})}.$$

  4. Return the private key $sk_{\omega,\alpha,\beta} = \{(x, D_{x,\alpha,\beta}, d_{x,\alpha,\beta})\}_{x \in \mathsf{Path}(v)}$.

Output: $\mathcal{A}$ outputs two message m_0 and m_1. $\mathcal{B}$ picks a random bit $b \leftarrow \{0,1\}$ and generates the challenging ciphertext $c^* = (c_1^*, c_2^*, c_{\omega^*}, c_{t^*})$ and then sends c^* to $\mathcal{A}$. $\mathcal{A}$ outputs a bit d. If $b = d$, $\mathcal{B}$ outputs 1 else output 0.

$$c_1^* = m_b \cdot W^{r_{1,\omega^*} \cdot r_{2,\omega^*}}, c_2^* = Z, c_{\omega^*} = Z^{f(\omega^*)}, c_{t^*} = Z^{f(t^*)}.$$

If any oracles abort, $\mathcal{B}$ outputs 1.

A.1 Analysis

Let $\mathsf{sreal}, \mathsf{srand}$ denote the events that none of the oracles abort in $\mathsf{Exp}_{\mathcal{G},\mathcal{B}}^{dbdh-real}(k)$, $\mathsf{Exp}_{\mathcal{G},\mathcal{B}}^{dbdh-rand}(k)$ respectively. Then

$$\Pr[\mathsf{sreal}] = \Pr[\mathsf{srand}] \geq 1/2.$$

The probability that $\mathcal{O}_{\mathcal{SK}}(pk_i, \omega)$, $\mathcal{O}_{\mathcal{MSK}}(\omega, pk_\alpha, pk_\beta)$ and $\mathcal{O}_{\mathcal{KU}}(pk_i, t)$ oracles abort depends on the bit rev which are chosen independently from whether $\mathcal{B}$ is in $\mathsf{Exp}_{\mathcal{G},\mathcal{B}}^{dbdh-real}(k)$ or $\mathsf{Exp}_{\mathcal{G},\mathcal{B}}^{dbdh-rand}(k)$. So, $\Pr[\mathsf{sreal}] = \Pr[\mathsf{srand}]$.

$\mathcal{O}_{\mathcal{SK}}(pk_i, \omega)$ and $\mathcal{O}_{\mathcal{MSK}}(\cdot, \cdot, \omega^*)$ oracles can be queried on ω^* without constrain only if $\mathcal{O}_{\mathcal{R}}(\omega, t)$ oracle was queried on (ω^*, t) for any $t \leq t^*$. Thus, we have

$$\begin{aligned}
&\Pr[\omega = \omega^*] \leq \Pr[(\omega^*, t) \in rl_{\omega^*}, \forall t \leq t^*] \\
\Rightarrow\ &1 - \Pr[\omega = \omega^*] \geq \Pr[(\omega^*, t) \notin rl_{\omega^*}, \forall t \leq t^*] \\
\Rightarrow\ &1 - \Pr[\omega = \omega^*] \geq \Pr[(t = t^*) \wedge (\omega^*, t) \notin rl_{\omega^*}, \forall t \leq t^*]
\end{aligned}$$

We see that $\mathcal{O}_{\mathcal{SK}}(pk_i, \omega)$ oracles abort if $\omega = \omega^*$ and $\mathcal{O}_{\mathcal{KU}}(t)$ oracle aborts if $rev = 1, t = t^*$ and $\exists t \leq t^*\ (\omega^*, t) \notin rl_{\omega^*}$. Thus,

$$\begin{aligned}
\Pr[\overline{\mathsf{sreal}}] &= \Pr[(rev = 0) \wedge (\omega = \omega^*)] \\
&\quad + \Pr[(rev = 1) \wedge (t = t^*) \wedge ((\omega^*, t) \notin rl_{\omega^*}, \forall t \leq t^*)] \\
&= \Pr[rev = 0] \cdot \Pr[\omega = \omega^*] \\
&\quad + \Pr[rev = 1] \cdot \Pr[(t = t^*) \wedge ((\omega^*, t) \notin rl_{\omega^*}, \forall t \leq t^*)] \\
&\leq 1/2 \cdot \Pr[\omega = \omega^*] + \frac{1}{2}(1 - \Pr[\omega = \omega^*]) \leq 1/2
\end{aligned}$$

$\mathcal{B}$ simulates the exact experiment $\mathsf{Exp}^{smrid-cpa}_{\mathcal{MRIBE},\mathcal{A},N,N_S}(k)$ for $\mathcal{A}$ when $\mathcal{B}$ is in $\mathsf{Exp}^{dbdh-real}_{\mathcal{G},\mathcal{B}}(k)$ and none of the oracles abort. So,

$$\Pr\left[\mathsf{Exp}^{dbdh-real}_{\mathcal{G},\mathcal{B}}(k) = 1 | \mathsf{sreal}\right] \geq \Pr\left[\mathsf{Exp}^{smrid-cpa}_{\mathcal{MRIBE},\mathcal{A},N,N_S}(k) = 1\right].$$

When $\mathcal{B}$ is $\mathsf{Exp}^{dbdh-rand}_{\mathcal{G},\mathcal{B}}(k)$ and none of the oracles abort then as explained earlier bit b is information-theoretically hidden from $\mathcal{A}$. So,

$$\Pr\left[\mathsf{Exp}^{dbdh-rand}_{\mathcal{G},\mathcal{B}}(k) = 1 | \mathsf{srand}\right] \leq 1/2.$$

Also, since $\mathcal{B}$ outputs 1 when either of the oracles aborts, so

$$\begin{aligned}
\Pr\left[\mathsf{Exp}^{dbdh-real}_{\mathcal{G},\mathcal{B}}(k) = 1 | \overline{\mathsf{sreal}}\right] &= 1, \\
\Pr\left[\mathsf{Exp}^{dbdh-rand}_{\mathcal{G},\mathcal{B}}(k) = 1 | \overline{\mathsf{srand}}\right] &= 1.
\end{aligned}$$

Thus,

$$\begin{aligned}
\mathsf{Adv}^{dbdh}_{\mathcal{G},\mathcal{B}}(k) &= \Pr\left[\mathsf{Exp}^{dbdh-real}_{\mathcal{G},\mathcal{B}}(k) = 1\right] - \Pr\left[\mathsf{Exp}^{dbdh-rand}_{\mathcal{G},\mathcal{B}}(k) = 1\right] \\
&\geq 1/2 \cdot \left(\Pr\left[\mathsf{Exp}^{dbdh-real}_{\mathcal{G},\mathcal{B}}(k) = 1 | \mathsf{sreal}\right] - \frac{1}{2}\right) \\
&\geq 1/2 \cdot \frac{1}{2} \cdot \left(2 \cdot \Pr\left[\mathsf{Exp}^{dbdh-real}_{\mathcal{G},\mathcal{B}}(k) = 1 | \mathsf{sreal}\right] - 1\right) \\
&\geq 1/2 \cdot \mathsf{Adv}^{smrid-cpa}_{\mathcal{MRIBE},\mathcal{A},N,N_S}(k).
\end{aligned}$$

References

1. Boldyreva, A., Goyal, V., Kumar, V.: Identity-based encryption with efficient revocation. In: Ning, P., Syverson, P.F., Jha, S. (eds.) CCS, pp. 417–426. ACM (2008)
2. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24676-3_14
3. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). doi:10.1007/3-540-44647-8_13
4. Dodis, Y., Fazio, N.: Public key broadcast encryption for stateless receivers. In: Feigenbaum, J. (ed.) DRM 2002. LNCS, vol. 2696, pp. 61–80. Springer, Heidelberg (2003). doi:10.1007/978-3-540-44993-5_5
5. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006). doi:10.1007/11761679_27
6. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: FOCS, pp. 464–479. IEEE (1984)
7. Halevy, D., Shamir, A.: The LSD broadcast encryption scheme. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 47–60. Springer, Heidelberg (2002). doi:10.1007/3-540-45708-9_4
8. Hanaoka, Y., Hanaoka, G., Shikata, J., Imai, H.: Identity-based hierarchical strongly key-insulated encryption and its application. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 495–514. Springer, Heidelberg (2005). doi:10.1007/11593447_27
9. Lee, K., Lee, D.H., Park, J.H.: Efficient revocable identity-based encryption via subset difference methods. IACR, 2014:132 (2014)
10. Liang, K., Liu, J.K., Wong, D.S., Susilo, W.: An efficient cloud-based revocable identity-based proxy re-encryption scheme for public clouds data sharing. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8712, pp. 257–272. Springer, Cham (2014). doi:10.1007/978-3-319-11203-9_15
11. Libert, B., Vergnaud, D.: Adaptive-ID secure revocable identity-based encryption. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 1–15. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00862-7_1
12. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001). doi:10.1007/3-540-44647-8_3
13. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). doi:10.1007/11426639_27
14. Seo, J.H., Emura, K.: Efficient delegation of key generation and revocation functionalities in identity-based encryption. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 343–358. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36095-4_22
15. Seo, J.H., Emura, K.: Revocable identity-based encryption revisited: security model and construction. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 216–234. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36362-7_14
16. Seo, J.H., Emura, K.: Adaptive-ID secure revocable hierarchical identity-based encryption. In: Tanaka, K., Suga, Y. (eds.) IWSEC 2015. LNCS, vol. 9241, pp. 21–38. Springer, Cham (2015). doi:10.1007/978-3-319-22425-1_2

17. Seo, J.H., Emura, K.: Revocable hierarchical identity-based encryption: history-free update, security against insiders, and short ciphertexts. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 106–123. Springer, Cham (2015). doi:10.1007/978-3-319-16715-2_6
18. Shamir, A.: How to share a secret. ACM **22**(11), 612–613 (1979)
19. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985). doi:10.1007/3-540-39568-7_5
20. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005). doi:10.1007/11426639_7
21. Yang, Y., Liu, J.K., Liang, K., Choo, K.-K.R., Zhou, J.: Extended proxy-assisted approach: achieving revocable fine-grained encryption of cloud data. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9327, pp. 146–166. Springer, Cham (2015). doi:10.1007/978-3-319-24177-7_8