An Efficient and Provable Secure Revocable Identity-Based Encryption Scheme

Changji Wang^{1,2*}, Yuan Li^{1,2}, Xiaonan Xia^{1,2}, Kangjia Zheng^{1,2}

1 School of Information Science and Technology, Sun Yat-sen University, Guangzhou, China, **2** Guangdong Province Information Security Key Laboratory, Sun Yat-sen University, Guangzhou, China

Abstract

Revocation functionality is necessary and crucial to identity-based cryptosystems. Revocable identity-based encryption (RIBE) has attracted a lot of attention in recent years, many RIBE schemes have been proposed in the literature but shown to be either insecure or inefficient. In this paper, we propose a new scalable RIBE scheme with decryption key exposure resilience by combining Lewko and Waters' identity-based encryption scheme and complete subtree method, and prove our RIBE scheme to be semantically secure using dual system encryption methodology. Compared to existing scalable and semantically secure RIBE schemes, our proposed RIBE scheme is more efficient in term of ciphertext size, public parameters size and decryption cost at price of a little looser security reduction. To the best of our knowledge, this is the first construction of scalable and semantically secure RIBE scheme with constant size public system parameters.

Citation: Wang C, Li Y, Xia X, Zheng K (2014) An Efficient and Provable Secure Revocable Identity-Based Encryption Scheme. PLoS ONE 9(9): e106925. doi:10.1371/journal.pone.0106925

Editor: Cheng-Yi Xia, Tianjin University of Technology, China

Received: April 15, 2014; **Accepted:** August 1, 2014; **Published:** September 19, 2014

Copyright: © 2014 Wang et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Data Availability: The authors confirm that all data underlying the findings are fully available without restriction. All relevant data are within the paper.

Funding: This paper is jointly supported by the National Natural Science Foundation of China (Grant No. 61173189) and Guangdong Province Information Security Key Laboratory Project. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: The authors have declared that no competing interests exist.

* Email: isschwj@mail.sysu.edu.cn

Introduction

Shamir [1] first introduced the concept of identity-based public key cryptography (ID-PKC) where a public key can be an arbitrary string such as an email address or a telephone number, while the corresponding private key can only be generated by a private key generator (PKG) who has the knowledge of the master secret. The first secure and practical identity-based encryption (IBE) scheme was proposed by Boneh and Franklin [2] from bilinear pairings, which is proved to be semantically secure against adaptive chosen ciphertext attack (IND-ID-CCA) under the Decisional Bilinear Diffie-Hellman (DBDH) assumption in the random oracle model.

Boneh and Franklin's work spurred a great deal of research on IBE. One important research direction is to construct provably secure IBE schemes in the standard model, because random oracle model only provides heuristic security [3]. Canetti, Halevi, and Katz [4] defined a weaker security notion for IBE, known as selective-ID model, in which the adversary commits ahead of time to the identity that it intends to attack. Boneh and Boyen [5] proposed two efficient IBE schemes that are secure in the selective-ID model without random oracle. The first IBE construction (BB1-IBE) is based on the DBDH assumption, while the second IBE construction (BB2-IBE) is based on a non-standard Decision Bilinear Diffie-Hellman Inversion (DBDHI) assumption. Waters [6] improved BB1-IBE scheme and proposed an efficient IBE scheme which is proved to be semantically secure without random oracles under the DBDH assumption in adaptive-ID model. Gentry [7] presented an IBE scheme with short public parameters which is proved to be semantically secure without random oracles

under a non-static assumption in adaptive-ID model. Waters [8] introduced a new technique called dual system encryption and proposed an IBE scheme that is proved to be semantically secure without random oracle under standard (static) assumption in adaptive-ID model. Recently, Lewko and Waters [9] gave a new dual system encryption realization of IBE from composite order bilinear groups, which is proved to be semantically secure without random oracle under the subgroup decision assumption in adaptive-ID model.

Another important research direction is to construct IBE schemes with efficient revocation. Suppose that Alice has left the organization or her private key is compromised or stolen by an adversary in some scenarios [10]. On the one hand, Alice will be withdrawn from the right of accessing the information with respect to her public key. On the other hand, Alice's private key will be revoked to prevent the adversary with her compromised private key to access confidential data encrypted under her public key. Thus, revocation functionality is necessary and crucial to public-key cryptosystems. In the public key infrastructure setting, numerous solutions have been proposed, such as periodic publication mechanisms (e.g. certificate revocation list) and online query mechanisms (e.g. online certificate status protocol). In the ID-PKC setting, however, key revocation is non-trivial. This is because a user's identity is itself a public key, thus one can not simply change her public key, as this changes her identity as well. An ideal revocation method for IBE is that a sender can generate a ciphertext as the same as that of IBE without worrying about the revocation of a receiver and only the receiver needs to check the revocation of his private key to decrypt the ciphertext.

Revocable IBE (RIBE) has attracted a lot of attention in recent years, many RIBE schemes have been proposed [2,11–15]. Boneh and Franklin [2] proposed a trivial method to achieve revocation functionality for IBE (BF-RIBE for short) by representing an identity as $ID\|T$ where ID is the real identity and T is a current time. Since new decryption keys are needed to be issued by the PKG for each time period, this introduces huge overheads for PKG that are linearly increased in the number of users and a secure channel is needed between PKG and users to transmit updated private key. Thus, BF-RIBE is not scalable.

Boldyreva et al. [11] proposed the first scalable RIBE scheme (BGK-RIBE for short) by combining Sahai and Waters' fuzzy IBE scheme [16] and Naor et al.'s complete subtree method [17], where the PKG's overhead increases logarithmically (instead of linearly) in the number of users. The idea of BGK-RIBE scheme consists in assigning users to the leaves of a complete binary tree. Each user is provided by PKG with a set of private keys sk_{ID} corresponding to his/her identity ID for each node on the path from his/her associated leaf to the root of the tree via a secure channel as in IBE scheme. PKG broadcasts key updates ku_T in each time period T for a set Y of nodes that contains no ancestors of revoked users and exactly one ancestor of any non-revoked one (as illustrated in Figure 1 where the nodes of Y are the squares). Then, a user assigned to leaf η is able to form an effective decryption key $dk_{ID,T}$ for period T if the set Y contains a node on the path from the root to η . By doing so, every update of the revocation list RL only requires PKG to perform logarithmic work in the overall number of users and no secure channel is required between PKG and users. The size of users' private keys also logarithmically depends on the maximal number of users.

Another idea of BGK-RIBE scheme consists in applying fuzzy IBE primitive. In fuzzy IBE systems, identities are regarded as sets of descriptive attributes instead of a single identity string in IBE systems, and a user with private key for the attribute set $\vec{w}$ is able to decrypt a ciphertext encrypted for an attribute set $\vec{w}'$ if and only if $\vec{w}$ and $\vec{w}'$ have an overlap of at least d attributes. The BGK-RIBE scheme uses a special kind of fuzzy IBE where ciphertexts are encrypted using the receiver's identity and the period number as "attributes". The decryption key of the receiver has to match both attributes to decrypt the ciphertext. For each node on the path from the root to its assigned leaf, the user is given a key attribute that is generated using a new polynomial with degree 1 for which the constant term is always the master secret. The same polynomials are used, for each node, to generate key updates. To

compute a decryption key for period T , each user thus needs to combine two key attributes associated with the same node of the tree. Since there is no adaptive-ID secure fuzzy IBE scheme in the literature, BGK-RIBE scheme [11] is only proved to be secure in selective-ID model.

Later, Libert and Vergnaud [12] proposed the first adaptive-ID secure scalable RIBE scheme (LV-RIBE for short) based on same idea as BGK-RIBE scheme, but, instead of using fuzzy IBE scheme, they applied the idea of two-level hierarchical IBE scheme (HIBE for short). They use adaptive-ID secure Libert and Vergnaud's black-box accountable authority IBE scheme [18] in the first level to handle user's long term private keys (associated with identities), and use selective-ID secure Boneh and Boyen's BB1-IBE scheme [5] in the second level to handle decryption keys (associate with time periods). Seo and Emura [13] refined the security model of RIBE by considering the decryption key exposure attacks, and proposed a scalable RIBE scheme (SE-RIBE for short) with decryption key exposure resistance based on same idea as LV-RIBE scheme. Seo and Emura use adaptive-ID secure Waters IBE scheme [6] in the first level to handle user's long term private keys, and use selective-ID secure BB1-IBE scheme [5] in the second level to handle decryption keys. Recently, Park et al. [14] proposed a scalable RIBE scheme with shorter private key and update key by using multilinear maps, but the size of the public parameters is dependent to the number of users. Lee et al. [15] presented a new technique for RIBE that uses the subset difference method instead of using the complete subtree method to improve the size of update keys.

Existing adaptive-ID secure scalable RIBE constructions are built on combining two-level HIBE schemes and complete subtree method, and proved security with partition strategy in which the space of identities is partitioned into the set of identities for which a valid secret key can be simulated and those for which a valid challenge ciphertext can be simulated.

In this paper, we propose an efficient adaptive-ID secure scalable RIBE scheme by combining two-level Lewko and Waters HIBE scheme [9] and complete subtree method. To prove security for our RIBE scheme in adaptive-ID model, we adopt Waters dual system encryption methodology [8]. However, we can not use dual system encryption methodology directly to prove the security of RIBE schemes. This is because an adversary in RIBE schemes can issue private key query for the challenge identity ID^* as long as ID^* has been revoked before the challenge time T^* , while an adversary in IBE schemes can not issue private key query for the

Figure 1. Example of KUNode Algorithm. Assume that the user associated with node 9 is revoked. As figure illustrated, user assigned to leaf node 7 has subkeys of node 7, 3, 1 and root. In time period T , only user assigned to leaf node 9 is revoked, the square nodes are update nodes set outputted by the KUNode algorithm, it's obvious that this set does not contain any node on the path from node 9 to root node.

doi:10.1371/journal.pone.0106925.g001

challenge identity ID^* . Furthermore, as stated in Seo and Emura [13], an adversary in a scalable RIBE scheme with decryption key exposure resistance may obtain not a private key sk_{ID^*} but a decryption key $dk_{ID^*,T}$, and ID^* can still be alive in the system in the challenge time period $T^* \neq T$.

To make dual system encryption methodology work properly, we need to make sure that all decryption keys, including those generated by the adversary, are semi-functional in the last step. It is not a trivial job to accomplish this transformation directly. To circumvent this issue, our approach is to design semi-functional private key and semi-functional update key, and generate a semi-functional decryption key from a semi-functional private key or a semi-functional update key.

During registration, PKG assigns a user with identity ID to a leaf node η of a complete binary tree, and issues the private key sk_{ID} for identity ID which is composed by a set of subkeys $sk_{ID,\theta} = \{(K_1, K_2, K_3)\}_{\theta \in \text{Path}(\eta)}$, wherein each subkey is associated with a node on $\text{Path}(\eta)$. At time period T , PKG broadcasts the update key ku_T which is composed by a set of subkeys $ku_{T,\theta} = \{(U_1, U_2, U_3)\}_{\theta \in Y}$, wherein each subkey is associated with a node in Y . An intuitive way to make all decryption keys be semi-functional in the last step is to transform all subkeys of all private keys or all subkeys of update keys from normal form into semi-functional form. However, similar to the security proof in Lewko and Waters' IBE scheme [9], the adversary cannot issue private key query for identities which are equal to the challenge identity ID^* modulo p_2 , and cannot issue update key query for time periods which are equal to the challenge time T^* modulo p_2 , namely all subkeys of these private keys and all subkeys of these update keys can not be transformed. On the one hand, if we transform either all subkeys of the corresponding private key sk_{ID} satisfying $ID \neq ID^* \bmod p_2$ or all subkeys of the corresponding update key ku_T satisfying $T \neq T^* \bmod p_2$ from normal form into semi-functional form independently, the resulting decryption keys $dk_{ID,T}$ may not be semi-functional. On the other hand, if we transform all subkeys of the corresponding update key ku_T from normal form into semi-functional form, this will result in security degradation $O(r \log(N/r))$ when $r \leq N/2$, and security degradation about $O(N-r)$ when $r > N/2$, where N is the number of users and r is the number of revoked users.

To solve the problem of security degradation, we take advantage of the special structure of complete subtree method. We do not need to transform all subkeys of sk_{ID} that satisfy $ID \neq ID^* \bmod p_2$ and all subkeys of ku_T that satisfy $T \neq T^* \bmod p_2$ from normal form into semi-functional form, we just need to transform subkeys of above sk_{ID} that satisfy $\theta \in \text{Path}(\eta) \wedge \theta \notin \text{Path}(\eta^*)$, and subkeys of above ku_T that satisfy $\theta' \in Y \wedge \theta' \notin \text{Path}(\eta^*)$ from norm form into semi-functional form, where η and η^* are leaf nodes of binary tree that assigned to ID and ID^* , respectively. Thus, security degradation is reduced to $O(1)$ per transformation of a update key.

Compared to existing adaptive-ID secure scalable RIBE schemes, our RIBE scheme is more efficient in term of ciphertext size, public parameters size and decryption cost at price of a little looser security reduction. To the best of our knowledge, this is the first construction of scalable semantically secure RIBE scheme with constant size public system parameters. Table 1 shows a comparison between our RIBE scheme and existing RIBE schemes.

The rest of the paper is organized as follows. In Section 2, we introduce some preliminary works necessary for our constructions, such as bilinear group generator and complexity assumptions. In Section 3, we give formal syntax and security definitions of RIBE. In Section 4, we describe our RIBE construction. In Section 5, we

Table 1. Comparison among RIBE schemes.

RIBE schemes	Security model	Complexity assumption	Scalability	DKE resistance	CT size	Dec. cost	Mpk size
BF-RIBE [2]	Adaptive RO	DBDH	No	Yes	$ G + P $	t_e	$3 G $
BGK-RIBE [11]	Selective RO	DBDH	Yes	No	$3 G + P $	$4t_e$	$6 G $
LV-RIBE [12]	Adaptive Standard	DBDH	Yes	No	$3 G + 2 P $	$3t_e$	$(n+6) G $
SE-RIBE [13]	Adaptive Standard	DBDH	Yes	Yes	$3 G + P $	$3t_e$	$(n+6) G $
Our RIBE	Adaptive Standard	SGD	Yes	Yes	$2 G + P $	$2t_e$	$4 G + G_r $

$|G|$ and $|G_r|$ are the sizes of groups G and G_r , respectively. $|P|$ is the size of plaintext space, and n is the size of identity space. t_e is the cost for performing a bilinear pairing $e(G, G) \rightarrow G_r$. Selective (Adaptive, respectively) is a selective-identity security model (adaptive-identity security model, respectively). RO (Standard, respectively) is a random oracle model (standard model, respectively). DBDH is Decisional Bilinear Diffie-Hellman assumption, and SGD is Subgroup Decision assumption. DKE is decryption key exposure.

doi:10.1371/journal.pone.0106925.t001

prove our RIBE construction are IND-RID-CPA secure. Finally, we conclude the paper in Section 6.

Preliminaries

Bilinear group generator and complexity assumptions

Definition 1. (Bilinear Group Generator) A bilinear group generator $\mathcal{G}$ is an algorithm that takes as input a security parameter κ and outputs a bilinear group $(n, \mathbf{G}, \mathbf{G}_T, \hat{e})$, where $\mathbf{G}$ and $\mathbf{G}_T$ are cyclic groups of order n , and $\hat{e}: \mathbf{G} \times \mathbf{G} \rightarrow \mathbf{G}_T$ is a bilinear map with the following properties:

- *Bilinearity:* For all $g, h \in \mathbf{G}$ and $a, b \in \mathbb{Z}_n$, we have $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$.
- *Non-degeneracy:* There is an element $g \in \mathbf{G}$ such that $\hat{e}(g, g)$ has order n in $\mathbf{G}_T$.
- *Computability:* There is an efficient algorithm to compute $\hat{e}(g_1, g_2)$ for all $g_1, g_2 \in \mathbf{G}$.

Denote $\mathcal{G}(1^\kappa) \rightarrow (n = p, \mathbf{G}, \mathbf{G}_T, \hat{e})$ a prime order bilinear groups generator, where p is a prime. We call $\mathcal{G}(1^\kappa) \rightarrow (n = p_1 p_2 p_3, \mathbf{G}, \mathbf{G}_T, \hat{e})$ a composite order bilinear groups generator, where p_1, p_2 and p_3 are distinct primes. The subgroups of order p_1, p_2 and p_3 in $\mathbf{G}$ are denoted by $\mathbf{G}_{p_1}, \mathbf{G}_{p_2}$ and $\mathbf{G}_{p_3}$, respectively. Note that when $h_i \in \mathbf{G}_{p_i}$ and $h_j \in \mathbf{G}_{p_j}$ for $i \neq j$, we have $\hat{e}(h_i, h_j)$ is the identity element in $\mathbf{G}_T$.

Definition 2. (Decision Bilinear Diffie-Hellman Assumption) Given a prime order bilinear group $(p, \mathbf{G}, \mathbf{G}_T, \hat{e})$ generated by $\mathcal{G}(1^\kappa)$, we define the following two distributions:

$$\mathcal{D}_0(\kappa) = (g, g^a, g^b, g^c, \hat{e}(g, g)^{abc}) \text{ and } \mathcal{D}_1(\kappa) = (g, g^a, g^b, g^c, \hat{e}(g, g)^z)$$

where $g \xleftarrow{\$} \mathbf{G}$ and $a, b, c, z \xleftarrow{\$} \mathbb{Z}_p$. The DBDH problem in the prime order bilinear group $(p, \mathbf{G}, \mathbf{G}_T, \hat{e})$ is to decide a bit b from given $\mathcal{D}_b$, where $b \xleftarrow{\$} \{0, 1\}$. The advantage of an algorithm $\mathcal{A}$ in solving the DBDH problem in the prime order bilinear group $(p, \mathbf{G}, \mathbf{G}_T, \hat{e})$ is defined by

$$\text{Adv}_{\mathcal{G}, \mathcal{A}}^{\text{DBDH}}(\kappa) = |\Pr[\mathcal{A}(\mathcal{D}_0(\kappa)) \rightarrow 1] - \Pr[\mathcal{A}(\mathcal{D}_1(\kappa)) \rightarrow 1]|$$

We say that the DBDH assumption holds in the prime order bilinear group $(p, \mathbf{G}, \mathbf{G}_T, \hat{e})$ if no probabilistic polynomial time (PPT) algorithm has a non-negligible advantage in solving the DBDH problem in the prime order bilinear group $(p, \mathbf{G}, \mathbf{G}_T, \hat{e})$.

Assumption 1. (Subgroup decision problem for 3 primes) Given a composite order bilinear group generator $\mathcal{G}(1^\kappa)$, we define the following two distributions:

$$\mathbb{G} = (n = p_1 p_2 p_3, \mathbf{G}, \mathbf{G}_T, \hat{e}) \xleftarrow{\$} \mathcal{G}(1^\kappa), g \xleftarrow{\$} \mathbf{G}_{p_1}, X_3 \xleftarrow{\$} \mathbf{G}_{p_3},$$

$$D = (\mathbb{G}, g, X_3), L_1 \xleftarrow{\$} \mathbf{G}_{p_1 p_2}, L_2 \xleftarrow{\$} \mathbf{G}_{p_1}.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking the subgroup decision assumption 1 to be:

$$\text{Adv}_{\mathcal{G}, \mathcal{A}}(\kappa) = |\Pr[\mathcal{A}(D, L_1) = 1] - \Pr[\mathcal{A}(D, L_2) = 1]|.$$

We note that L_1 can be written (uniquely) as the product of an element of $\mathbf{G}_{p_1}$ and an element of $\mathbf{G}_{p_2}$. We refer to these elements as the “ $\mathbf{G}_{p_1}$ part of L_1 ” and the “ $\mathbf{G}_{p_2}$ part of L_1 ” respectively.

Definition 3. We say that $\mathcal{G}$ satisfies the subgroup decision Assumption 1 if $\text{Adv}_{\mathcal{G}, \mathcal{A}}(\kappa)$ is a negligible function of κ for any polynomial time algorithm $\mathcal{A}$.

Assumption 2. (Subgroup decision problem for 3 primes) Given a composite order bilinear group generator $\mathcal{G}(1^\kappa)$, we define the following two distributions:

$$\begin{aligned} \mathbb{G} = (n = p_1 p_2 p_3, \mathbf{G}, \mathbf{G}_T, \hat{e}) &\xleftarrow{\$} \mathcal{G}(1^\kappa), g, X_1 \xleftarrow{\$} \mathbf{G}_{p_1}, X_2, Y_2 \\ &\xleftarrow{\$} \mathbf{G}_{p_2}, X_3, Y_3 \xleftarrow{\$} \mathbf{G}_{p_3}, \end{aligned}$$

$$D = (\mathbb{G}, g, X_1 X_2, X_3, Y_2 Y_3), L_1 \xleftarrow{\$} \mathbf{G}, L_2 \xleftarrow{\$} \mathbf{G}_{p_1 p_3},$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking the subgroup decision assumption 2 to be:

$$\text{Adv}_{\mathcal{G}, \mathcal{A}}(\kappa) = |\Pr[\mathcal{A}(D, L_1) = 1] - \Pr[\mathcal{A}(D, L_2) = 1]|.$$

Definition 4. We say that $\mathcal{G}$ satisfies the subgroup decision Assumption 2 if $\text{Adv}_{\mathcal{G}, \mathcal{A}}(\kappa)$ is a negligible function of κ for any polynomial time algorithm $\mathcal{A}$.

Assumption 3. (Subgroup decision problem for 3 primes) Given a composite order bilinear group generator $\mathcal{G}(1^\kappa)$, we define the following two distributions:

$$\mathbb{G} = (n = p_1 p_2 p_3, \mathbf{G}, \mathbf{G}_T, \hat{e}) \xleftarrow{\$} \mathcal{G}(1^\kappa), \alpha, s \xleftarrow{\$} \mathbb{Z}_n,$$

$$g \xleftarrow{\$} \mathbf{G}_{p_1}, X_2, Y_2, Z_2 \xleftarrow{\$} \mathbf{G}_{p_2}, X_3 \xleftarrow{\$} \mathbf{G}_{p_3},$$

$$D = (\mathbb{G}, g, g^\alpha X_2, X_3, g^s Y_2, Z_2), L_1 = \hat{e}(g, g)^{\alpha s}, L_2 \xleftarrow{\$} \mathbf{G}_T.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking the subgroup decision assumption 3 to be:

$$\text{Adv}_{\mathcal{G}, \mathcal{A}}(\kappa) = |\Pr[\mathcal{A}(D, L_1) = 1] - \Pr[\mathcal{A}(D, L_2) = 1]|.$$

Definition 5. We say that $\mathcal{G}$ satisfies the subgroup decision Assumption 3 if $\text{Adv}_{\mathcal{G}, \mathcal{A}}(\kappa)$ is a negligible function of κ for any polynomial time algorithm $\mathcal{A}$.

KUNode Algorithm

The KUNode algorithm was proposed by Boldyreva et al. [11] to achieve efficient revocation for IBE schemes. In the description hereafter, we employ similar notations as in [11]. Denote the root node of the tree $\mathbb{T}$ by root . If η is a leaf node, we denote the set of nodes on the path from η to root by $\text{Path}(\eta)$. If η is a non-leaf node, we denote the left and right child of η by η_L and η_R , respectively.

At each time period, KUNode algorithm determines the smallest subset $\mathbf{Y} \subset \mathbb{T}$ of nodes that contains an ancestor of all leaves corresponding to non-revoked users. This minimal set precisely contains nodes for which key updates have to be publicized in such a way that only non-revoked users will be able to generate the appropriate decryption key for the matching period. To identify the set $\mathbf{Y}$, KUNode algorithm takes as input a binary tree $\mathbb{T}$, revocation list $\mathbf{RL}$ and a period number T . If a user (assigned to η) is revoked on time T , then $(\eta, T) \in \mathbf{RL}$. KUNode algorithm first marks all ancestors of users that were revoked by time T as revoked nodes. Then, it inserts in $\mathbf{Y}$ the non-revoked children of revoked nodes. The description of $\text{KUNode}(\mathbb{T}, \mathbf{RL}, T)$ is given in Table 2 Algorithm 2.

The example illustrated in Figure 1 can be used to help the reader understand the $\text{KUNode}(\mathbb{T}, \mathbf{RL}, T)$ algorithm. Assume that a user associated with node x_9 is revoked, then $\mathbf{X} = \text{Path}(x_9) = \{x_9, x_4, x_1, \text{root} = x_0\}$ and $\mathbf{Y} = \{x_2, x_3, x_{10}\}$. Intuitively, all users, except the user associated with node x_9 , have a node $x \in \mathbf{Y}$ that is contained in the set of nodes on the path from their assigned node to root, whereas $\mathbf{Y} \cap \text{Path}(x_9) = \emptyset$.

When a user joins the system, PKG assigns a leaf node η of a complete binary tree to the user, and issues a set of keys, wherein each key is associated with a node on $\text{Path}(\eta)$. At time period T , PKG broadcasts key updates for a set $\text{KUNode}(\mathbb{T}, \mathbf{RL}, T)$. Then, only non-revoked users have at least one key corresponding to a node in $\text{KUNode}(\mathbb{T}, \mathbf{RL}, T)$ and are able to generate decryption keys on time T .

Dual System Encryption

Dual system encryption is a proof methodology first introduced by Waters [8], which opens up a new way to prove adaptive

security under simple assumptions for IBE and related encryption systems.

In a dual system encryption system, both ciphertexts and private keys can take on one of two indistinguishable forms [9]. A private key or ciphertext is normal if they are generated from the system's key generation or encryption algorithm. Semi-functional ciphertexts and private keys are not used in the real system, they are only used in the security proof. A normal private key can decrypt normal or semi-functional ciphertexts, and a normal ciphertext can be decrypted by normal or semi-functional private keys. However, decryption will fail with high probability if one attempts to decrypt a semi-functional ciphertext with a semi-functional private key.

Unlike previous proof technique called partitioning strategy which partitions the identity space into two parts, dual system encryption defines a sequence of games and proves their indistinguishability with the real game. The first game is the real security game in which the challenge ciphertext and private keys are normal. In the next game, the ciphertext is switched from normal to semi-functional, while all the private keys are normal. For an adversary that makes q private key requests, games 1 through q follow. In game k , the first k private keys are semi-functional while the remaining private keys are normal. In game q , all the private keys and the challenge ciphertext given to the adversary are semi-functional. Hence none of the given private keys are useful for decrypting the challenge ciphertext. At this point, At this point proving security becomes relatively easy since the reduction algorithm does not need to present any normal private keys to the adversary and all semi-functional private keys are useless for decrypting a semi-functional ciphertext.

Syntax and Security Definitions of RIBE

In this section, we recall the syntax and security model of RIBE as defined in [13]. Unlike the syntax definition in [13], we define the decryption key generation algorithm as probabilistic rather than deterministic. A RIBE scheme can be defined by the following seven polynomial-time algorithms:

Setup The stateful setup algorithm is run by the PKG, which takes a security parameter κ and a maximal number of users N as input, it outputs the public parameter mpk , the master secret key msk , the initial revocation list $\mathbf{RL} = \emptyset$, and a state $\mathbf{ST}$. We assume that the message space $\mathbf{M}$ and the identity space $\mathbf{I}$, the time space $\mathbf{T}$, and the ciphertext space $\mathbf{CT}$ are contained in mpk .

Extract The stateful private key extract algorithm is run by the PKG, which takes mpk , msk , an identity $\text{ID} \in \mathbf{I}$, a state $\mathbf{ST}$ as input, it outputs a secret key sk_{ID} associated with ID and an updated state $\mathbf{ST}$.

KeyUpdate The key update generation algorithm is run by the PKG, which takes mpk , msk , the key update time $T \in \mathbf{T}$, the current revocation list $\mathbf{RL}$, and $\mathbf{ST}$ as input, it outputs the key update ku_T .

DKeyGen The probabilistic decryption key generation algorithm is run by a user, which takes mpk , sk_{ID} , and ku_T as input, it outputs a decryption key $dk_{\text{ID}, T}$ to be used during period T or a special symbol $\perp$ indicating that ID was revoked.

Encrypt The probabilistic encryption algorithm is run by a sender, which takes mpk , $\text{ID} \in \mathbf{I}$, $T \in \mathbf{T}$, and a message $m \in \mathbf{M}$ as input, it outputs a ciphertext c .

Decrypt The deterministic decryption algorithm is run by the receiver, which takes mpk , $dk_{\text{ID}, T}$, and c as input, it outputs m or $\perp$ if C is an invalid ciphertext.

Revoke The stateful revocation algorithm is run by the PKG, which takes an identity to be revoked $\text{ID} \in \mathbf{I}$, a revocation time

Table 2. Algorithm 2: KUNode Algorithm $\text{KUNode}(\mathbb{T}, \mathbf{RL}, T)$.

$\mathbf{X}, \mathbf{Y} \leftarrow \emptyset$.
$\forall (\eta_i, T_i) \in \mathbf{RL}$
if $T_i \leq T$ then
Add $\text{Path}(\eta_i)$ to $\mathbf{X}$
end if
$\forall x \in \mathbf{X}$
if $x_L \notin \mathbf{X}$ then
Add x_L to $\mathbf{Y}$
end if
if $x_R \notin \mathbf{X}$ then
Add x_R to $\mathbf{Y}$
end if
if $\mathbf{Y} = \emptyset$ then
Add root to $\mathbf{Y}$
end if
Return $\mathbf{Y}$

doi:10.1371/journal.pone.0106925.t002

$T \in \mathbb{T}$, the current revocation list $\mathbf{RL}$, and a state $\mathbf{ST}$ as input, it outputs an updated $\mathbf{RL}$ by adding $\mathbf{ID}$ as a revoked user at time T .

We have a basic consistency requirement that for any $(mpk, msk) \leftarrow \mathbf{Setup}(1^\kappa, N)$, $m \in \mathbf{M}$, all possible state $\mathbf{ST}$, and a revocation list $\mathbf{RL}$, if $\mathbf{ID} \in \mathbf{I}$ is not revoked before or at time $T \in \mathbb{T}$, then for $(sk_{\mathbf{ID}}, \mathbf{ST}) \leftarrow \mathbf{Extract}(mpk, msk, \mathbf{ID}, \mathbf{ST})$, $ku_T \leftarrow \mathbf{KeyUpdate}(mpk, msk, T, \mathbf{RL}, \mathbf{ST})$, and $dk_{\mathbf{ID}, T} \leftarrow \mathbf{DKeyGen}(mpk, sk_{\mathbf{ID}}, ku_T)$, the following equation holds.

$$\mathbf{Decrypt}(mpk, dk_{\mathbf{ID}, T}, \mathbf{Encrypt}(mpk, \mathbf{ID}, T, m)) = m$$

The property of indistinguishability under adaptively chosen identity and chosen plaintext attack (IND-ID-CPA) is considered a basic requirement for provably secure IBE schemes. For RIBE scheme, we define indistinguishability under adaptively chosen revocable identity and chosen plaintext attack (IND-RID-CPA) by the following game between an adversary and a challenger. Note that the security model captures realistic threats including decryption key exposure [13].

Definition 6. Let Π be a RIBE scheme, we say that Π is IND-RID-CPA secure if any PPT adversary $\mathcal{A}$ has negligible advantage in this following experiment:

$$\text{Exp}_{\Pi, \mathcal{A}}^{\text{IND-RID-CPA}}(1^\kappa, N)$$

$$(mpk, msk, \mathbf{RL}, \mathbf{st}) \leftarrow \mathbf{Setup}(1^\kappa, N),$$

$$(m_0, m_1, \mathbf{ID}^*, T^*, \mathbf{ST}) \leftarrow \mathcal{A}^O(\text{Find}, mpk) \text{ such that } |m_0| = |m_1|.$$

$$b \xleftarrow{\$} \{0, 1\}, c^* \leftarrow \mathbf{Encrypt}(mpk, \mathbf{ID}^*, T^*, m_b),$$

$$b' \leftarrow \mathcal{A}^O(\text{Guess}, c^*, \mathbf{ST}),$$

$$\text{return } 1 \text{ if } b' = b \text{ and } 0 \text{ otherwise.}$$

The adversary $\mathcal{A}$'s advantage is defined as follows.

$$\text{Adv}_{\Pi, \mathcal{A}}^{\text{IND-RID-CPA}}(\kappa, N) = |\Pr[\text{Exp}_{\Pi, \mathcal{A}}^{\text{IND-RID-CPA}}(1^\kappa, N) = 1] - \frac{1}{2}|.$$

In the above experiment, $\mathcal{O}$ is a set of oracles defined as follows.

- **Extract Oracle:** For $\mathbf{ID} \in \mathbf{I}$, it runs $\mathbf{Extract}(mpk, msk, \mathbf{ID}, \mathbf{ST}) \rightarrow (sk_{\mathbf{ID}}, \mathbf{ST})$, then returns $sk_{\mathbf{ID}}$ and update state $\mathbf{ST}$.
- **KeyUpdate Oracle:** For $T \in \mathbb{T}$, it runs $\mathbf{KeyUpdate}(mpk, msk, T, \mathbf{RL}, \mathbf{ST}) \rightarrow ku_T$, then returns ku_T .
- **Revoke Oracle:** For $\mathbf{ID} \in \mathbf{I}$ and $T \in \mathbb{T}$, it runs $\mathbf{Revoke}(mpk, \mathbf{ID}, T, \mathbf{RL}, \mathbf{ST}) \rightarrow \mathbf{RL}$, then returns the updated revocation list $\mathbf{RL}$.

- **DKeyGen Oracle:** For $\mathbf{ID} \in \mathbf{I}$ and $T \in \mathbb{T}$, it runs $\mathbf{Extract}(mpk, msk, \mathbf{ID}, \mathbf{ST}) \rightarrow sk_{\mathbf{ID}}$ and $\mathbf{DKeyGen}(mpk, sk_{\mathbf{ID}}, ku_T) \rightarrow dk_{\mathbf{ID}, T}$, then returns $dk_{\mathbf{ID}, T}$.

The adversary $\mathcal{A}$ is allowed to query above oracles with the following restrictions:

- **KeyUpdate Oracle** and **Revoke Oracle** can be queried on time which is greater than or equal to the time of all previous queries, i.e. the adversary is allowed to query only in non-decreasing order of time.
- **Revoke Oracle** cannot be queried on time T if **KeyUpdate Oracle** was queried on T .
- If $\mathbf{Extract}(\mathbf{ID}^*)$ was queried, then $\mathbf{Revoke}(\mathbf{ID}^*, T)$ must be queried for $T \leq T^*$.
- **DKeyGen Oracle** cannot be queried on time T before **KeyUpdate Oracle** was queried on T .
- **KeyGen**($\mathbf{ID}^*, T^*$) cannot be queried.

This definition naturally extends to the chosen ciphertext scenario where the adversary is further granted access to a **Decrypt Oracle** that, on input of a ciphertext c and a pair $(\mathbf{ID}, T)$, it returns $m \in \mathbf{M}$ or $\perp$ by running $\mathbf{Decrypt}(mpk, dk_{\mathbf{ID}, T}, c)$. Of course, **Decrypt Oracle** cannot be queried on the ciphertext c^* for the pair $(\mathbf{ID}^*, T^*)$.

Our Construction

In this section, we propose an efficient and provable secure RIBE scheme by exploiting Lewko and Waters IBE scheme [9] and KUNode algorithm.

Setup The PKG runs composite order bilinear group generator $\mathcal{G}(1^\kappa) \rightarrow (n = p_1 p_2 p_3, \mathbf{G}, \mathbf{G}_T, \hat{e})$, chooses $g, u_1, u_2, h \xleftarrow{\$} \mathbf{G}_{p_1}$ and $\alpha \xleftarrow{\$} \mathbf{Z}_n$. The PKG publishes the public system parameters as follows.

$$mpk = \{n, g, u_1, h, u_2, \hat{e}(g, g)^\alpha\}.$$

The master secret keys are α and a generator of $\mathbf{G}_{p_3}$.

Extract The PKG chooses an unassigned leaf η from $\mathbb{T}$ at random, and stores $\mathbf{ID}$ in the node η . For each node $\theta \in \text{Path}(\eta)$, PKG performs as follows.

- Recall g_θ if it was defined. Otherwise, $g_\theta \xleftarrow{\$} \mathbf{G}_{p_1}$ and store $(g_\theta, \tilde{g}_\theta = g^\alpha / g_\theta)$ in the node θ .
- Choose $r_\theta \xleftarrow{\$} \mathbf{Z}_n$ and $R_3, R'_3, R''_3 \xleftarrow{\$} \mathbf{G}_{p_3}$ at random. Note that we can get a random elements of $\mathbf{G}_{p_3}$ by taking a generator of $\mathbf{G}_{p_3}$ and raising it to random exponents modulo n .
- Compute $(K_1, K_2, K_3) = (g^{r_\theta} R_3, u_1^{r_\theta} R'_3, g_\theta (u_1^{\mathbf{ID}} h)^{r_\theta} R''_3)$.
- Return $sk_{\mathbf{ID}} = \{(K_1, K_2, K_3)\}_{\theta \in \text{Path}(\eta)}$.

KeyUpdate The PKG parses $\mathbf{ST} = \mathbb{T}$, and performs the following steps for each node $\theta \in \text{KUNode}(\mathbb{T}, \mathbf{RL}, T)$.

- Retrieve $\tilde{g}_\theta$ (note that $\tilde{g}_\theta$ is always pre-defined in the **Extract** algorithm).
- Choose $s_\theta \xleftarrow{\$} \mathbf{Z}_n$ and $Q_3, Q'_3, Q''_3 \xleftarrow{\$} \mathbf{G}_{p_3}$.
- Compute $(U_1, U_2, U_3) = (g^{s_\theta} Q_3, u_1^{s_\theta} Q'_3, \tilde{g}_\theta (u_1^{\mathbf{ID}} h)^{s_\theta} Q''_3)$.
- Return $ku_T = \{(U_1, U_2, U_3)\}_{\theta \in \text{KUNode}(\mathbb{T}, \mathbf{RL}, T)}$.

DKeyGen User parses $sk_{ID} = \{(\theta, D_1, D_2, D_3)\}_{\theta \in \mathbf{J}_1}$ and $ku_T = \{(\theta, U_1, U_2, U_3)\}_{\theta \in \mathbf{J}_2}$. If $\mathbf{J}_1 \cap \mathbf{J}_2 = \emptyset$, then outputs error symbol $\perp$. Otherwise, user chooses $\theta \in \mathbf{J}_1 \cap \mathbf{J}_2$ and $r \xleftarrow{\$} \mathbf{Z}_n$ and outputs

$$dk_{ID,T} = (D_1, D_2) = (K_1 U_1 g^r, K_2^T U_2^{ID} K_3 U_3 (u_1^{ID} u_2^T h)^r)$$

Encrypt A sender chooses a random integer $t \xleftarrow{\$} \mathbf{Z}_n$ and outputs

$$C_0 = M \hat{e}(g, g)^{zt}, C_1 = (u_1^{ID} u_2^T h)^t, C_2 = g^t.$$

Decrypt The receiver parses $C = (C_0, C_1, C_2)$ and $dk_{ID,T} = (D_1, D_2)$ and outputs

$$C_0 \frac{\hat{e}(D_2, C_2)}{\hat{e}(D_1, C_1)}$$

Revoke Let η be the leaf node associated with ID . The PKG updates the revocation list by $\mathbf{RL} \leftarrow \mathbf{RL} \cup \{(\eta, T)\}$ and returns the updated revocation list.

The correctness of our RIBE construction can be verified as follow.

$$\begin{aligned} \frac{\hat{e}(D_2, C_2)}{\hat{e}(D_1, C_1)} &= \frac{\hat{e}(K_2^T U_2^{ID} K_3 U_3 (u_1^{ID} u_2^T h)^r, g^t)}{\hat{e}(K_1 U_1 g^r, (u_1^{ID} u_2^T h)^t)} \\ &= \frac{\hat{e}(g_0 \tilde{g}_0 (u_1^{ID} u_2^T h)^{r\theta + s_\theta + r} R_3^r Q_3^r Q_3^{ID} R_3^T, g^t)}{\hat{e}(g^{r\theta + s_\theta + r} R_3 Q_3, (u_1^{ID} u_2^T h)^t)} \\ &= \hat{e}(g_0 \tilde{g}_0, g^t) \\ &= \hat{e}(g, g)^{zt} \end{aligned}$$

Security Proofs

To prove the security of our RIBE scheme, we first define three additional structures: semi-functional ciphertexts, semi-functional private keys and semi-functional update keys. For the semi-functional type, we let g_2 denote a fixed generator of the subgroup $\mathbf{G}_{p_2}$.

- **Semi-functional Ciphertext:** A normal ciphertext $C' = (C'_0, C'_1, C'_2)$ is first generated by the encryption algorithm. It then chooses $x, z_c \xleftarrow{\$} \mathbf{Z}_n$ and sets:

$$C_0 = C'_0, C_1 = C'_1 g_2^{xz_c}, C_2 = C'_2 g_2^x.$$

- The semi-functional ciphertext is $C = (C_0, C_1, C_2)$.

- **Semi-functional Private Key:** A normal private key $sk'_{ID} = \{(K'_1, K'_2, K'_3)\}_{\theta \in \text{Path}(\eta)}$ is generated by the private key generation algorithm for an identity ID . It then chooses $\gamma, z_k, z'_k \xleftarrow{\$} \mathbf{Z}_n$ and sets:

$$K_1 = K'_1 g_2^\gamma, K_2 = K'_2 g_2^{z'_k \gamma}, K_3 = K'_3 g_2^{z'_k}.$$

- The semi-functional private key is $sk_{ID} = \{(K_1, K_2, K_3)\}_{\theta \in \text{Path}(\eta)}$.

- **Semi-functional Update Key:** A normal update key $ku_T = \{(U'_1, U'_2, U'_3)\}_{\theta \in \text{KUNode}(\mathbf{T}, \mathbf{RL}, \mathbf{T})}$ is generated by the update key generation algorithm. It then chooses $\lambda, z_u, z'_u \xleftarrow{\$} \mathbf{Z}_n$ and sets:

$$U_1 = U'_1 g_2^\lambda, U_2 = U'_2 g_2^{z'_u \lambda}, U_3 = U'_3 g_2^{z'_u}.$$

- The semi-functional update key is $ku_T = \{(U_1, U_2, U_3)\}_{\theta \in \text{KUNode}(\mathbf{T}, \mathbf{RL}, \mathbf{T})}$.

- **Semi-functional Decryption Key:** A normal decryption key $dk'_{ID,T} = (D'_1, D'_2)$ is generated by the decryption key generation algorithm. It then chooses $\rho, z_d \xleftarrow{\$} \mathbf{Z}_n$ and sets:

$$D_1 = D'_1 g_2^\rho, D_2 = D'_2 g_2^{z_d \rho}.$$

- The semi-functional decryption key is $dk_{ID,T} = (D_1, D_2)$.

Note that when a semi-functional decryption key is used to decrypt a semi-functional ciphertext, the decryption algorithm will compute the blinding factor multiplied by the additional term $\hat{e}(g_2, g_2)^{x\rho(z_d - z_c)}$. If $z_d = z_c$, decryption will still work. In this case, the decryption key is nominally semi-functional. In our proof, normal decryption keys are generated by normal subkeys of private keys and normal subkeys of update keys, while semi-functional decryption keys are generated by semi-functional subkeys of private keys and normal subkeys of update key, or normal subkeys of private keys and semi-functional subkeys of update keys.

There are two types of adversaries in simulation. Type-I adversary issues private key queries on the challenge identity ID^* , but the challenge identity should be revoked before the challenge time T^* ; Type-II adversary will never issue private key queries on the challenge identity. Obviously, if a RIBE scheme is secure against Type-I adversary, it is definitely secure against Type-II adversary. For this reason, we only consider Type-I adversary in the following security proofs.

Denote by q_{sk} and q_{ku} the number of private key queries for non-challenge identities and update key queries for non-challenge

time issued by an adversary, respectively. Denote by ℓ the maximum node number a private key involves, and those nodes are not on the path from the root node to the challenge node η^* .

We give our proof as a sequence of games, which are defined in the order as follows.

- Game_A : The actual RIBE security game, where all private keys, update keys, decryption keys and the challenge ciphertext are normal.
- Game_R : The restricted game, is the actual security game except that adversary can not issue private key queries for $\text{ID} = \text{ID}^* \bmod p_2$ and update key queries for $T = T^* \bmod p_2$. Note that adversary can issue private key queries for $\text{ID} = \text{ID}^* \bmod n$, but ID^* should be revoked before T^* .
- $\text{Game}_{i,j}^{\text{sk}}$: The restricted security game where the challenge ciphertext, all ℓ subkeys of first $i-1$ private keys and all first j subkeys $sk_{\text{ID},\theta}$ of the i -th private key sk_{ID} are semi-functional, while all subkeys of the rest private keys and all subkeys of update keys are normal. Here $0 \leq j \leq \ell$, $1 \leq i \leq q_{sk}$ and $\theta \notin \text{Path}(\eta^*)$.
- $\text{Game}_k^{\text{ku}}$: The restricted security game where the challenge ciphertext, all ℓ subkeys of all private keys, and subkeys $ku_{T,\theta}$ of the first k update key ku_T are semi-functional, while the rest subkeys of q_{sk} private key and the rest subkeys of q_{sk} update keys are normal. Here $\theta \in \text{Path}(\eta^*)$. It is obvious that $\text{Game}_{q_{sk},\ell}^{\text{sk}} = \text{Game}_0^{\text{ku}}$.
- Game_F : The final game, is the same as security game $\text{Game}_{q_{sk},\ell}^{\text{ku}}$ except that the challenge ciphertext is a semi-functional encryption of a random message.

Next, we prove the indistinguishability of those games by following lemmas.

Lemma 1. Suppose there exists an algorithm $\mathcal{A}$ such that $\text{Adv}_{\mathcal{A}}^{\text{Game}_A} - \text{Adv}_{\mathcal{A}}^{\text{Game}_R} = \epsilon$, then we can build an algorithm $\mathcal{B}$ with advantage $\frac{\epsilon}{2}$ in breaking Assumption 2.

Proof. Given $g, X_1, X_2, X_3, Y_2, Y_3$, algorithm $\mathcal{B}$ can simulate Game_A with $\mathcal{A}$. Assume that $\mathcal{A}$ produces identities ID and ID^* such that $\text{ID} \neq \text{ID}^* \bmod n$ and p_2 divides $\text{ID} - \text{ID}^*$ with probability ϵ (If $\mathcal{A}$ fails to do this, $\mathcal{B}$ simply guesses at random). $\mathcal{B}$ uses these identities to produce a nontrivial factor of n by computing $a = \gcd(\text{ID} - \text{ID}^*, n)$. Set $b = \frac{n}{a}$, and consider the following three cases:

- Case 1 one of a, b is p_1 , and the other is $p_2 p_3$
- Case 2 one of a, b is p_2 , and the other is $p_1 p_3$
- Case 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 $\mathbf{G}_{p_2}$ component or not by testing if $\hat{e}(T^a, X_1 X_2)$ is the identity element. If it is not, then T has a $\mathbf{G}_{p_2}$ 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 them 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. Without loss of generality, we assume that $a = p_2$ and $b = p_1 p_3$. Then, $\mathcal{B}$ can learn whether T has a $\mathbf{G}_{p_2}$ component or not by testing if T^b is the identity element. If it is not, then T has a $\mathbf{G}_{p_2}$ component.

$\mathcal{B}$ can determine that Case 3 has occurred when the tests for both Cases 1 and Case 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. Without loss of generality, we assume that $a = p_3$. $\mathcal{B}$ can learn whether T has a $\mathbf{G}_{p_2}$ component or not by testing whether $\hat{e}(T^a, Y_2 Y_3)$ is the identity. If it is not, then $\mathbf{G}_{p_2}$ has a $\mathbf{G}_{p_2}$ component.

This completes the proof. $\square$

Lemma 2. Suppose there exists an algorithm $\mathcal{A}$ such that $\text{Adv}_{\mathcal{A}}^{\text{Game}_R} - \text{Adv}_{\mathcal{A}}^{\text{Game}_{0,0}^{\text{sk}}} = \epsilon$, then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 1.

Proof. $\mathcal{B}$ first receives g, X_3, L , then simulates Game_R or Game_0 with $\mathcal{A}$. $\mathcal{B}$ chooses $\alpha, a_1, b, a_2 \xleftarrow{\$} \mathbf{Z}_n$, sets public parameters as $g = g$, $u_1 = g^{a_1}$, $u_2 = g^{a_2}$, $h = g^b$, and sends the public parameters to $\mathcal{A}$.

- When $\mathcal{B}$ is asked to provide a update key with time period T . For each node $\theta \in \text{KUNode}(T, \mathbf{RL}, T)$, $\mathcal{B}$ performs the following steps.
 - Retrieve $\tilde{g}_\theta$ (Note that $\tilde{g}_\theta$ is always pre-defined in the **Extract** algorithm).
 - Choose $s_\theta, t_\theta, t'_\theta, t''_\theta \xleftarrow{\$} \mathbf{Z}_n$.
 - Compute $(U_1, U_2, U_3) = (g^{s_\theta} X_3^{t_\theta}, u_1^{s_\theta} X_3^{t'_\theta}, \tilde{g}_\theta(u_2^T h)^{s_\theta} X_3^{t''_\theta})$.
 - Return $ku_T = \{(U_1, U_2, U_3)\}_{\theta \in \text{KUNode}(T, \mathbf{RL}, T)}$.
- When $\mathcal{B}$ is asked for a private key with identity ID . For each node $\theta \in \text{Path}(\eta)$ where η is the leaf node assigned to ID , $\mathcal{B}$ performs the following steps.
 - Recall g_θ if it was defined. Otherwise, $g_\theta \xleftarrow{\$} \mathbf{G}_{p_1}$ (We can do this by taking the generator of $\mathbf{G}_{p_1}$, g , and raising it to random exponents modulo n) and store $(g_\theta, \tilde{g}_\theta = g^\alpha / g_\theta)$ in the node θ .
 - Choose $r_\theta, y_\theta, y'_\theta, y''_\theta \xleftarrow{\$} \mathbf{Z}_n$.
 - Compute $(K_1, K_2, K_3) = (g^{r_\theta} X_3^{y_\theta}, u_2^{r_\theta} X_3^{y'_\theta}, g_\theta(u_1^{\text{ID}} h)^{r_\theta} X_3^{y''_\theta})$.
 - Return $sk_{\text{ID}} = \{(K_1, K_2, K_3)\}_{\theta \in \text{Path}(\eta)}$.
- When $\mathcal{B}$ is asked for a decryption key with identity ID and time period T , then $\mathcal{B}$ successively runs the **Extract** algorithm, **KeyUpdate** algorithm and **DKeyGen** algorithm.

$\mathcal{A}$ sends $\mathcal{B}$ two message, M_0 and M_1 , and a challenge identity, ID^* , challenge time period, T^* . $\mathcal{B}$ chooses $\beta \in \{0, 1\}$ randomly. The ciphertext is formed as follows.

$$C_0 = M_\beta \hat{e}(L, g)^\alpha, \quad C_1 = L^{z_c}, \quad C_2 = L.$$

This implicitly sets g^s equal to the $\mathbf{G}_{p_1}$ part of L . If $L \in \mathbf{G}_{p_1 p_2}$, then this is a semi-functional ciphertext with $z_c = a_1 \text{ID}^* + a_2 T^* + b$. We note that the value of z_c modulo p_2 is not correlated with the values of a and b modulo p_1 , so z_c is properly distributed. If $L \in \mathbf{G}_{p_1}$, this is a normal ciphertext. Hence, simulator $\mathcal{B}$ can use the output of $\mathcal{A}$ to distinguish between these possibilities for L .

This completes the proof. $\square$

Lemma 3. Suppose there exists an algorithm $\mathcal{A}$ such that $\text{Adv}_{\mathcal{A}}^{\text{Game}_{k,k'}^{\text{sk}}} - \text{Adv}_{\mathcal{A}}^{\text{Game}_{k,k'}^{\text{sk}}} = \epsilon$, then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 2.

Proof. $\mathcal{B}$ first receives $g, X_1 X_2, X_3, Y_2 Y_3, L$, and picks $\alpha, a_1, b, a_2 \xleftarrow{\$} \mathbb{Z}_n$, then $\mathcal{B}$ sets the public parameters as $g = g, u_1 = g^{a_1}, h = g^b, u_2 = g^{a_2}, \hat{e}(g, g)^z$ and sends the public parameters to $\mathcal{A}$.

- When $\mathcal{A}$ issues the i -th private key query for all subkeys corresponding to the challenge identity, or subkeys that associated with nodes are not on the path from the node associated with challenge identity to the root node, $\mathcal{B}$ generates normal private keys by calling the normal private key generation algorithm. Otherwise, $\mathcal{B}$ generate the j -th subkey, associated with those ℓ subkeys, of the i -th private key as follows.

1. For $i < k \vee (i = k \wedge j \leq k')$ and $\theta \in \text{Path}(\eta)$

- Recall g_θ if it was defined. Otherwise, $g_\theta \xleftarrow{\$} \mathbf{G}_{p_1}$ and store $(g_\theta, \tilde{g}_\theta = g^z / g_\theta)$ in the node θ .
- Choose $r_\theta, y_\theta, y'_\theta, y''_\theta \xleftarrow{\$} \mathbb{Z}_n$ randomly.
- Compute $(K_1, K_2, K_3) = (g^{r_\theta} (Y_2 Y_3)^{y_\theta}, u_2^{r_\theta} (Y_2 Y_3)^{y'_\theta}, g_\theta (u_1^{\text{ID}} h)^{r_\theta} (Y_2 Y_3)^{y''_\theta})$.
- Return $sk_{\text{ID}} = \{(K_1, K_2, K_3)\}_{\theta \in \text{Path}(\eta)}$.

2. For $i = k \wedge j = k'$ and $\theta \in \text{Path}(\eta)$,

- Recall g_θ if it was defined. Otherwise, $g_\theta \xleftarrow{\$} \mathbf{G}_{p_1}$ and store $(g_\theta, \tilde{g}_\theta = g^z / g_\theta)$ in the node θ .
- Choose $w, w' \xleftarrow{\$} \mathbb{Z}_n$ randomly.
- Compute $(K_1, K_2, K_3) = (L, L^{a_2} X_3^{w'}, g_\theta L^{z_k} X_3^w)$.
- Return $sk_{\text{ID}} = \{(K_1, K_2, K_3)\}_{\theta \in \text{Path}(\eta)}$.

3. For $i > k \vee (i = k \wedge j > k')$, $\mathcal{B}$ generates normal private keys by calling the normal private key generation algorithm.

- When $\mathcal{A}$ issues a update key query with time period T , then $\mathcal{B}$ generates normal update keys by calling the normal update key generation algorithm.
- When $\mathcal{A}$ issues a decryption key query with identity ID and time period T , then $\mathcal{B}$ successively runs the **Extract** algorithm, **KeyUpdate** algorithm, and **DKeyGen** algorithm.

At some point $\mathcal{A}$ sends two messages, M_0 and M_1 , a challenge identity ID^* , and a challenge time period T^* to $\mathcal{B}$. $\mathcal{B}$ sets $\beta \in \{0, 1\}$ randomly. The challenge ciphertext is formed as follows.

$$C_0 = M_\beta \hat{e}(X_1 X_2, g)^\alpha, C_1 = (X_1 X_2)^{z_c}, C_2 = X_1 X_2.$$

We note that this sets $g^s = X_1$ and $z_c = a_1 \text{ID}^* + a_2 T^* + b$. Since $f(\text{ID}, T) = a_1 \text{ID} + a_2 T + b$ is a pairwise independent function modulo p_2 , as long as $\text{ID} \neq \text{ID}^* \bmod p_2$ and $T \neq T^* \bmod p_2$, z_k and z_c will seem randomly distributed to $\mathcal{A}$.

If $L \in \mathbf{G}_{p_1 p_3}$, then $\mathcal{B}$ has properly simulated $\text{Game}_{k, k'-1}^{\text{sk}}$. If $L \in \mathbf{G}$, then $\mathcal{B}$ has properly simulated $\text{Game}_{k, k'}^{\text{sk}}$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to distinguish between these possibilities for L .

This completes the proof. $\square$

Lemma 4. Suppose there exists an algorithm $\mathcal{A}$ such that $\text{Adv}_{\mathcal{A}}^{\text{Game}_{k, \ell}^{\text{sk}}} - \text{Adv}_{\mathcal{A}}^{\text{Game}_{k+1, 0}^{\text{sk}}} = \epsilon$, then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 2.

Proof. This proof is analogous to the proof of lemma 3. $\square$

Lemma 5. Suppose there exists an algorithm $\mathcal{A}$ such that $\text{Adv}_{\mathcal{A}}^{\text{Game}_{k-1}^{\text{ku}}} - \text{Adv}_{\mathcal{A}}^{\text{Game}_k^{\text{ku}}} = \epsilon$, then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 2.

Proof. $\mathcal{B}$ first receives $g, X_1 X_2, X_3, Y_2 Y_3, L$, and picks $\alpha, a_1, b, a_2 \xleftarrow{\$} \mathbb{Z}_n$, then $\mathcal{B}$ sets the public parameters as $g = g, u_1 = g^{a_1}, h = g^b, u_2 = g^{a_2}, \hat{e}(g, g)^z$ and sends the public parameters to $\mathcal{A}$.

- When $\mathcal{A}$ issues private key query for the challenge identity or subkeys that associated with nodes are not on the path from the node associated with challenge identity to the root node, $\mathcal{B}$ generates normal private keys by calling the normal private key generation algorithm. Otherwise, for each node $\theta \in \text{Path}(\eta)$, $\mathcal{B}$ performs as follows.

- Recall g_θ if it was defined. Otherwise, $g_\theta \xleftarrow{\$} \mathbf{G}_{p_1}$ and store $(g_\theta, \tilde{g}_\theta = g^z / g_\theta)$ in the node θ .
- Choose $r_\theta, y_\theta, y'_\theta, y''_\theta \xleftarrow{\$} \mathbb{Z}_n$ randomly.
- Compute $(K_1, K_2, K_3) = (g^{r_\theta} (Y_2 Y_3)^{y_\theta}, u_2^{r_\theta} (Y_2 Y_3)^{y'_\theta}, g_\theta (u_1^{\text{ID}} h)^{r_\theta} (Y_2 Y_3)^{y''_\theta})$.
- Return $sk_{\text{ID}} = \{(K_1, K_2, K_3)\}_{\theta \in \text{Path}(\eta)}$.

- When $\mathcal{A}$ issues the update key query for the challenge time period, $\mathcal{B}$ generates normal update keys by calling the normal update key generation algorithm. Otherwise, $\mathcal{B}$ performs as follows.

1. For $i < k$. When $\theta \in \text{KUNode}(\mathbb{T}, \mathbf{RL}, T) \wedge \theta \notin \text{Path}(\eta^*)$, $\mathcal{B}$ calls the normal update key generation algorithm. When $\theta \in \text{KUNode}(\mathbb{T}, \mathbf{RL}, T) \wedge \theta \in \text{Path}(\eta^*)$, $\mathcal{B}$ acts as follow. Note that there is only one node that satisfies this condition in each update node set.

- Retrieve $\tilde{g}_\theta$.
- Choose $s_\theta, t_\theta, t'_\theta, t''_\theta \xleftarrow{\$} \mathbb{Z}_n$.
- Compute $(U_1, U_2, U_3) = (g^{s_\theta} (Y_2 Y_3)^{t_\theta}, u_1^{s_\theta} (Y_2 Y_3)^{t'_\theta}, \tilde{g}_\theta (u_2^T h)^{s_\theta} (Y_2 Y_3)^{t''_\theta})$.
- Return $ku_T = \{(U_1, U_2, U_3)\}_{\theta \in \text{KUNode}(\mathbb{T}, \mathbf{RL}, T)}$.

2. For $i = k$. When $\theta \in \text{KUNode}(\mathbb{T}, \mathbf{RL}, T) \wedge \theta \notin \text{Path}(\eta^*)$, $\mathcal{B}$ generates normal update keys by calling the normal update key generation algorithm. When $\theta \in \text{Path}(\eta^*)$, $\mathcal{B}$ performs as follows.

- Retrieve $\tilde{g}_\theta$.
- Choose $s_\theta, w_\theta, w'_\theta \xleftarrow{\$} \mathbb{Z}_n$.
- Compute $(U_1, U_2, U_3) = (L, L^{a_1} X_3^{w'_\theta}, \tilde{g}_\theta L^{z_u} (X_3)^{w_\theta})$.
- Return $ku_T = \{(U_1, U_2, U_3)\}_{\theta \in \text{KUNode}(\mathbb{T}, \mathbf{RL}, T)}$.

— Here we note that $z_u = a_2 T + b$ and $T \neq T^*$, therefore both $z_u = a_2 T + b$ and $z_c = a_1 \text{ID}^* + a_2 T^* + b$ seem random in adversary's view. If $T = T^*$, namely we transform update key with time period T^* , then we can not ensure that $z_u = a_2 T^* + b$ and $z_c = a_1 \text{ID}^* + a_2 T^* + b$ seem random in adversary's view.

3. For $i > k$. When $\theta \in \text{KUNode}(\mathbb{T}, \mathbf{RL}, T)$, $\mathcal{B}$ generates normal update keys by calling the normal update key generation algorithm.

- When $\mathcal{B}$ is asked for a decryption key with identity ID and time period T , $\mathcal{B}$ successively runs the **Extract** algorithm, **KeyUpdate** algorithm and **DKeyGen** algorithm.

At some point $\mathcal{A}$ sends two messages, M_0 and M_1 , a challenge identity ID^* , and a challenge time period T^* to $\mathcal{B}$. $\mathcal{B}$ sets $\beta \in \{0,1\}$ randomly. The challenge ciphertext is formed as follows.

$$C_0 = M_\beta \hat{e}(X_1 X_2, g)^z, C_1 = (X_1 X_2)^{z_c}, C_2 = X_1 X_2.$$

If $L \in \mathbf{G}_{p_1 p_3}$, then $\mathcal{B}$ has properly simulated Game'_{k-1} . If $L \in \mathbf{G}$, then $\mathcal{B}$ has properly simulated Game'_k . Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to distinguish between these possibilities for L .

This completes the proof. $\square$

Lemma 6. Suppose there exists an algorithm $\mathcal{A}$ such that $\text{Adv}_{\mathcal{A}}^{\text{Game}_{\text{ku}}} - \text{Adv}_{\mathcal{A}}^{\text{Game}_F}$, then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 3.

Proof. $\mathcal{B}$ first receives $g, g^z X_2, X_3, g^s Y_2, Z_2, L$, chooses $a, b, a', b' \xleftarrow{\$} \mathbf{Z}_n$, then $\mathcal{B}$ sets the public parameters as $g = g, u_1 = g^{a_1}, h = g^b, u_2 = g^{a_2}, \hat{e}(g, g)^z = \hat{e}(g^z X_2, g)$ and sends the public parameters to $\mathcal{A}$.

- When $\mathcal{A}$ issues private key queries with the challenge identity or nodes on the path from challenge identity node to root node, $\mathcal{B}$ generates normal private keys by calling the normal private key generation algorithm. Otherwise, for each node $\theta \in \text{Path}(\eta)$, $\mathcal{B}$ performs as follows.

- Recall g_θ if it was defined. Otherwise, $g_\theta \xleftarrow{\$} \mathbf{G}_{p_1}$ and store $(g_\theta, \tilde{g}_\theta = 1/g_\theta)$ in the node θ .
- Choose $r_\theta, z_\theta, z'_\theta, v_\theta, v'_\theta, w_\theta, w'_\theta \xleftarrow{\$} \mathbf{Z}_n$ randomly.
- Compute $(K_1, K_2, K_3) = (g^{r_\theta} Z_2^{z_\theta} X_3^{v_\theta}, u_2^{r_\theta} Z_2^{z'_\theta} X_3^{v'_\theta}, g^z X_2 g_\theta (u_1^{\text{ID}} h)^{r_\theta} Z_2^{z_\theta} X_3^{v_\theta})$.
- Return $sk_{\text{ID}} = \{(K_1, K_2, K_3)\}_{\theta \in \text{Path}(\eta)}$.

- When $\mathcal{A}$ issues update key query with the challenge time period, $\mathcal{B}$ generates normal update keys by calling the normal update key generation algorithm. Otherwise, $\mathcal{B}$ performs as follows.

- If $\theta \notin \text{Path}(\eta^*)$, $\mathcal{B}$ generates normal update keys by calling the normal update key generation algorithm.
- If $\theta \in \text{Path}(\eta^*)$, $\mathcal{B}$ performs the following steps. Note that there is only one such node in each time period T .

- * Retrieve $\tilde{g}_\theta$ (note that $\tilde{g}_\theta$ is always pre-defined in the **Extract** algorithm).
- * Choose $s_\theta, v_\theta, v'_\theta, w_\theta, w'_\theta \xleftarrow{\$} \mathbf{Z}_n$.
- * Compute $(U_1, U_2, U_3) = (g^{s_\theta} Z_2^{v_\theta} X_3^{w_\theta}, u_1^{s_\theta} Z_2^{v'_\theta} X_3^{w'_\theta}, g^z X_2 \tilde{g}_\theta (u_2^T h)^{s_\theta} Z_2^{v_\theta} X_3^{w_\theta})$.
- * Return $ku_T = \{(U_1, U_2, U_3)\}_{\theta \in \text{KUNode}(T, \mathbf{RL}, T)}$.

- When $\mathcal{A}$ issues decryption key query with identity ID and time period T , $\mathcal{B}$ successively runs the **Extract** algorithm, **KeyUpdate** algorithm and **DKeyGen** algorithm.

At some point $\mathcal{A}$ sends two messages, M_0 and M_1 , a challenge identity ID^* , and a challenge time period T^* to $\mathcal{B}$. $\mathcal{B}$ sets $\beta \in \{0,1\}$ randomly. The challenge ciphertext is formed as follows.

$$C_0 = M_\beta L, C_1 = (g^s Y_2)^{z_c}, C_2 = g^s Y_2.$$

Here $z_c = a_1 ID^* + a_2 T^* + b$. We note that the value of z_c only matters modulo p_2 , whereas $u_1 = g^{a_1}, u_2 = g^{a_2}$ and $h = g^b$ are elements of $\mathbf{G}_{p_1}$, so when a_1, a_2 and b modulo p_1 are chosen randomly modulo n , there is no correlation between the values of a_1, a_2 and b modulo p_1 and the value $z_c = a_1 ID^* + a_2 T^* + b \bmod p_2$.

If $L = \hat{e}(g, g)^{zs}$, then this is a properly distributed semi-functional ciphertext with message M_β . If L is a random element of $\mathbf{G}_T$, then this is a semi-functional ciphertext with a random message. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to distinguish between these possibilities for L .

This completes the proof. $\square$

Theorem 1. If above lemmas hold, then our RIBE scheme is adaptively secure under assumption 1, 2 and 3. More precisely, for any adversary $\mathcal{A}$ that makes at most q_1 private key queries, q_2 update key queries against our RIBE scheme, we have

$$\text{Adv}_{\mathcal{A}}^{\text{RIBE}} \leq \frac{1}{2} q_1 q_2 (q_1 \text{Adv}_{\mathcal{G}, \mathcal{A}}^{\text{SGD}^2}(\kappa) \log N_{\max} + q_2 \text{Adv}_{\mathcal{G}, \mathcal{A}}^{\text{SGD}^2}(\kappa) + \text{Adv}_{\mathcal{G}, \mathcal{A}}^{\text{SGD}^3}(\kappa))$$

Proof. If above assumptions hold, then we have shown by the previous lemmas that the real security game is indistinguishable from $\text{Game}_{\text{Final}}$, in which the value of β is information-theoretically hidden from the adversary. Hence the adversary can attain no advantage in breaking our RIBE scheme.

This completes the proof.

Conclusion

In this paper, we presented a scalable RIBE scheme with decryption key exposure resilience in the composite order group setting by combining Lewko and Waters' IBE scheme and complete subtree method, and proved our proposed RIBE scheme to be adaptive-ID secure by employing the recent dual system encryption methodology. Compared to existing adaptive-ID secure LV-RIBE scheme and SE-RIBE scheme, our proposed RIBE construction is more efficient in term of ciphertext size, public parameters size and decryption cost at price of a little looser security reduction. In our future work, we will focus on constructing an adaptive-ID secure RIBE scheme with decryption key exposure resilience in the prime order group setting and devising an adaptive-ID secure RIBE scheme that can resist decryption key exposure attack with a tighter reduction.

Acknowledgments

The authors would like to thank the anonymous reviewers of this paper for his/her objective comments and helpful suggestions while at the same time helping us to improve the English spelling and grammar throughout the manuscript.

Author Contributions

Conceived and designed the experiments: CW YL. Performed the experiments: CW YL XX KZ. Analyzed the data: CW YL. Contributed

reagents/materials/analysis tools: CW YL XX KZ. Wrote the paper: CW YL. Constructed the scheme: CW YL XX KZ. Proved the security of scheme: CW YL.

References

- Shamir A (1985) Identity-based cryptosystems and signature schemes. In: *Advances in Cryptology- CRYPTO 84*. California, USA: Springer Berlin Heidelberg, volume 196 of Lecture Notes in Computer Science, pp. 47–53.
- Boneh D, Franklin M (2001) Identity-based encryption from the weil pairing. In: *Advances in Cryptology - CRYPTO 2001*. California, USA: Springer Berlin Heidelberg, volume 2139 of Lecture Notes in Computer Science, pp. 213–229.
- Canetti R, Goldreich O, Halevi S (2004) The random oracle methodology, revisited. *Journal of the ACM* 51: 557–594.
- Canetti R, Halevi S, Katz J (2004) Chosen-ciphertext security from identity-based encryption. In: *Advances in Cryptology - EUROCRYPT 2004*. Interlaken, Switzerland: Springer Berlin Heidelberg, volume 3027 of Lecture Notes in Computer Science, pp. 207–222.
- Boneh D, Boyen X (2004) Efficient selective-id secure identity-based encryption without random oracles. In: *Advances in Cryptology - EUROCRYPT 2004*. Interlaken, Switzerland: Springer Berlin Heidelberg, volume 3027 of Lecture Notes in Computer Science, pp. 223–238.
- Waters B (2005) Efficient identity-based encryption without random oracles. In: *Advances in Cryptology - EUROCRYPT 2005*. Aarhus, Denmark: Springer Berlin Heidelberg, volume 3494 of Lecture Notes in Computer Science, pp. 114–127.
- Gentry C (2006) Practical identity-based encryption without random oracles. In: *Advances in Cryptology - EUROCRYPT 2006*. St. Petersburg, Russia: Springer Berlin Heidelberg, volume 4004 of Lecture Notes in Computer Science, pp. 445–464.
- Waters B (2009) Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In: *Advances in Cryptology - CRYPTO 2009*. California, USA: Springer Berlin Heidelberg, volume 5677 of Lecture Notes in Computer Science, pp. 619–636.
- Lewko A, Waters B (2010) New techniques for dual system encryption and fully secure hibe with short ciphertexts. In: *Theory of Cryptography - TCC 2010*. Zurich, Switzerland: Springer Berlin Heidelberg, volume 5978 of Lecture Notes in Computer Science, pp. 455–479.
- Zhang Y, Wang L, Zhang Y, Li X (2012) Toward a temporal network analysis of interactive wifusers. *Europhysics Letters* 98: 68002.
- Boldyreva A, Goyal V, Kumar V (2008) Identity-based encryption with efficient revocation. In: *Proceedings of the 15th ACM Conference on Computer and Communications Security*. Virginia, USA: ACM, CCS 2008, pp. 417–426.
- Libert B, Vergnaud D (2009) Adaptive-id secure revocable identity-based encryption. In: *Topics in Cryptology - CT-RSA 2009*. California, USA: Springer Berlin Heidelberg, volume 5473 of Lecture Notes in Computer Science, pp. 1–15.
- Seo J, Emura K (2013) Revocable identity-based encryption revisited: Security model and construction. In: *Public Key Cryptography - PKC 2013*. Nara, Japan: Springer Berlin Heidelberg, volume 7778 of Lecture Notes in Computer Science, pp. 2161–234.
- Park S, Lee K, Lee D (2013). New constructions of revocable identity-based encryption from multilinear maps. *Cryptology ePrint Archive*. Report 2013/880.
- Lee K, Lee D, Park J (2014). Efficient revocable identity-based encryption via subset difference methods. *Cryptology ePrint Archive*. Report 2014/132.
- Sahai A, Waters B (2005) Fuzzy identity based encryption. In: *Advances in Cryptology - EUROCRYPT 2005*. Aarhus, Denmark: Springer Berlin Heidelberg, volume 3494 of Lecture Notes in Computer Science, pp. 457–473.
- Naor D, Naor M, Lotspiech J (2001) Revocation and tracing schemes for stateless receivers. In: *Advances in Cryptology - CRYPTO 2001*. California, USA: Springer Berlin Heidelberg, volume 2139 of Lecture Notes in Computer Science, pp. 41–62.
- Libert B, Vergnaud D (2009) Towards black-box accountable authority ibe with short ciphertexts and private keys. In: *Public Key Cryptography - PKC 2009*. California, USA: Springer Berlin Heidelberg, volume 5443 of Lecture Notes in Computer Science, pp. 235–255.