Public-Key Revocation and Tracing Schemes with Subset Difference Methods Revisited

Kwangsu Lee[*] Woo Kwon Koo[†] Dong Hoon Lee[‡] Jong Hwan Park[§]

Abstract

Trace and revoke is broadcast encryption with the traitor tracing functionality. It is a very powerful primitive since it can revoke users whose private keys are compromised by finding them using a tracing algorithm if a pirate decoder is given. *Public-key trace and revoke (PKTR)* is a special type of trace and revoke such that anyone can run the tracing algorithm and anyone can create an encrypted message by using a public key. Although public-key trace and revoke schemes are attractive tools, the currently known PKTR schemes are a little bit inefficient in terms of the private key size and the public key size compared with public-key broadcast encryption schemes.

In this paper, we propose a new technique to construct an efficient PKTR scheme by using the subset cover framework. First, we introduce a new concept of public-key encryption named *single revocation encryption (SRE)* and propose an efficient SRE scheme in the random oracle model. The universe of SRE consists of many groups and each group consists of many members. A user in SRE is represented as a group that he belongs and a member in the group. In SRE, a sender can create a ciphertext for a specified group where one member in the group is revoked, and a receiver can decrypt the ciphertext if he belongs to the group in the ciphertext and he is not revoked in the group. Second, we show that the subset difference (SD) scheme (or the layered subset difference (LSD) scheme) and an SRE scheme can be combined to construct a PKTR scheme. Our PKTR scheme using the LSD scheme and our SRE scheme has the ciphertext size of $O(r)$, the private key size of $O(\log^{1.5} N)$, and the public key size of $O(1)$ where N is the total number of users in the system and r is the size of a revoked set. Our PKTR scheme is the first one that achieves the private key size of $O(\log^{1.5} N)$ and the public key size of $O(1)$.

Keywords: Public-key encryption, Broadcast encryption, Traitor tracing, Trace and revoke, Bilinear maps.

[*]Korea University, Seoul, Korea. Email: `guspin@korea.ac.kr`.

[†]Korea University, Seoul, Korea. Email: `kwk4386@korea.ac.kr`.

[‡]Korea University, Seoul, Korea. Email: `donghlee@korea.ac.kr`.

[§]Sangmyung University, Seoul, Korea. Email: `jhpark@smu.ac.kr`. This work was partially done at Korea University.

1 Introduction

Broadcast encryption, introduced by Fiat and Naor [12], is a mechanism to efficiently send an encrypted message to a set S of receivers by using a broadcast channel. The application of broadcast encryption includes pay-TV systems, DVD content distribution systems, and file systems and many others. Broadcast encryption itself is a very powerful primitive, but the functionality of broadcast encryption can also be increased when it is combined with traitor tracing functionality. Traitor tracing was introduced by Chor, Fiat, and Naor [8], and it enables a tracer to find a traitor who participated the creation of a pirate decoder when a pirate decoder is given to the tracer. Trace and revoke is a mechanism that combines broadcast encryption and traitor tracing, and it first finds a traitor by using the tracing algorithm of traitor tracing and then revoke him by using the encrypt algorithm of broadcast encryption [7, 21, 23].

Public-key trace and revoke (PKTR) is a special type of trace and revoke such that anyone can trace a traitor and revoke the users by using a publicly known public key. Although PKTR is a powerful primitive, we can not construct a PKTR scheme by simply combining a public-key broadcast encryption (PKBE) scheme and a public-key traitor tracing (PKTT) scheme since a simply combined PKTR scheme is not secure against a *collusion attack* [7]. There are two general methods for the construction of fully collusion resistant PKTR schemes. The first method is to combine a subset cover scheme in the framework of Naor, Naor, and Lotspiech [21] and an identity-based encryption (IBE) scheme (or a hierarchical IBE (HIBE) scheme) [11, 21]. The PKTR schemes of this method are suitable for the revocation scenario where a small number of users are revoked since the ciphertext size of the schemes is linear to the size of revoked users. However, the most efficient scheme of this method that combines the layered subset difference (LSD) scheme of Halevy and Shamir [18] and the HIBE scheme of Boneh et al. [3] has a demerit such that the size of private keys is $O(\log^{2.5} N)$ where N is the total number of users in the system. The second method is to combine a private linear broadcast encryption (PLBE) scheme that was introduced by Boneh, Sahai, and Waters [6] and a PKBE scheme [7, 15, 26]. The PKTR schemes of this method are quite efficient in terms of the ciphertext size since the size of ciphertexts is independent of the size of a receiver set. However, the storage requirement of these schemes are quite large since the size of private keys and public keys of these schemes is $O(\sqrt{N})$ where N is the total number of users in the system.

Reducing the size of private keys is very important since these cryptographic key materials are securely stored in expensive tamper-resistant memory. In case of small devices, the size of private keys and public keys is critical issue since the manufacturing cost of small devices is limited. Although there are many PKBE schemes that can meet this requirement [20], there is no acceptable PKTR scheme as far as we know.

1.1 Our Contributions

In this paper, we revisit the method of Dodis and Fazio [11] that combines the SD scheme in the subset cover framework and the variation scheme of IBE to construct an efficient PKTR scheme, and propose a new method for PKTR that can reduce the size of private keys and public keys. The subset cover framework of Naor et al. [21] was very successful to construct broadcast encryption or trace and revoke schemes in the symmetric-key setting [17, 18, 21]. However, the trace and revoke schemes of the subset cover framework in the public-key setting does not provide the same efficiency parameter as that in the symmetric-key setting since the underlying HIBE scheme multiplies the private key size and the public key size of PKTR by $\log N$ factor [3, 11]. For instance, the PKTR scheme that combines the LSD scheme and the HIBE scheme of Boneh et al. [3] has the ciphertext size of $O(r)$, the private key size of $O(\log^{2.5} N)$, and the public key size of $O(\log N)$.

To construct an efficient PKTR scheme by using the subset cover framework, we first introduce single

Table 1: Comparison of public-key broadcast encryption schemes

Scheme	CT Size	SK Size	PK Size	Decrypt Time	Tracing	Assumption
BGW [5]	$O(1)$	$O(1)$	$O(N)$	2P	No	q-Type
BGW [5]	$O(\sqrt{N})$	$O(1)$	$O(\sqrt{N})$	2P	No	q-Type
Delerablée [9]	$O(1)$	$O(1)$	$O(s_{max})$	2P	No	q-Type
LSW [20]	$O(r)$	$O(1)$	$O(1)$	rE + 2P	No	q-Type
NNL [22]	$O(r \log \frac{N}{r})$	$O(\log N)$	$O(1)$	1P	Yes	BDH
DF [11]	$O(r)$	$O(\log^{2.5} N)$	$O(\log N)$	2P	Yes	q-Type
BW [7]	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(\sqrt{N})$	4P	Yes	Static
Ours	$O(r)$	$O(\log^{1.5} N)$	$O(1)$	2E + 2P	Yes	q-Type

N = the maximum number of users, s_{max} = the maximum size of a receiver set, r = the size of a revoked set

E = exponentiation, P = pairing

revocation encryption (SRE) that can be efficiently combined with the subset difference (SD) scheme, and propose an efficient SRE scheme that is secure in the random oracle model. A user in SRE is represented as a group label and a member label in the group, and a sender can send an encrypted message to one specified group except one member that was revoked in that group. If a user who belongs to the group is not revoked in the group, then he can decrypt the ciphertext by using this private key. Our SRE scheme has the ciphertext size of $O(1)$, the private key size of $O(1)$, and the public key size of $O(1)$, and it is secure in the random oracle model under q-type assumption.

Next, we show that it is possible to construct an efficient PKTR scheme by combining the SD scheme (or the LSD scheme) and the SRE scheme. Compared to the PKTR scheme that combines the LSD scheme and the HIBE scheme of Boneh et al. [3], our PKTR scheme that combines the LSD scheme and our SRE scheme has the shorter size of private keys and public keys. The comparison between previous PKBE schemes, PKTR schemes, and our schemes is given in the Table 1. In the table, the PKTR scheme of Dodis and Fazio is a scheme that combines the LSD scheme and the HIBE scheme of Boneh et al. [3], and our PKTR scheme is a scheme that combines the LSD scheme and our SRE scheme.

1.2 Our Technique

The main idea of our PKTR scheme is to invent a new type of public-key encryption (PKE) that has short private key size and can be integrated with the SD scheme of the subset cover framework. In order to understand our technique, we first review the SD scheme of Naor et al. [21]. In a full binary tree $\mathcal{T}$, a subtree T_i rooted at a node v_i is defined as the set of all nodes in T_i and a subtree $T_{i,j}$ is defined as the set of nodes in $T_i - T_j$ where a node v_j is a descendant of a node v_i. In the SD scheme, a user in the SD scheme is assigned to a leaf node in $\mathcal{T}$, and a subset $S_{i,j}$ is defined as the set of leaf nodes in $T_{i,j}$. A user in a leaf node v_u is associated with the set PV_u of subsets $S_{i,j}$ where v_i and v_j are two nodes in the path from the root node of $\mathcal{T}$ to the leaf node v_u. The set S of receivers is associated with the set CV of disjoint subsets $S_{i,j}$ that covers S. If a user u is not revoked, then he can find two subsets $S_{i,j} \in CV$ and $S_{i',j'} \in PV_u$ such that $v_i = v_{i'}$, $d_j = d_{j'}$, and $v_j \neq v_{j'}$ where d_j is the depth of a node v_j. Next, the user can decrypt the ciphertext component that is related with $S_{i,j}$ by using the private key components that are related with PV_u.

One critical condition for the decryption using the SD scheme is that the inequality $v_j \neq v_{j'}$ should be satisfied. For this inequality, Naor et al. [21] used the key derivation property of a key assignment algorithm, and Dodis and Fazio [11] used the delegation property of a key generation algorithm in HIBE. To devise a new technique to solve this issue, we look at the IBRE scheme of Lewko, Sahai, and Waters [20]. The notable property of the IBRE scheme is that the decryption is successful only when ID is not equal to ID' where ID is associated with a ciphertext and ID' is associated with a private key. However, the direct combination of this IBRE scheme and the SD scheme is not successful since the IBRE scheme does not support an equality condition. Therefore, we construct an SRE scheme by modifying this IBRE scheme to support two conditions of equality and inequality.

As described in the previous section, a user in SRE is represented as labels (GL, ML) where GL is a group label and ML is a member label in the group, and a sender creates a ciphertext with labels (GL', ML') for all member in the group GL' except the one member ML' in the group. Thus a receiver who has a private key with labels (GL, ML) can decrypt the ciphertext with labels (GL', ML') if $(GL = GL') \wedge (ML \neq ML')$. Therefore, SRE supports the equality of group labels and the inequality of member labels. To integrate the SRE scheme that uses group and member labels (GL, ML) with the SD scheme that uses subsets $S_{i,j}$ in a full binary tree, we need a mapping from the subset $S_{i,j}$ to the labels (GL, ML). A subset $S_{i,j}$ is defined by two nodes v_i, v_j and a subtree T_i is defined by one node v_i. For the mapping function from the subset $S_{i,j}$ to labels (GL, ML), we define the set of all nodes in the subtree T_i that has the same depth as v_j as a one group, and we also define the nodes in the group as the members of the group. That is, if the nodes v_i and v_j of $S_{i,j}$ in the SD scheme have identifiers L_i and L_j respectively, then the labels in the SRE scheme are represented as $GL = L_i \| d_j$ and $ML = L_j$ where d_j is the depth of v_j.

1.3 Related Work

Broadcast Encryption. As mentioned, the concept of broadcast encryption was introduced by Fiat and Naor [12] and broadcast encryption can efficiently send an encrypted message to a set of receivers through a broadcast channel. Many broadcast encryption schemes including the scheme of Fiat and Naor were designed to be secure against a collusion of t users. In this case, if an attacker can compromise the private keys of more than t users, then he can easily breaks the security of the broadcast encryption scheme.

To construct a fully collusion resilient broadcast encryption scheme that is secure without a bound on the number of colluded users, Naor, Naor, and Lotspiech [21, 22] proposed a general method called the subset cover framework, and they proposed symmetric-key revocation schemes such that a center can broadcast an encrypted message to all users except r number of revoked users. They proposed two broadcast encryption schemes of the subset cover framework, named as the complete subtree (CS) and the subset difference (SD) scheme. The CS scheme has the ciphertext size of $O(r \log N/r)$ and the private key size of $O(\log N)$, and the SD scheme has the ciphertext size of $O(r)$ and the private key size of $O(\log^2 N)$ where N is the number of users in the system and r is the number of revoked users. Halevy and Shamir [18] proposed the layered subset difference (LSD) scheme that has the ciphertext size of $O(r)$ and the private key size of $O(\log^{1.5} N)$, and Goodrich et al. [17] proposed the stratified subset difference (SSD) scheme that has the ciphertext size of $O(r)$ and the private key size of $O(\log N)$.

Public-key broadcast encryption (PKBE) is a special type of broadcast encryption such that anyone can send an encrypted message to a set of receivers through a broadcast channel by using a public key. Naor et al. [21, 22] observed that their CS scheme can be combined with the identity-based encryption (IBE) scheme of Boneh and Franklin [4] to reduce the size of public keys in PKBE. Dodis and Fazio [11] showed that the SD scheme (or the LSD scheme) can also be combined with a hierarchical IBE (HIBE) scheme to construct an efficient PKBE scheme. For instance, if the LSD scheme is combined with the HIBE scheme of Boneh, Boyen and Goh [3], then the PKBE scheme has the ciphertext size of $O(r)$, the private key size of $O(\log^{2.5} N)$, and the public key size of $O(\log N)$. Note that the private key size of this PKBE scheme is larger than that of the original LSD scheme in the symmetric-key setting.

The PKBE scheme of the subset cover framework could be inefficient when the set of revoked users is a medium sized set such as $\sqrt{N}$ since the size of ciphertexts is linear to r. To solve this problem, Boneh, Gentry, and Waters [5] proposed the first fully collusion-resistant PKBE scheme that has constant size of ciphertexts based on bilinear groups. Their first PKBE scheme has the ciphertext size of $O(1)$, the private key size of $O(1)$, and the public key size of $O(N)$, and their second PKBE scheme has the ciphertext size of $O(\sqrt{N})$, the private key size of $O(1)$, and the public key size of $O(\sqrt{N})$ where N is the number of users in the system. After the construction of Boneh et al. [5], many other PKBE schemes based on bilinear groups were proposed [10, 16, 24, 27]. Delerablée [9] proposed an identity-based broadcast encryption (IBBE) scheme such that the total number of users is not fixed, the ciphertext is related to the set S of receivers, and the public key size is linear to the maximum size of receiver sets. Lewko et al. [20] proposed an identity-based revocation encryption (IBRE) scheme such that the ciphertext is related to the set of revoked users R instead of the set of receiver users S. Their IBRE scheme has the ciphertext size of $O(r)$, the private key size of $O(1)$, and the public key size of $O(1)$.

Traitor Tracing. The concept of traitor tracing was introduced by Chor, Fiat, and Naor [8] and traitor tracing enables a tracer who is given a pirate decoder to detect at least one user who participated the creation of the pirate decoder. Many traitor tracing schemes were designed to be secure against a collusion of t users. Fully collusion resistant traitor tracing schemes were proposed based on bilinear groups [6,15,25]. Abdalla et al. [1] proposed the concept of identity-based traitor tracing (IBTT) and constructed an IBTT scheme.

Trace and Revoke. Trace and revoke is broadcast encryption combined with traitor tracing such that it first finds a user whose private key is compromised by using the tracing algorithm of traitor tracing and then it revokes the user by using the revocation algorithm of broadcast encryption [21,23]. Many trace and revoke schemes were secure against a collusion of t users [23]. Naor et al. [21,22] proposed the first fully collusion resistant trace and revoke schemes by using the general method of the subset cover framework.

Public-key trace and revoke (PKTR) is a special type of trace and revoke such that anyone can trace traitors and revoke the user by using a public key. The PKBE scheme of Dodis and Fazio [11] can also be a PKTR scheme since their scheme also follows the subset cover framework. Boneh and Waters [7] proposed a fully collusion resistant PKTR scheme based on composite order bilinear groups and proved its adaptive security by combining the PKBE scheme of Boneh et al. [5] and the traitor tracing scheme of Boneh et al. [6]. The efficiency of this scheme was improved by using prime order bilinear groups [15,26]. Furukawa and Attrapadung [14] proposed a PKTR scheme with short private keys, but the public key size of this is quite large and the security is only proven in the generic group model.

2 Preliminaries

In this section, we briefly review bilinear groups and introduce the complexity assumption of our scheme.

2.1 Bilinear Groups

Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$. The bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ has the following properties:

1. Bilinearity: $\forall u, v \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$.

2. Non-degeneracy: $\exists g$ such that $e(g, g)$ has order p, that is, $e(g, g)$ is a generator of $\mathbb{G}_T$.

We say that $\mathbb{G}, \mathbb{G}_T$ are bilinear groups if the group operations in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the bilinear map e are all efficiently computable.

2.2 Complexity Assumptions

To prove the security of our PKRE scheme, we introduce a new assumption called q-Simplified Multi-Exponent Bilinear Diffie-Hellman (q-SMEBDH) assumption. This q-SMEBDH assumption is derived from the q-Multi-Exponent Bilinear Diffie-Hellman (q-MEBDH) assumption that was introduced by Lewko, Sahai, and Waters [20] with a slight simplification. Our new assumption is secure in the generic group model by using the master theorem of Boneh, Boyen, and Goh [3].

Assumption 2.1 (q-Simplified Multi-Exponent Bilinear Diffie-Hellman, q-SMEBDH)**.** *Let $(p, \mathbb{G}, \mathbb{G}_T, e)$ be a description of the bilinear group of prime order p with the security parameter λ. Let g be a generator of $\mathbb{G}$. The q-SMEBDH assumption is that if the challenge values*

$$D = ((p, \mathbb{G}, \mathbb{G}_T, e),\ g,\ \{g^{a_i},\ g^{b/a_i}\}_{1 \le i \le q},\ \{g^{ba_i/a_j}\}_{1 \le i,j, i \ne j \le q},\ g^c) \text{ and } T$$

are given, no PPT algorithm $\mathcal{B}$ can distinguish $T = T_0 = e(g,g)^{bc}$ from $T = T_1 = e(g,g)^d$ with more than a negligible advantage. The advantage of $\mathcal{B}$ is defined as $\textbf{Adv}_{\mathcal{B}}^{q\text{-}SMEBDH}(\lambda) = \big|\Pr[\mathcal{B}(D, T_0) = 0] - \Pr[\mathcal{B}(D, T_1) = 0]\big|$ where the probability is taken over the random choice of $a_1, \ldots, a_q, b, c, d \in \mathbb{Z}_p$.

3 Single Revocation Encryption

In this section, we define single revocation encryption (SRE) and the security model of SRE, and then we propose an SRE scheme and prove its security in the random oracle model.

3.1 Definitions

Single revocation encryption (SRE) is a special type of public-key broadcast encryption (PKBE) such that a single user in a group can be revoked. That is, a sender in SRE can securely transmit a message to the members of a specified group except the single revoked member of the group. In SRE, the universe $\mathcal{U}$ is defined as the set of many groups that consist of many members. Note that the maximum number of groups and the maximum number of members in a group is a polynomial number in a security parameter. A center first generates a master key and a public key for SRE by using a setup algorithm, and each user specified by a group label and a member label can receive his private key from the center. To transmit a message, a sender computes a ciphertext by specifying a group label and a revoked member in the group. If a user belongs to the group in the ciphertext and he is not revoked, then he can decrypt the ciphertext by using his private key. The following is the syntax of SRE.

Definition 3.1 (Single Revocation Encryption)**.** *A SRE scheme for the universe $\mathcal{U}$ of groups and members consists of four algorithms **Setup**, **GenKey**, **Encrypt**, and **Decrypt**, which are defined as follows:*

***Setup**($1^\lambda, \mathcal{U}$). The setup algorithm takes as input a security parameter 1^λ and the universe $\mathcal{U}$ of groups and members. It outputs a master key MK and a public key PK.*

***GenKey**($(GL, ML), MK, PK$). The key generation algorithm takes as input labels (GL, ML), the master key MK, and the public key PK. It outputs a private key SK for the labels (GL, ML).*

Encrypt$((GL,ML),M,PK)$*. The encryption algorithm takes as input labels* (GL,ML)*, a message* $M \in \mathcal{M}$*, and the public key PK. It outputs a ciphertext CT for* (GL,ML) *and M.*

Decrypt(CT,SK,PK)*. The decryption algorithm takes as input a ciphertext CT for labels* (GL,ML)*, a private key SK for labels* (GL',ML')*, and the public key PK. It outputs an encrypted message M or* $\perp$*.*

The correctness property of SRE is defined as follows: For all MK,PK *generated by* ***Setup****, all* u,S*, any* SK_u *generated by* ***GenKey****, and any M, it is required that*

- *If* $(GL = GL') \wedge (ML \neq ML')$*, then* $\textbf{\textit{Decrypt}}(\textbf{\textit{Encrypt}}((GL,ML),M,PP),SK_{(GL',ML')},PK) = M$*.*
- *If* $(GL \neq GL') \vee (ML = ML')$*, then* $\textbf{\textit{Decrypt}}(\textbf{\textit{Encrypt}}((GL,ML),M,PP),SK_{(GL',ML')},PK) = \perp$ *with all but negligible probability.*

The security property of SRE is defined as indistinguishability. The indistinguishability game of SRE can be similarly defined by modifying the indistinguishability game of PKBE. In this game, an adversary is first given a public key of SRE, and then he can obtain many private keys for labels. In the challenge step, the adversary submits challenge labels and two challenge messages, and then he receives a challenge ciphertext. Finally, the adversary outputs a guess for the random coin that is used to create the challenge ciphertext. If the guess of the adversary is correct, then the adversary wins the game. The following is the formal definition of indistinguishability.

Definition 3.2 (Indistinguishability)**.** *The indistinguishability property of SRE under a chosen plaintext attack is defined in terms of the following game between a challenger* $\mathcal{C}$ *and a PPT adversary* $\mathcal{A}$*:*

1. ***Setup****:* $\mathcal{C}$ *runs* ***Setup****(*$1^\lambda,\mathcal{U}$*) to generate a master key MK and a public key PK. It keeps MK to itself and gives PK to* $\mathcal{A}$*.*
2. ***Query****:* $\mathcal{A}$ *adaptively requests private keys for labels* $(GL_1,ML_1),\ldots,(GL_q,ML_q)$*. In response,* $\mathcal{C}$ *gives the corresponding private keys* $SK_1,\ldots,SK_q$ *to* $\mathcal{A}$ *by running* ***GenKey****(*$(GL_i,ML_i),MK,PK$*).*
3. ***Challenge****:* $\mathcal{A}$ *submits challenge labels* (GL^*,ML^*) *and two messages* M_0^*,M_1^* *with the equal length subject to the restriction: for all* (GL_i,ML_i) *of private key queries, it is required that* $(GL_i \neq GL^*)$ *or* $((GL_i = GL^*) \wedge (ML_i = ML^*))$*.* $\mathcal{C}$ *flips a random coin* $\gamma \in \{0,1\}$ *and gives the challenge ciphertext* CT^* *to* $\mathcal{A}$ *by running* ***Encrypt****(*$(GL^*,ML^*),M_\gamma^*,PK$*).*
4. ***Guess****:* $\mathcal{A}$ *outputs a guess* $\gamma' \in \{0,1\}$ *of* γ*, and wins the game if* $\gamma = \gamma'$*.*

The advantage of $\mathcal{A}$ *is defined as* $\textbf{\textit{Adv}}_{\mathcal{A}}^{SRE}(\lambda) = \big|\Pr[\gamma = \gamma'] - \frac{1}{2}\big|$ *where the probability is taken over all the randomness of the game. A SRE scheme is indistinguishable under a chosen plaintext attack if for all PPT adversary* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *in the above game is negligible in the security parameter* λ*.*

3.2 Construction

Our SRE scheme is inspired by the IBRE scheme of Lewko, Sahai, and Waters [20] that employs the "two equation" technique. In the two equation technique, a ciphertext is associated with a revoked set $R = \{ID_1,\ldots,ID_r\}$ of users and a user is associated with an identity ID. If a user is not revoked ($ID \neq ID_i$), then he will obtain two independent equations and can decrypt the ciphertext. However, if a user is revoked ($ID = ID_i$), then he will obtain two dependent equations and thus cannot decrypt the ciphertext. Lewko et al. [20] constructed a IBRE scheme that has private keys of constant size, public keys of constant size, and ciphertexts of $O(r)$ size. We construct an SRE scheme that enables a sender to broadcast a ciphertext to a given group except a one specified member in the group by slightly modifying the IBRE scheme of Lewko et al. First, the IBRE scheme can be modified to revoke a single user instead of multiple users, and then the modified scheme has a private key $SK = (g^{\alpha} w^{r}, (hw^{ID})^{r}, g^{-r})$ and a ciphertext $CT = (e(g,g)^{\alpha t} M, g^{t}, (hw^{ID})^{t})$ where ID is a user identifier. However this modified scheme does not support groups. To support groups, we first represent a user identifier ID as labels (GL, ML) of a group and a member, and use hash functions H_1, H_2 to select unique h, w values for each group. Then the modified scheme has a private key $SK = (g^{\alpha} H_2(GL)^{r}, (H_1(GL) H_2(GL)^{ML})^{r}, g^{-r})$ and a ciphertext $CT = (e(g,g)^{\alpha t}, g^{t}, (H_1(GL) H_2(GL)^{ML})^{t})$ where GL is a group label and ML is a member label.

Let $\mathcal{U} = \{(GL_i, \{ML_j\})\}$ be the universe of groups and members where the maximum number U_g of groups is a polynomial number in a security parameter and the maximum number U_m of members in a group is also a polynomial numbers in a security parameter. Our SRE scheme for the universe $\mathcal{U}$ is described as follows:

SRE.Setup(1^{λ}, $\mathcal{U}$): This algorithm first generates the bilinear groups $\mathbb{G}$ of prime order p of bit size $\Theta(\lambda)$. It chooses a random element $g \in \mathbb{G}$. It selects a random exponent $\alpha \in \mathbb{Z}_p$. It outputs a master key $MK = g^{\alpha}$ and a public key as

$$PK = \Big((p, \mathbb{G}, \mathbb{G}_T, e),\ g,\ H_1, H_2,\ \Omega = e(g,g)^{\alpha} \Big).$$

SRE.GenKey($(GL, ML), MK, PK$): This algorithm takes as input labels (GL, ML), the master key MK, and the public key PK. It selects a random exponent $r \in \mathbb{Z}_p$ and outputs a private key by implicitly including (GL, ML) as

$$SK_{(GL,ML)} = \Big(K_0 = g^{\alpha} H_2(GL)^{r},\ K_1 = (H_1(GL) H_2(GL)^{ML})^{r},\ K_2 = g^{-r} \Big).$$

SRE.Encrypt($(GL, ML), M, PK$): This algorithm takes as input labels (GL, ML), a message $M \in \mathbb{G}_T$, and the public key PK. It chooses a random exponent $t \in \mathbb{Z}_p$ and outputs a ciphertext by implicitly including (GL, ML) as

$$CT_{(GL,ML)} = \Big(C_0 = \Omega^{t} M,\ C_1 = g^{t},\ C_2 = (H_1(GL) H_2(GL)^{ML})^{t} \Big).$$

SRE.Decrypt($CT_{(GL,ML)}, SK_{(GL',ML')}, PK$): This algorithm takes as input a ciphertext $CT_{(GL,ML)}$, a private key $SK_{(GL',ML')}$, and the public key PK. If $(GL = GL') \wedge (ML \neq ML')$, then it outputs a message as

$$M = C_0 \cdot e(C_1, K_0)^{-1} \cdot (e(C_1, K_1) \cdot e(C_2, K_2))^{1/(ML' - ML)}.$$

Otherwise, it outputs $\perp$.

The correctness of the above SRE scheme is easily verified by the following equation.

$$\begin{aligned}
& e(C_1, K_0) / \left(e(C_1, K_1) \cdot e(C_2, K_2)\right)^{1/(ML'-ML)} \\
&= e(g^{t}, g^{\alpha} H_2(GL)^{r}) / \Big(e(g^{t}, (H_1(GL) H_2(GL)^{ML'})^{r}) \cdot e((H_1(GL) H_2(GL)^{ML})^{t}, g^{-r}) \Big)^{1/(ML'-ML)} \\
&= e(g^{t}, g^{\alpha} H_2(GL)^{r}) / \Big(e(g, H_2(GL))^{tr \cdot (ML'-ML)} \Big)^{1/(ML'-ML)} \\
&= e(g,g)^{\alpha t}.
\end{aligned}$$

3.3 Security Analysis

Theorem 3.3. *The above SRE scheme is indistinguishable under a chosen plaintext attack in the random oracle model if the q-SMEBDH assumption holds where $U_m \leq q$.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that breaks the indistinguishability game of the SRE scheme with a non-negligible advantage. A simulator $\mathcal{B}$ that solves the q-SMEBDH assumption using $\mathcal{A}$ is given: a challenge tuple $D = ((p, \mathbb{G}, \mathbb{G}_T, e), g, \{g^{a_i}, g^{b/a_i}\}_{1 \leq i \leq q}, \{g^{ba_i/a_j}\}_{1 \leq i,j,i \neq j \leq q}, g^c)$ and T where $T = T_0 = e(g,g)^{bc}$ or $T = T_1 = e(g,g)^d$. Then $\mathcal{B}$ that interacts with $\mathcal{A}$ is described as follows:

Setup: $\mathcal{B}$ first guesses challenge labels (GL', ML') such that ML' is a member of GL'. Next, it initializes two lists H_1-List and H_2-List for random oracles as empty sets. It implicitly sets $\alpha = b$ and creates the public key as $PK = \big((p, \mathbb{G}, \mathbb{G}_T, e), g,\ H_1, H_2, \Omega = e(g^{a_1}, g^{b/a_1})\big)$.

Query: $\mathcal{A}$ may adaptively request hash queries or private key queries. Let **MemSet**(GL) be a function that takes a group label GL as an input and outputs the set $\{ML_i\}$ of members in the group, $\rho(GL, ML)$ be a function that takes a group label GL and a member label ML as inputs and outputs an index k of the member in the group, and $\textbf{RevSet}_{GL',ML'}(GL)$ be a function that outputs **MemSet**(GL) if $GL \neq GL'$ or $\{ML'\}$ if $GL = GL'$. For notational convenience, we use **RevSet**(GL) instead of $\textbf{RevSet}_{GL',ML'}(GL)$.

If this is an i-th H_1 hash query on a label GL, then $\mathcal{B}$ handles this query as follows:

1. If there exists a tuple $(GL, -, -)$ in the H_1-List, then it returns $H_1(GL)$ from the H_1-List.
2. It sets $H_1(GL) = \prod_{\forall ML_k \in \textbf{RevSet}(GL)} (g^{a_{\rho(GL,ML_k)}})^{-ML_k} \cdot g^{h_{1,i}}$ by choosing a random exponent $h_{1,i} \in \mathbb{Z}_p$. Note that if $GL = GL'$, then it sets $H_1(GL') = (g^{a_{\rho(GL,ML_{j'})}})^{-ML'} g^{h_{1,i}}$ since $\textbf{RevSet}(GL') = \{ML'\}$ where j' is the index of $ML_{j'}$ such that $ML' = ML_{j'}$.
3. It saves a tuple $(GL, h_{1,i}, H_1(GL))$ to the H_1-List and returns $H_1(GL)$.

If this is a H_2 hash query on a label GL, then $\mathcal{B}$ handles this query as follows:

1. If there exists a tuple $(GL, -, -)$ in the H_2-List, then it returns $H_2(GL)$ from the H_2-List.
2. It sets $H_2(GL) = \prod_{\forall ML_k \in \textbf{RevSet}(GL)} g^{a_{\rho(GL,ML_k)}} \cdot g^{h_{2,i}}$ by choosing a random exponent $h_{2,i} \in \mathbb{Z}_p$. Note that if $GL = GL'$, then it sets $H_2(GL') = g^{a_{\rho(GL,ML_{j'})}} g^{h_{2,i}}$ since $\textbf{RevSet}(GL') = \{ML'\}$ where j' is the index of $ML_{j'}$ such that $ML' = ML_{j'}$.
3. It saves a tuple $(GL, h_{2,i}, H_2(GL))$ to the H_2-List and returns $H_2(GL)$.

If this is a private key query for labels (GL, ML) where $ML = ML_j$ and $\rho(GL, ML) = j$, then $\mathcal{B}$ handles this query as follows:

1. If $(GL = GL') \wedge (ML \neq ML')$, then it aborts since it cannot create a private key.
2. It first retrieves a tuple $(GL, h_{1,i}, H_1(GL))$ for GL from H_1-List and a tuple $(GL, h_{2,i}, H_2(GL))$ for GL from H_2-List.
3. Next, it selects a random exponent $r' \in \mathbb{Z}_p$ and creates a private key $SK_{(GL,ML)}$ by implicitly setting $r = -b/a_{\rho(GL,ML_j)} + r'$ as

$$K_0 = \prod_{\forall ML_k \in \textbf{RevSet}(GL) \setminus \{ML_j\}} (g^{a_{\rho(GL,ML_k)}/a_{\rho(GL,ML_j)} \cdot b})^{-1} (g^{1/a_{\rho(GL,ML_j)} \cdot b})^{-h_{2,i}} \cdot H_2(GL)^{r'},$$

$$K_1 = \prod_{\forall ML_k \in \mathbf{RevSet}(GL)\setminus\{ML_j\}} \big(g^{a_{\rho(GL,ML_k)}/a_{\rho(GL,ML_j)}\cdot b}\big)^{ML_k - ML_j}\big(g^{1/a_{\rho(GL,ML_j)}\cdot b}\big)^{-h_{1,i}-h_{2,i}ML_j} \cdot$$
$$\big(H_1(GL)H_2(GL)^{ML_j}\big)^{r'},$$
$$K_2 = g^{1/a_{\rho(GL,ML_j)}\cdot b} g^{-r'}.$$

Challenge: $\mathcal{A}$ submits challenge labels (GL^*, ML^*) and two messages M_0^*, M_1^*. If $(GL' \neq GL^*) \vee (ML' \neq ML^*)$, then $\mathcal{B}$ aborts the simulation since it failed to guess the challenge labels. Otherwise, $\mathcal{B}$ flips a random coin $\gamma \in \{0,1\}$ internally. Next, it retrieves tuples $(GL^*, h_1^*, H_1(GL^*))$ and $(GL^*, h_2^*, H_2(GL^*))$ from H_1-List and H_2-List respectively. It implicitly sets $t = c$ and creates a challenge ciphertext as

$$C_0 = T \cdot M_\gamma^*,\ C_1 = g^c,\ C_2 = (g^c)^{h_1^* + h_2^* ML^*}.$$

Output: Finally, $\mathcal{A}$ outputs a guess γ'. If $\gamma = \gamma'$, $\mathcal{B}$ outputs 0. Otherwise, it outputs 1.

To finish the proof, we first show that hash outputs, private keys, and the challenge ciphertext are correctly distributed. The hash outputs are correctly distributed since new random elements h_1 and h_2 are chosen for H_1 and H_2 hash queries. The private key is correctly distributed since it satisfies the following equation

$$\begin{aligned}
K_0 &= g^\alpha H_2(GL)^r = g^b\big(\prod_{\forall ML_k \in \mathbf{RevSet}(GL)} g^{a_{\rho(GL,ML_k)}} \cdot g^{h_{2,i}}\big)^{-b/a_{\rho(GL,ML_j)}+r'} \\
&= \prod_{\forall ML_k \in \mathbf{RevSet}(GL)\setminus\{ML_j\}} \big(g^{a_{\rho(GL,ML_k)}/a_{\rho(GL,ML_j)}\cdot b}\big)^{-1}\big(g^{1/a_{\rho(GL,ML_k)}\cdot b}\big)^{-h_{2,i}} \cdot H_2(GL)^{r'}, \\
K_1 &= \big(H_1(GL)H_2(GL)^{ML_j}\big)^r \\
&= \Big(\prod_{\forall ML_k \in \mathbf{RevSet}(GL)} (g^{a_{\rho(GL,ML_k)}})^{-ML_k} \cdot g^{h_{1,i}} \cdot \big(\prod_{\forall ML_k \in \mathbf{RevSet}(GL)} g^{a_{\rho(GL,ML_k)}} \cdot g^{h_{2,i}}\big)^{ML_j}\Big)^{-b/a_{\rho(GL,ML_j)}+r'} \\
&= \prod_{\forall ML_k \in \mathbf{RevSet}(GL)\setminus\{ML_j\}} \big(g^{a_{\rho(GL,ML_k)}/a_{\rho(GL,ML_j)}\cdot b}\big)^{ML_k - ML_j} \cdot \big(g^{1/a_{\rho(GL,ML_j)}\cdot b}\big)^{-h_{1,i}-h_{2,i}ML_j} \cdot \\
&\quad \big(H_1(GL)H_2(GL)^{ML_j}\big)^{r'}, \\
K_2 &= g^{-r} = g^{b/a_{\rho(GL,ML_j)}-r'} = g^{1/a_{\rho(GL,ML_j)}\cdot b} g^{-r'}.
\end{aligned}$$

Note that it cannot create a private key for (GL, ML) such that $(GL = GL') \wedge (ML \neq ML')$ since the element g^b cannot be removed because of $\mathbf{RevSet}(GL') \setminus \{ML_j\} = \emptyset$. The challenge ciphertext is also correctly distributed since it satisfies the following equation

$$\begin{aligned}
C_0 &= e(g,g)^{\alpha t} M_\gamma^* = e(g,g)^{bc} M_\gamma^*, \\
C_1 &= g^t = g^c, \\
C_2 &= (H_1(GL^*)H_2(GL^*)^{ML^*})^t = \big((g^{a_{\rho(GL^*,ML^*)}})^{-ML^*} g^{h_1^*} \cdot (g^{a_{\rho(GL^*,ML^*)}} g^{h_2^*})^{ML^*}\big)^c = (g^c)^{h_1^* + h_2^* ML^*}.
\end{aligned}$$

Finally, we analyze the success probability of the above simulation. Let Good be the event that the simulator successfully guesses the challenge labels. We have that $\Pr[\mathsf{Good}] \geq \frac{1}{U_g \cdot U_m}$. If the event Good occurs, then the simulator does not abort. Therefore, the success probability of the simulation is bounded by $\frac{1}{U_g \cdot U_m}$. This completes our proof. □

3.4 Discussions

Fast Decryption. The simple decryption algorithm of our SRE scheme requires three pairing operations and a one exponentiation operation. We can improve the performance of the decryption algorithm by modifying the computation of the algorithm as $M = C_0 \cdot e(C_1, K_0^{-1} K_1^{1/(ML'-ML)}) \cdot e(C_2, K_2^{1/(ML'-ML)})$. In this case, the decryption algorithm just consists of two pairing operations and two exponentiation operations.

Chosen-Ciphertext Security. The indistinguishability under a chosen-ciphertext attack (IND-CCA) is similar to the indistinguishability under a chosen-plaintext attack (IND-CPA) except that an adversary is allowed to request decryption queries on ciphertexts. To provide the security of IND-CCA, we can use the transformation of Fujisaki and Okamoto [13] since our scheme is proven in the random oracle model.

Removing Random Oracles. The proposed SRE scheme is only secure when two hash functions H_1 and H_2 are modeled as random oracles. We can easily remove the random oracles by simply selecting random group elements h_i and w_i for $H_1(GL_i)$ and $H_2(GL_i)$ in the public key since the set of group labels is fixed and the total number of group labels is a polynomial number in a security parameter. However, the public key size of this method is quite large.

4 Subset Cover Framework

In this section, we define the subset cover framework and describe the subset difference (SD) scheme and the layered subset difference (LSD) scheme.

4.1 Definitions

The subset cover framework, introduced by Naor, Naor, and Lotspiech [21], is a general methodology to construct efficient revocation systems. They constructed symmetric-key broadcast encryption schemes by combining their subset cover framework that includes a key assignment method with a symmetric-key encryption scheme. In this paper, we define a subset cover scheme by excluding the key assignment method from the subset cover framework since we apply the subset cover scheme in public-key settings.

In the subset cover scheme, a center first defines a collection $\mathcal{S}$ of subsets $S_1, \ldots, S_w$ such that $S_i \subseteq \mathcal{N}$ where $\mathcal{N}$ is the set of all users. A user $u \in \mathcal{N}$ is assigned a private set PV_u that consists of subsets S_i associated with u by the center. The user may obtain the real private keys that are related with the private set PV_u from the center by running the key generation algorithm of symmetric-key encryption (SKE) or public-key encryption (PKE). A sender finds a covering set CV_R that is a partition of the non-revoked users $\mathcal{N} \setminus R$ into disjoint subsets $S_{i_1}, \ldots, S_{i_m}$. The sender may construct the ciphertext of broadcast encryption by running the encryption algorithm of SKE or PKE. A receiver can find matching sets from the covering set and the private set by running the matching algorithm. The receiver may recover a message from the ciphertext by running the decryption algorithm of SKE or PKE. The following is the syntax of the subset cover scheme.

Definition 4.1 (Subset Cover). *A subset cover scheme for the set $\mathcal{N} = \{1, \ldots, N\}$ of users consists of four PPT algorithms **Setup**, **Assign**, **Cover**, and **Match**, which are defined as follows:*

***Setup**(N). The setup algorithm takes as input the maximum number N of users and outputs a collection $\mathcal{S}$ of subsets $S_1, \ldots, S_w$ where $S_i \subseteq \mathcal{N}$.*

***Assign**($\mathcal{S}, u$). The assigning algorithm takes as input the collection $\mathcal{S}$ and a user $u \in \mathcal{N}$, and outputs a private set $PV_u = \{S_{j_1}, \ldots, S_{j_n}\}$ that is associated with the user u.*

Cover$(\mathcal{S}, R)$*. The covering algorithm takes as the collection* $\mathcal{S}$ *and a revoked set* $R \subset \mathcal{N}$ *of users, and outputs a covering set* $CV_R = \{S_{i_1} \ldots, S_{i_m}\}$ *that is a partition of the non-revoked users* $\mathcal{N} \setminus R$ *into disjoint subsets* $S_{i_1}, \ldots, S_{i_m}$ *such that* $\mathcal{S} \setminus R = \bigcup_{k=1}^{m} S_{i_k}$*.*

Match(CV_R, PV_u)*. The matching algorithm takes as input a covering set* $CV_R = \{S_{i_1}, \ldots, S_{i_m}\}$ *and a private set* $PV_u = \{S_{j_1}, \ldots, S_{j_n}\}$ *of a user u. It outputs* $(S_{i_k}, S_{j_{k'}})$ *such that* $S_{i_k} \in CV_R$*,* $u \in S_{i_k}$*, and* $S_{j_{k'}} \in PV_u$*, or it outputs* $\perp$*.*

The correctness property of subset cover is defined as follows: For all $\mathcal{S}$ *generated by* ***Setup****, all* PV_u *generated by* ***Assign****, and any R, it is required that:*

- *If* $u \notin R$*, then* ***Match***$(\textbf{Cover}(\mathcal{S}, R), PV_u) = (S_{i_k}, S_{j_{k'}})$*.*
- *If* $u \in R$*, then* ***Match***$(\textbf{Cover}(\mathcal{S}, R), PV_u) = \perp$*.*

4.2 Full Binary Tree

A full binary tree $\mathcal{T}$ is a tree data structure where each node except the leaf nodes has two child nodes. Let N be the number of leaf nodes in $\mathcal{T}$. The number of all nodes in $\mathcal{T}$ is $2N-1$ and for any $1 \le i \le 2N-1$ we denote by v_i a node in $\mathcal{T}$. The depth d_i of a node v_i is the length of the path from the root node to the node. The root node is at depth zero. The depth of $\mathcal{T}$ is the length of the path from the root node to a leaf node. A level of $\mathcal{T}$ is a set of all nodes at given depth. For any node $v_i \in \mathcal{T}$, T_i is defined as a subtree that is rooted at v_i. For any two nodes $v_i, v_j \in \mathcal{T}$ such that v_j is a descendant of v_i, $T_{i,j}$ is defined as a subtree $T_i - T_j$, that is, all nodes that are descendants of v_i but not v_j. For any node $v_i \in \mathcal{T}$, S_i is defined as the set of leaf nodes in T_i. Similarly, $S_{i,j}$ is defined as the set of leaf nodes in $T_{i,j}$, that is, $S_{i,j} = S_i \setminus S_j$.

For any node $v_i \in \mathcal{T}$, L_i is defined as an identifier that is a fixed and unique string. The identifier of each node in the tree is assigned as follows: Each edge in the tree is assigned with 0 or 1 depending on whether the edge is connected to its left or right child node. The identifier L_i of a node v_i is defined as the bitstring obtained by reading all the labels of edges in the path from the root node to the node v_i. We define $ID(v_i)$ be a mapping from a node v_i to an identifier L_i. We also define $ID(T_i)$ be a mapping from a subtree T_i to the identifier L_i of the node v_i and $ID(T_{i,j})$ be a mapping from a subtree $T_{i,j}$ to a tuple (L_i, L_j) of identifiers. Similarly, we can define $ID(S_i) = ID(T_i)$ and $ID(S_{i,j}) = ID(T_{i,j})$.

For a full binary tree $\mathcal{T}$ and a subset R of leaf nodes, $ST(\mathcal{T}, R)$ is defined as the Steiner Tree induced by the set R and the root node, that is, the minimal subtree of $\mathcal{T}$ that connects all the leaf nodes in R and the root node. we simply denote $ST(\mathcal{T}, R)$ by $ST(R)$.

4.3 SD Scheme

The subset difference (SD) scheme is the subset cover scheme proposed by Naor et al. [21]. We describe the SD scheme with a slight modification for the integration with our SRE scheme.

SD.Setup(N**):** This algorithm takes as input the maximum number N of users. Let $N = 2^n$ for simplicity. It first sets a full binary tree $\mathcal{T}$ of depth n. Each user is assigned to a different leaf node in $\mathcal{T}$. The collection $\mathcal{S}$ of SD is the set of all subsets $\{S_{i,j}\}$ where $v_i, v_j \in \mathcal{T}$ and v_j is a descendant of v_i. It outputs the full binary tree $\mathcal{T}$.

SD.Assign($\mathcal{T}, u$**):** This algorithm takes as input the tree $\mathcal{T}$ and a user $u \in \mathcal{N}$. Let v_u be the leaf node of $\mathcal{T}$ that is assigned to the user u. Let $(v_{k_0}, v_{k_1}, \ldots, v_{k_n})$ be the path from the root node v_{k_0} to the leaf node

$v_{k_n} = v_u$. It first sets a private set PV_u as an empty one. For all $i, j \in \{k_0, k_1, \ldots, k_n\}$ such that v_j is a descendant of v_i, it adds the subset $S_{i,j}$ defined by two nodes v_i and v_j in the path into PV_u. It outputs the private set $PV_u = \{S_{i,j}\}$.

SD.Cover($\mathcal{T}, R$): This algorithm takes as input the tree $\mathcal{T}$ and a revoked set R of users. It first sets a subtree T as $ST(R)$, and then it builds a covering set CV_R iteratively by removing nodes from T until T consists of just a single node as follows:

1. It finds two leaf nodes v_i and v_j in T such that the least-common-ancestor v of v_i and v_j does not contain any other leaf nodes of T in its subtree. Let v_l and v_k be the two child nodes of v such that v_i is a descendant of v_l and v_j is a descendant of v_k. If there is only one leaf node left, it makes $v_i = v_j$ to the leaf node, v to be the root of T and $v_l = v_k = v$.
2. If $v_l \neq v_i$, then it adds the subset $S_{l,i}$ to CV_R; likewise, if $v_k \neq v_j$, it adds the subset $S_{k,j}$ to CV_R.
3. It removes from T all the descendants of v and makes v a leaf node.

It outputs the covering set $CV_R = \{S_{i,j}\}$.

SD.Match(CV_R, PV_u): This algorithm takes input as a covering set $CV_R = \{S_{i,j}\}$ and a private set $PV_u = \{S'_{i,j}\}$. It finds two subsets $S_{i,j}$ and $S'_{i',j'}$ such that $S_{i,j} \in CV_R$, $S'_{i',j'} \in PV_u$, $i = i'$, $d_j = d_{j'}$, and $j \neq j'$ where d_j is the depth of v_j. If it found two subsets, then it outputs $(S_{i,j}, S'_{i',j'})$. Otherwise, it outputs $\perp$.

The correctness of the SD scheme is easy to show. We first show that if $u \notin R$, then the matching algorithm finds two subsets that meet the conditions. A covering set CV_R contains only one subset $S_{i,j}$ such that $v_u \in S_{i,j}$ if $u \notin R$ since the covering algorithm outputs disjoint subsets that cover the non-revoked users $\mathcal{N} \setminus R$. The subset $S_{i,j}$ is represented by two nodes v_i and v_j where v_i is an ancestor of a leaf node v_u and v_j is not an ancestor of the leaf node v_u since $v_u \in S_{i,j}$. A private set PV_u for the user u contains all subsets $S'_{i',j'}$ that are represented by two nodes $v_{i'}$ and $v_{j'}$ where $v_{i'}$ and $v_{j'}$ are nodes in the path from the root node to the leaf node v_u. That is, $v_{i'}$ and $v_{j'}$ of any $S'_{i',j'}$ are ancestors of v_u. Therefore, the matching algorithm can find two subsets $S_{i,j} \in CV_R$ and $S'_{i',j'} \in PV_u$ such that $v_u \in S_{i,j}$, $v_i = v_{i'}$, $d_j = d_{j'}$, and $v_j \neq v_{j'}$ since v_j is not an ancestor of v_u where d_j is the depth of v_j. We next show that if $u \in R$, then any algorithm cannot find two subsets that meet the conditions. Suppose there exists two subsets $S_{i,j} \in CV_R$ and $S'_{i',j'} \in PV_u$ such that $v_i = v_{i'}, d_j = d_{j'}$, and $v_j \neq v_{j'}$. In this case, v_i is an ancestor of v_u but v_j is not an ancestor of v_u. This means that $v_u \in S_{i,j}$. However, this contradicts the condition $u \in R$. Therefore, if $u \in R$, any algorithm cannot find two subsets.

Lemma 4.2 ([21]). *Let N be the number of leaf nodes in a full binary tree and r be the size of a revoked set. In the SD scheme, the size of a private set is $O(\log^2 N)$ and the size of a covering set is at most $2r - 1$.*

4.4 LSD Scheme

The layered subset difference (LSD) scheme is the subset cover scheme proposed by Halevy and Shamir [18]. The LSD scheme can reduce the size of a private set that is given to a user by increasing the size of a covering set. We describe the LSD scheme with a slight modification for the integration with our SRE scheme.

LSD.Setup(N): This algorithm takes as input the maximum number N of users. Let $N = 2^n$ for simplicity. It first sets a full binary tree $\mathcal{T}$ of depth n. Each user is assigned to a different leaf node in $\mathcal{T}$. Some levels of $\mathcal{T}$ are defined as "special". The root node is at a special level, and every level of depth $k\sqrt{\log N}$ as special for $k = 1$ to $\sqrt{\log N}$. The levels between and including adjacent special levels are defined as "layer". The collection $\mathcal{S}$ of LSD is the set of all subsets $\{S_{i,j}\}$ where $v_i, v_j \in \mathcal{T}$ and v_j is a descendant of v_i with the restriction that v_i and v_j belong to the same layer, or v_i is at a special level. It outputs the full binary tree $\mathcal{T}$.

LSD.Assign($\mathcal{T}, u$): This algorithm takes as input the tree $\mathcal{T}$ and a user index $u \in \mathcal{N}$. Let v_u be the leaf node of $\mathcal{T}$ that is assigned to the user u. Let $(v_{k_0}, v_{k_1}, \ldots, v_{k_n})$ be the path from the root node v_{k_0} to the leaf node $v_{k_n} = v_u$. It first sets a private set PV_u as an empty one. For all $i, j \in \{k_0, \ldots, k_n\}$ such that v_j is a descendant of v_i, it proceeds as follows: If v_i is at special level, then it adds $S_{i,j}$ into PV_u. Otherwise, that is v_i is at layer level, it adds $S_{i,j}$ into PV_u if v_i and v_j are at the same layer. It outputs a private set $PV_u = \{S_{i,j}\}$.

LSD.Cover($\mathcal{T}, R$): This algorithm takes as input the tree $\mathcal{T}$ and a revoked set R of users. It first finds a covering set CV'_R by running the covering algorithm of the SD scheme. For all $S_{i,j} \in CV'_R$, it proceeds as follows: If v_i and v_j are not at the same layer, then it splits $S_{i,j}$ into $S_{i,k}$ and $S_{k,j}$ where v_k is at special level and v_i, v_k, v_j are nodes which occur in this order on a path from v_i to v_j in the tree. It adds $S_{i,k}, S_{k,j}$ into CV_R. Otherwise, that is v_i and v_j are at the same layer, then it adds $S_{i,j}$ into CV_R. It outputs the covering set $CV_R = \{S_{i,j}\}$.

LSD.Match(CV_R, PV_u): This algorithm is the same as the matching algorithm of the SD scheme.

Lemma 4.3 ([18]). *Let N be the number of leaf nodes in a full binary tree and r be the size of a revoked set. In the LSD scheme, the size of a private set is $O(\log^{1.5} N)$ and the size of a covering set is at most $4r - 2$.*

5 Revocation Encryption

In this section, we first propose a public-key revocation encryption (PKRE) scheme by combining the SRE scheme and the subset cover scheme, and then we prove its security.

5.1 Definitions

Broadcast encryption, introduced by Fiat and Naor [12], is an encryption method that can efficiently transmit a ciphertext to a receiver set S of users. Revocation encryption is a slight variation of broadcast encryption such that the encryption algorithm is specified by a revoked set R instead of a receiver set S[1]. Public-key revocation encryption (PKRE) is a special type of RE such that anyone can create a ciphertext for all users except a revoked set R of users by using a public key [11, 20]. The following is the syntax of PKRE.

Definition 5.1 (Public-Key Revocation Encryption). *A public-key revocation encryption (PKRE) scheme for the set $\mathcal{N} = \{1, \ldots, N\}$ of users consists of four algorithms **Setup**, **GenKey**, **Encrypt**, and **Decrypt**, which are defined as follows:*

***Setup**($1^\lambda, N$). The setup algorithm takes as input a security parameter 1^λ and the maximum number N of users. It outputs a master key MK and a public key PK.*

[1]In the public-key setting, RE is equivalent to broadcast encryption since a receiver set can be specified as $S = \mathcal{N} \setminus R$ where $\mathcal{N}$ is the set of all users in the system and the size of $\mathcal{N}$ is a polynomial number in a security parameter. However, in the identity-based setting, RE is not equivalent to BE since the size of all users in the system is an exponential number in a security parameter.

***GenKey**(u, MK, PK). The key generation algorithm takes as input a user's index $u \in \mathcal{N}$, the master key MK, and the public key PK. It outputs a private key SK_u for the user u.*

***Encrypt**(R, M, PK). The encryption algorithm takes as input a revoked set R of users such that $R \subseteq \mathcal{N}$, a message $M \in \mathcal{M}$, and the public key PK. It outputs a ciphertext CT_R for R and M.*

***Decrypt**(CT_R, SK_u, PK). The decryption algorithm takes as input a ciphertext CT_R for a revoked set R, a private key SK_u for an index u, and the public key PK. It outputs an encrypted message M or $\perp$.*

*The correctness property of PKRE is defined as follows: For all MK, PK generated by **Setup**, all u, S, any SK_u generated by **GenKey**, and any M, it is required that*

- *If $u \notin R$, then $\textbf{Decrypt}(\textbf{Encrypt}(R,M,PP), SK_u, PP) = M$.*
- *If $u \in R$, then $\textbf{Decrypt}(\textbf{Encrypt}(R,M,PP), SK_u, PP) = \perp$ with all but negligible probability.*

The security property of PKRE is defined as indistinguishability. The indistinguishability game of PKRE can be easily defined by slightly modifying the indistinguishability game of PKBE [5, 21]. In this game, an adversary is first given a target public key, and then he may adaptively request many private keys of users. In the challenge step, the adversary submits a challenge revoked set R^* and two challenge messages M_0^*, M_1^*, and then he is given a challenge ciphertext. Finally, the adversary outputs a guess for the random coin that is used to create the challenge ciphertext. If the guess of the adversary is correct, then he wins. The following is the formal definition of indistinguishability.

Definition 5.2 (Indistinguishability)**.** *The indistinguishability property of PKRE under a chosen plaintext attack is defined in terms of the following game between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$:*

1. ***Setup**: $\mathcal{C}$ runs **Setup**($1^\lambda, N$) to generate a master key MK and a public key PK. It keeps MK to itself and gives PK to $\mathcal{A}$.*
2. ***Query**: $\mathcal{A}$ may adaptively request private keys for users $u_1, \ldots, u_q \in \mathcal{N}$. In response, $\mathcal{C}$ gives the corresponding private keys $SK_{u_1}, \ldots, SK_{u_q}$ to $\mathcal{A}$ by running **GenKey**(u_i, MK, PP).*
3. ***Challenge**: $\mathcal{A}$ submits a challenge revoked set R^* of users and two messages M_0^*, M_1^* with the equal length subject to the restriction: for all u_i of private key queries, $u_i \in R^*$. $\mathcal{C}$ flips a random coin $\gamma \in \{0,1\}$ and gives the challenge ciphertext CT^* to $\mathcal{A}$ by running **Encrypt**(R^*, M_γ^*, PK).*
4. ***Guess**: $\mathcal{A}$ outputs a guess $\gamma' \in \{0,1\}$ of γ, and wins the game if $\gamma = \gamma'$.*

The advantage of $\mathcal{A}$ is defined as $\textbf{Adv}_{\mathcal{A}}^{PKRE}(\lambda) = \left|\Pr[\gamma = \gamma'] - \frac{1}{2}\right|$ where the probability is taken over all the randomness of the game. A PKRE scheme is indistinguishable under a chosen plaintext attack if for all PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the above game is negligible in the security parameter λ.

5.2 Generic Construction

The basic idea of our PKRE scheme is to combine the SD scheme and the SRE scheme that is a special type of public-key encryption (PKE). The idea of combining the SD scheme with a PKE scheme was introduced by Dodis and Fazio [11]. Dodis and Fazio showed that the key assignment method of Naor et al. [21] for the SD scheme can be mimicked by using the delegation property of HIBE. In contrast to the method of Dodis and Fazio, we show that a subset $S_{i,j}$ in the SD scheme can be easily mapped to the group and member labels

(GL, ML) of the SRE scheme by using the revocation property of the SRE scheme that can revoke a single member in a group. That is, a subset $S_{i,j}$ in the SD scheme is defined as the set of leaf nodes that belong to T_i but not belong to T_j where T_i and T_j are subtrees with root nodes v_i and v_j respectively. This subset $S_{i,j}$ is represented by two nodes v_i and v_j that have labels L_i and L_j respectively. To map the subset $S_{i,j}$ to labels (GL, ML), we define a group GL as the set of nodes in T_i at the same level as the node v_j and define a revoked member ML as the node v_j.

Before presenting our PKRE scheme, we first define the universe $\mathcal{U}$ of SRE that is derived from a full binary tree $\mathcal{T}$ as follows: Let T_i be a subtree of $\mathcal{T}$ that is rooted at v_i. A single group in $\mathcal{U}$ is defined as a set of nodes that are in the same level of T_i except the level of v_i. Suppose that the tree $\mathcal{T}$ has the number N of leaf nodes. In this case, the maximum number of groups in $\mathcal{U}$ is $N \log N$ and the maximum number of members in a groups is N since the number of internal nodes is $N-1$ and the maximum depth of each subtree is $\log N - 1$. The subset $S_{i,j}$ of the SD scheme that uses $\mathcal{T}$ is easily converted to the labels $(GL = L_i \| d_j, ML = L_j)$ of the SRE scheme where (L_i, L_j) is the identifier of $S_{i,j}$ and d_j is the depth of L_j.

Our generic PKRE scheme from an SRE scheme for the set $\mathcal{N} = \{1, \ldots, N\}$ of users is described as follows:

PKRE.Setup($1^\lambda, N$): This algorithm first defines a full binary tree $\mathcal{T}$ by running **SD.Setup**(N). Next, it obtains MK_{SRE} and PK_{SRE} by running **SRE.Setup**$(1^\lambda, \mathcal{U})$ where $\mathcal{U}$ is defined from $\mathcal{T}$. It outputs a master key $MK = MK_{SRE}$ and a public key as $PK = (\mathcal{T}, PK_{SRE})$.

PKRE.GenKey(u, MK, PK): This algorithm takes as input a user $u \in \mathcal{N}$, the master key MK, and the public key PK. It first obtains a private set $PV_u = \{S_{i,j}\}$ by running **SD.Assign**$(\mathcal{T}, u)$. Let d_j be the depth of a node v_j associated with L_j. For all $S_{i,j} \in PV_u$, it obtains (L_i, L_j) by applying $ID(S_{i,j})$ and computes $SK_{SRE, S_{i,j}}$ by running **SRE.GenKey**$((L_i \| d_j, L_j), MK_{SRE}, PK_{SRE})$. It outputs a private key as $SK = (PV_u, \{SK_{SRE, S_{i,j}}\}_{S_{i,j} \in PV_u})$.

PKRE.Encrypt(R, M, PK): This algorithm takes as input a revoked set $R \subseteq \mathcal{N}$, a message $M \in \mathbb{G}_T$, and the public key PK. It first finds a covering set $CV_R = \{S_{i,j}\}$ by running **SD.Cover**$(\mathcal{T}, R)$. Let d_j be the depth of a node v_j associated with L_j. For all $S_{i,j} \in CV_R$, it obtains (L_i, L_j) by applying $ID(S_{i,j})$ and computes $CT_{SRE, S_{i,j}}$ by running **SRE.Encrypt**$((L_i \| d_j, L_j), M, PK_{SRE})$. It outputs a ciphertext as $CT = (CV_R, \{CT_{SRE, S_{i,j}}\}_{S_{i,j} \in CV_R})$.

PKRE.Decrypt(CT, SK, PK): This algorithm takes as input a ciphertext CT, a private key SK, and the public key PK. If $u \in R$, then it outputs $\perp$. Otherwise, it first obtains $(S_{i,j}, S'_{i',j'})$ by running **SD.Match**(CV_R, PV_u). It outputs a message M by running **SRE.Decrypt**$(CT_{SRE, S_{i,j}}, SK_{SRE, S'_{i',j'}}, PK_{SRE})$.

The correctness of the above PKRE scheme easily follows the correctness of the SD scheme and that of the SRE scheme. If $u \notin R$, then a user u can obtain two subsets $S_{i,j} \in CV_R$ and $S'_{i',j'} \in PV_u$ from a ciphertext CT and his private key SK such that $i = i', d_j = d_{j'}$, and $j \neq j'$ from the correctness of the SD scheme. Next, he can derive two labels $(GL = L_i \| d_j, ML = L_j)$ and $(GL' = L_{i'} \| d_{j'}, ML' = L_{j'})$ for the SRE scheme from the two subsets $S_{i,j}$ and $S'_{i',j'}$ where $(L_i, L_j) = ID(S_{i,j})$ and $(L_{i'}, L_{j'}) = ID(S'_{i',j'})$. Note that $L_i = L_{i'}, d_j = d_{j'}$, and $L_j \neq L_{j'}$. Therefore, he can obtains a message M from the correctness of the SRE scheme since $GL = GL'$ and $ML \neq ML'$. If $u \in R$, then a user u cannot obtain two subsets $S_{i,j} \in CV_R$ and $S'_{i',j} \in PV_u$ such that $i = i', d_j = d_{j'}$, and $j \neq j'$ from the correctness of the SD scheme. Note that the correctness property is only satisfied when an honest user simply runs the decryption algorithm of our PKRE scheme.

5.3 Security Analysis

Theorem 5.3. *The generic PKRE scheme is indistinguishable under a chosen plaintext attack if the SRE scheme is indistinguishable under a chosen plaintext attack.*

Proof. The basic idea of this proof is to convert the challenge ciphertext of the PKRE scheme from an encryption on a message M_0^* to an encryption on a message M_1^* by using hybrid games that change each ciphertext component of the SRE scheme from an encryption on M_0^* to an encryption on M_1^*. If an adversary cannot distinguish the changes of each ciphertext component of the SRE scheme with more than a non-negligible probability, then he also cannot distinguish the changes of the challenge ciphertext of the PKRE scheme with more than a non-negligible probability since the number of hybrid games is just polynomial.

Suppose that CV_{R^*} is the covering set of the challenge revoked set R^* and the size of CV_{R^*} is w. The challenge ciphertext is described as $CT^* = (CV_R, CT_{SRE,1}, \dots, CT_{SRE,w})$. The hybrid games $\mathbf{G}_0, \dots, \mathbf{G}_i, \dots, \mathbf{G}_w$ for the security proof are defined as follows:

Game $\mathbf{G}_0$ In this game, all ciphertext components $CT_{SRE,j}$ of the challenge ciphertext are encryption on the message M_0^*. That is, the challenge ciphertext CT^* is an encryption on the message M_0^*. Note that this game is the original security game except that the challenge bit γ is fixed to 0.

Game $\mathbf{G}_h$ This game is almost identical to the game $\mathbf{G}_{h-1}$ except the ciphertext component $CT_{SRE,h}$ since $CT_{SRE,h}$ in this game is an encryption on the message M_1^*. Specifically, in this game, the ciphertext component $CT_{SRE,j}$ for $j \le h$ is an encryption on the message M_1^* and the ciphertext component $CT_{SRE,j}$ for $h < j$ is an encryption on the message M_0^*.

Game $\mathbf{G}_w$ In this game, all ciphertext components $CT_{SRE,j}$ of the challenge ciphertext are encryption on the message M_1^*. That is, the challenge ciphertext CT^* is an encryption on the message M_1^*. Note that this game is the original security game except that the challenge bit γ is fixed to 1.

Let $S_{\mathcal{A}}^{G_h}$ be the event that $\mathcal{A}$ outputs 0 in $\mathbf{G}_h$. In Lemma 5.4, we prove that it is hard for $\mathcal{A}$ to distinguish $\mathbf{G}_{h-1}$ from $\mathbf{G}_h$ if the SRE scheme is secure. Thus, we have that

$$\begin{aligned}\Pr[S_{\mathcal{A}}^{G_0}] - \Pr[S_{\mathcal{A}}^{G_w}] &= \Pr[S_{\mathcal{A}}^{G_0}] + \sum_{h=1}^{w-1} \big(\Pr[S_{\mathcal{A}}^{G_h}] - \Pr[S_{\mathcal{A}}^{G_h}]\big) - \Pr[S_{\mathcal{A}}^{G_w}] \\ &\le \sum_{h=1}^{w} \big|\Pr[S_{\mathcal{A}}^{G_{h-1}}] - \Pr[S_{\mathcal{A}}^{G_h}]\big| \le 2w \cdot \mathbf{Adv}_{\mathcal{B}}^{SRE}(\lambda).\end{aligned}$$

Finally, we obtain the following inequality relation as

$$\begin{aligned}\mathbf{Adv}_{\mathcal{A}}^{PKRE}(\lambda) &= \big|\Pr[\gamma=0]\cdot\Pr[\gamma=\gamma'|\gamma=0] + \Pr[\gamma=1]\cdot\Pr[\gamma=\gamma'|\gamma=1] - \frac{1}{2}\big| \\ &= \big|\frac{1}{2}\cdot\Pr[\gamma'=0|\gamma=0] + \frac{1}{2}\cdot(1-\Pr[\gamma'=0|\gamma=1]) - \frac{1}{2}\big| \\ &= \frac{1}{2}\cdot\big|\Pr[\gamma'=0|\gamma=0] - \Pr[\gamma'=0|\gamma=1]\big| \\ &\le \frac{1}{2}\cdot\big|\Pr[S_{\mathcal{A}}^{G_0}] - \Pr[S_{\mathcal{A}}^{G_w}]\big| \le w\cdot\mathbf{Adv}_{\mathcal{B}}^{SRE}(\lambda).\end{aligned}$$

Note that we already have $\mathbf{Adv}^{SRE}(\lambda) \le N^2 \log N \cdot \mathbf{Adv}^{q\text{-}SMEBDH}(\lambda)$ from Theorem 3.3 since $U_g \le N \log N$ and $U_m \le N$. This completes our proof. □

Lemma 5.4. *If the SRE scheme is indistinguishable under a chosen plaintext attack, then no polynomial time adversary can distinguish between* $\mathbf{G}_{h-1}$ *and* $\mathbf{G}_h$ *with non-negligible advantage.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that distinguishes between $\mathbf{G}_{h-1}$ and $\mathbf{G}_h$ with non-negligible advantage. A simulator $\mathcal{B}$ that breaks the indistinguishability game of the SRE scheme is first given: a challenge public key PK'_{SRE}. Then $\mathcal{B}$ that interacts with $\mathcal{A}$ is described as follows:

Setup: $\mathcal{B}$ first sets a full binary tree $\mathcal{T}$ by running **SD.Setup**(N) and gives $PK = (\mathcal{T}, PK'_{SRE})$ to $\mathcal{A}$.

Query: $\mathcal{A}$ adaptively requests a private key query for a user $u \in \mathcal{N}$. $\mathcal{B}$ proceeds this query as follows:

1. It obtains a private set $PV_u = \{S_{i,j}\}$ by running **SD.Assign**$(\mathcal{T}, u)$.
2. For each $S_{i,j} \in PV_u$, it sets labels (GL, ML) from $S_{i,j}$ and requests a private key $SK_{SRE,S_{i,j}}$ for labels (GL, ML) to the key generation oracle that simulates **SRE.GenKey**.
3. It sets the private key $SK_u = \{SK_{SRE,S_{i,j}}\}$ and gives this to $\mathcal{A}$.

Challenge: $\mathcal{A}$ outputs a challenge revoked set R^* and two challenge messages M_0^*, M_1^* subject to the restrictions. It sets $\gamma = 0$ and proceeds as follows:

1. It obtains a covering set $CV_{R^*} = \{S_{i_1,j_1}, \dots, S_{i_w,j_w}\}$ by running **SD.Cover**$(\mathcal{T}, R^*)$.
2. For $1 \le k \le h-1$, it computes $CT_{SRE,S_{i_k,j_k}}$ by running **SRE.Encrypt**$((L_{i_k} \| d_{j_k}, L_{j_k}), M_1^*, PK'_{SRE})$ where $(L_{i_k}, L_{j_k}) = ID(S_{i_k,j_k})$ and d_{j_k} is the depth of the node v_{j_k}.
3. For $k = h$, it gives challenge labels $GL' = L_{i_h} \| d_{j_h}, ML' = L_{j_h}$ and challenge messages $M'_0 = M_0^*, M'_1 = M_1^*$ to the challenge oracle that simulates **SRE.Encrypt**, and receives CT'_{SRE}. It sets $CT_{SRE,h} = CT'_{SRE}$.
4. For $h+1 \le k \le w$, it computes $CT_{SRE,S_{i_k,j_k}}$ by running **SRE.Encrypt**$((L_{i_k} \| d_{j_k}, L_{j_k}), M_0^*, PK'_{SRE})$ where $(L_{i_k}, L_{j_k}) = ID(S_{i_k,j_k})$ and d_{j_k} is the depth of the node v_{j_k}.
5. It gives a challenge ciphertext $CT = (CV_{R^*}, CT_{SRE,S_{i_1,j_1}}, \dots, CT_{SRE,S_{i_w,j_w}})$ to $\mathcal{A}$.

Guess: Finally, $\mathcal{A}$ outputs a bit γ'. $\mathcal{B}$ sets $c' = \gamma'$ and outputs c'.

To finish the proof, we first show that the distribution of the above simulation is correct. It is easy to check that the public key and the private keys are correctly distributed. Let c be the challenger oracle's random bit of the SRE scheme. If $c = 0$, then CT'_{SRE} is the encryption of labels (GL', ML') and a message M_0^*. If $c = 1$, then CT'_{SRE} is the encryption of labels (GL', ML') and a message M_1^*. Thus, the challenge ciphertext is the same as $\mathbf{G}_{h-1}$ if $c = 0$. Similarly, we obtain that the challenge ciphertext is the same as $\mathbf{G}_h$ if $c = 1$.

We next show that the above simulation satisfies the restriction of the SRE indistinguishability game. The restriction of the SRE indistinguishability game is that for all labels (GL_i, ML_i) of SRE private keys, it is required that $(GL_i \ne GL') \vee (ML_i = ML')$ where GL', ML' are the labels of the challenge SRE ciphertext. The restriction of the PKRE indistinguishability game is that for all user index u of PKRE private keys, it is required that $u \in R^*$ where R^* is the revoked set of users in the challenge PKRE ciphertext. The above simulator honestly uses the algorithms of the SD scheme when it generates private keys and challenge ciphertext for the PKRE scheme. If $u \in R$, there are no two subsets $S_{i,j} \in CV_R, S'_{i',j'} \in PV_u$ such that $(v_i = v_{i'}) \wedge (d_j = d_{j'}) \wedge (v_j \ne v_{j'})$ from the correctness of the SD scheme. Thus if $u \in R$, then any two subset $S_{i,j} \in CV_R, S'_{i',j'}$ satisfy $(v_i \ne v_{i'}) \vee (d_j \ne d_{j'}) \vee (v_j = v_{j'})$. If $(v_i \ne v_{i'}) \vee (d_j \ne d_{j'})$, then $GL \ne GL'$ since the group label is defined as $GL_i = L_i \| d_j$ where L_i is the identifier of v_i. If $(v_j = v_{j'})$, then $(ML = ML')$ since the member label is defined as $ML_j = L_j$ where L_j is the identifier of v_j. Therefore, we obtains the restriction $(GL_i \neq GL') \vee (ML_i = ML')$ of the SRE indistinguishability game from the restriction $u \in R^*$ of the PKRE indistinguishability game.

Therefore, we obtain the following equation

$$\begin{aligned}
\mathbf{Adv}_{\mathcal{B}}^{SRE}(\lambda) &= \Pr[c=0] \cdot \big|\Pr[c'=c|c=0] - \frac{1}{2}\big| + \Pr[c=1] \cdot \big|\Pr[c'=c|c=1] - \frac{1}{2}\big| \\
&= \frac{1}{2} \cdot \big|\Pr[\gamma'=0|c=0] - \frac{1}{2}\big| + \frac{1}{2} \cdot \big|\Pr[\gamma'=1|c=1] - \frac{1}{2}\big| \\
&= \frac{1}{2} \cdot \big|\Pr[\gamma'=0|c=0]\big| + \frac{1}{2} \cdot \big|(1 - \Pr[\gamma'=0|c=1])\big| \\
&\geq \frac{1}{2} \cdot (\Pr[\gamma'=0|c=0]) - \frac{1}{2} \cdot (\Pr[\gamma'=0|c=1]) \\
&= \frac{1}{2} \cdot (\Pr[\gamma'=0|\gamma=0, c=0]) - \frac{1}{2} \cdot (\Pr[\gamma'=0|\gamma=0, c=1]) \\
&= \frac{1}{2} \cdot \Pr[S_{\mathcal{A}}^{G_{h-1}}] - \frac{1}{2} \cdot \Pr[S_{\mathcal{A}}^{G_h}].
\end{aligned}$$

This completes our proof. □

5.4 Discussions

Efficiency. In our PKRE scheme, a public key consists of $O(1)$ group elements, a private key consists of $O(\log^2 N)$ group elements, and a ciphertext consists of $O(r)$ group elements where r is the size of a revoked set. Compared with the PKRE scheme of Dodis and Fazio [11], our PKRE scheme reduces the number of group elements in a public key and a private key by a factor of $\log N$. Additionally, the decryption algorithm of our PKRE scheme just requires one decryption operation of the SRE scheme that consists of two pairing operations and two exponentiation operations.

LSD Scheme. To construct a generic PKRE scheme, we combined our SRE scheme with the SD scheme. We can also combine our SRE scheme with the LSD scheme to construct a PKRE scheme since the LSD scheme is just a special case of the SD scheme. If the LSD scheme is used instead of the SD scheme, then the group elements of a private key can be reduced from $O(\log^2 N)$ to $O(\log^{1.5} N)$ by doubling the number of group elements in a ciphertext.

Chosen-Ciphertext Security. By combining an SRE scheme that provides the IND-CCA security and an one-time signature scheme that provides the strong unforgeability (i.e., an adversary is unable to forge a new signature on the previously signed message.), we can construct a PKRE scheme that achieves the IND-CCA security. That is, the encryption algorithm of PKRE first generates the public key OPK and the signing key OSK of one-time signature, and then it builds ciphertext components by running the encryption algorithm of SRE on $OVK \| M$ instead of M. At last, it creates a ciphertext by attaching an one-time signature on the ciphertext components by running the signing algorithm of one-time signature.

Trace and Revoke. Our generic PKRE scheme also provides the tracing property since it is derived from the subset cover framework of Naor et al. [21]. The detailed description of our PKTR scheme is given in Appendix A. Note that the trace and revoke scheme derived from the subset cover framework can only trace to a subset pattern in some colluding scenarios [19].

6 Conclusion

In this paper, we revisited the methodology of the subset cover framework to construct PKTR schemes, and introduced a new type of PKE named single revocation encryption (SRE). Inspired by the IBRE scheme of Lewko et al. [20], we proposed an efficient SRE scheme with the constant size of ciphertexts, private keys, and public keys, and proved its security in the random oracle model under q-type assumption. The SRE scheme may have independent interests. One notable advantage of our SRE scheme is that the PKTR scheme using our SRE scheme maintains the same efficiency parameter as the SD scheme (or the LSD scheme).

There are many interesting problems. The first one is to construct an efficient SRE scheme with short public key without random oracles. We showed that the random oracles in our SRE scheme can be removed. However, this approach has the problem of large public key size. The second one is to construct a PKTR scheme with shorter private key size. One possible approach is to use the Stratified SD (SSD) scheme of Goodrich et al. [17], but it is not yet known whether the SSD scheme can be applicable in the public-key setting.

References

[1] Michel Abdalla, Alexander W. Dent, John Malone-Lee, Gregory Neven, Duong Hieu Phan, and Nigel P. Smart. Identity-based traitor tracing. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC*, volume 4450 of *Lecture Notes in Computer Science*, pages 361–376. Springer, 2007.

[2] Olivier Billet and Duong Hieu Phan. Traitors collaborating in public: Pirates 2.0. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 189–205. Springer, 2009.

[3] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005.

[4] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[5] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2005.

[6] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 573–592. Springer, 2006.

[7] Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 211–220. ACM, 2006.

[8] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Yvo Desmedt, editor, *CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 257–270. Springer, 1994.

[9] Cécile Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 200–215. Springer, 2007.

[10] Cécile Delerablée, Pascal Paillier, and David Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *Pairing 2007*, volume 4575 of *Lecture Notes in Computer Science*, pages 39–59. Springer, 2007.

[11] Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In Joan Feigenbaum, editor, *DRM 2002*, volume 2696 of *Lecture Notes in Computer Science*, pages 61–80. Springer, 2002.

[12] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer, 1993.

[13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.

[14] Jun Furukawa and Nuttapong Attrapadung. Fully collusion resistant black-box traitor revocable broadcast encryption with short private keys. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP 2007*, volume 4596 of *Lecture Notes in Computer Science*, pages 496–508. Springer, 2007.

[15] Sanjam Garg, Abishek Kumarasubramanian, Amit Sahai, and Brent Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pages 121–130. ACM, 2010.

[16] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 171–188. Springer, 2009.

[17] Michael T. Goodrich, Jonathan Z. Sun, and Roberto Tamassia. Efficient tree-based revocation in groups of low-state devices. In Matthew K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 511–527. Springer, 2004.

[18] Dani Halevy and Adi Shamir. The lsd broadcast encryption scheme. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 47–60. Springer, 2002.

[19] Aggelos Kiayias and Serdar Pehlivanoglu. Pirate evolution: How to make the most of your traitor keys. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 448–465. Springer, 2007.

[20] Allison B. Lewko, Amit Sahai, and Brent Waters. Revocation systems with very small private keys. In *IEEE Symposium on Security and Privacy*, pages 273–285. IEEE Computer Society, 2010.

[21] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer, 2001.

[22] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. *Electronic Colloquium on Computational Complexity (ECCC)*, (043), 2002.

[23] Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In Yair Frankel, editor, *FC 2000*, volume 1962 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2000.

[24] Jong Hwan Park, Hee Jean Kim, H.-M. Sung, and Dong Hoon Lee. Public key broadcast encryption schemes with shorter transmissions. *IEEE Trans. Broadcast.*, 54(3):401–411, 2008.

[25] Jong Hwan Park and Dong Hoon Lee. Fully collusion-resistant traitor tracing scheme with shorter ciphertexts. *Designs Codes Cryptogr.*, 60(3):255–276, 2011.

[26] Jong Hwan Park, Hyun Sook Rhee, and Dong Hoon Lee. Fully collusion-resistant trace-and-revoke scheme in prime-order groups. *Journal of Communications and Networks*, 13(5):428–441, 2011.

[27] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer, 2009.

A Trace and Revoke

In this section, we propose a public-key trace and revoke (PKTR) scheme and prove its security. The proposed PKTR scheme essentially uses the well-known fact that the broadcast encryption of the subset cover framework can support the traitor tracing functionality. We describe our PKTR scheme that uses the traitor tracing algorithm of Naor et al. [21] for the completeness of this paper.

A.1 Definitions

Public-key trace and revoke (PKTR) is PKRE with an additional tracing algorithm that can find a traitor who leaked his private key to create a pirate decoder. Traitor tracing, introduced by Chor, Fiat, and Naor [8], is a mechanism that provides the tracing functionality such that a tracer who is given a pirate decoder can find at least one user who participated to create a pirate decoder by using his private key. PKTR is a mechanism that can find a user suspected to be a traitor and revoke the user permanently [6, 7, 21]. A PKTR scheme consists of the four algorithm of PKRE and one additional tracing algorithm that calls a pirate decoder as a black-box. The following is the syntax of PKTR.

Definition A.1 (Public-Key Trace and Revoke)**.** *A public-key trace and revoke (PKTR) scheme for the set $\mathcal{N} = \{1,\ldots,N\}$ of users consists of five algorithms **Setup**, **GenKey**, **Encrypt**, **Decrypt**, and **Trace**$^{\mathcal{D}}$, which are defined as follows:*

***Setup**$(1^{\lambda}, N)$. The setup algorithm takes as input a security parameter 1^{λ} and the maximum number N of users. It outputs a master key MK and a public key PK.*

***GenKey**(u, MK, PK). The key generation algorithm takes as input a user's index $u \in \mathcal{N}$, the master key MK, and the public key PK. It outputs a private key SK_u for the user u.*

***Encrypt**(R, M, PK). The encryption algorithm takes as input a set R of revoked users such that $R \subseteq \mathcal{N}$, a message $M \in \mathcal{M}$, and the public key PK. It outputs a ciphertext CT_R for R and M.*

***Decrypt**(CT_R, SK_u, PK). The decryption algorithm takes as input a ciphertext CT_R for a set R, a private key SK_u for an index u, and the public key PK. It outputs an encrypted message M or $\perp$.*

***Trace**$^{\mathcal{D}}(R, \varepsilon, PK)$. The tracing algorithm takes as input a revoked set R of users, a parameter ε, and the public key PK. It interacts with a pirate decoder $\mathcal{D}$ and outputs a set $T \subseteq \mathcal{N}$.*

*The correctness property of PKTR is defined as follows: For all MK, PK generated by **Setup**, all u, S, any SK_u generated by **GenKey**, and any M, it is required that*

- *If $u \notin R$, then **Decrypt**(**Encrypt**$(R, M, PP), SK_u, PP) = M$.*
- *If $u \in R$, then **Decrypt**(**Encrypt**$(R, M, PP), SK_u, PP) = \perp$ with all but negligible probability.*

The security property of PKTR consists of indistinguishability and traceability. The indistinguishability of PKTR is the same as the indistinguishability of PKRE. The traceability of PKTR is defined as the tracing game between a challenger and an adversary. In this game, the adversary is given a public key and obtains private keys of users. Finally, the adversary outputs a revoked set $R_{\mathcal{D}}$ and a pirate decoder $\mathcal{D}$. If the challenger fails to find a traitor by using the tracing algorithm, then the adversary wins. The following is the formal definition of traceability.

Definition A.2 (Traceability)**.** *The traceability property of PKTR is defined in terms of the following tracing game between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$:*

1. ***Setup**: $\mathcal{C}$ runs **Setup**($1^\lambda, N$) to generate a master key MK and a public key PK. It keeps MK to itself and gives PK to $\mathcal{A}$.*
2. ***Query**: $\mathcal{A}$ may adaptively request private keys for users $u_1, \ldots, u_q$. In response, $\mathcal{C}$ gives the corresponding private keys $SK_{u_1}, \ldots, SK_{u_q}$ to $\mathcal{A}$ by running **GenKey**(u_i, MK, PP). Let E be the total set of users whose private keys were obtained by $\mathcal{A}$.*
3. ***Output**: Finally, $\mathcal{A}$ outputs a revoked set $R_{\mathcal{D}}$ of users and a pirate decoder $\mathcal{D}$ which is a probabilistic algorithm that takes as input a ciphertext CT and outputs some message M.*
4. ***Trace**: $\mathcal{C}$ obtains a traitor set $T \subseteq \mathcal{N}$ by running **Trace**$^{\mathcal{D}}(R_{\mathcal{D}}, \varepsilon, PK)$.*

$\mathcal{A}$ wins the game if the following two conditions hold: (i) $\Pr[\mathcal{D}(\textbf{Encrypt}(R_{\mathcal{D}}, M, PK)) = M] \geq \varepsilon$ for a randomly chosen M and (ii) the set T is either empty or is not a subset of $E \setminus R_{\mathcal{D}}$. The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins this game. A PKTR scheme is traceable if for all PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the above game is negligible in the security parameter λ.

A.2 Construction

Naor et al. [21] showed that broadcast encryption of the subset cover framework can support the traitor tracing functionality. Our PKRE scheme also can support the tracing functionality since it also uses the subset cover framework. We describe the tracing algorithm of our PKTR scheme by using the tracing algorithm of Naor et al. [21, 22]. The basic idea of the tracing algorithm in the subset cover framework is that the exists a subset tracing procedure that can find a subset that contains a traitor. By using the subset tracing procedure, we can iteratively split the subset that contains a traitor into two disjoint subsets by using the bifurcation property of the subset cover framework, and then we can eventually find a subset that contains one traitor.

For the tracing algorithm in the subset cover framework, an efficient subset tracing procedure is required. The subset tracing procedure takes a partition $P = \{S_{i_1}, \ldots, S_{i,m}\}$ and a pirate decoder as inputs and outputs one of two outputs: (1) a symbol $\perp$ that indicates the pirate decoder cannot decrypt a ciphertext for the partition P or (2) a subset $S_{i'} \in P$ that contains a traitor. Naor et al. [21] showed that there exists an efficient subset tracing procedure in the subset cover framework. Note that their subset tracing procedure applies to not only the CS scheme and the SD scheme of Naor et al. [21], but also the LSD scheme of Halevy and Shamir [18].

The **Setup**, **GenKey**, **Encrypt**, and **Decrypt** algorithm of our PKTR scheme is the same as those of our generic PKRE scheme. The **Trace** algorithm of our PKTR scheme is described as follows:

PKTR.Trace$^{\mathcal{D}}(R, \varepsilon, PK)$**:** This algorithm takes as input a revoked set R, a value ε, and the public key PK.

1. First, it initializes a traitor set T to the empty one.
2. It iteratively performs the following steps:
   (a) It obtains a current partition P by running **SD.Cover**$(\mathcal{T}, R \cup T)$.
   (b) To find one traitor, it iteratively performs the following steps:
      i. It obtains a subset $S_{i',j'} \in P$ by running the procedure **SubsetTracing** with the partition $P = \{S_{i_1,j_1}, \ldots, S_{i_m,j_m}\}$. If the procedure outputs $\perp$, then it stops the iteration and goto the step 3.
      ii. If $S_{i',j'}$ contains more than two users, then it splits $S_{i',j'}$ into two roughly equal subsets $S_{i'_1,j'_1}$ and $S_{i'_2,j'_2}$ by using the bifurcation property of the SD scheme. Next, it removes $S_{i',j'}$ from P and adds $S_{i'_1,j'_1}, S_{i'_2,j'_2}$ to P.
      iii. If $S_{i',j'}$ contains only one user u, then stops the iteration and goto the step (c).
   (c) It adds u to T.
3. Finally, it outputs the set T.

A.3 Security

Theorem A.3. *The above PKTR scheme is traceable if the above PKTR scheme is indistinguishable under a chosen plaintext attack.*

Proof Sketch. Suppose that the pirate decoder $\mathcal{D}$ can decrypt a ciphertext for the revoked set $R_{\mathcal{D}}$ and a random message M with more than ε probability. To prove the traceability, we should show that the set T outputted by the tracing algorithm is not empty and $T \subseteq E \setminus R_{\mathcal{D}}$.

First, we show that the set T is not empty. By the Lemma A.4, there exists an efficient subset tracing procedure that finds a subset that contains a traitor with a high probability if the underlying encryption scheme and the key assignment method of the scheme are secure. Here $\mathcal{D}$ is stateless, that is, there is a reset button that can turn $\mathcal{D}$ into an original one, and $\mathcal{D}$ only takes a ciphertext that is given by the challenger. Thus if our SRE scheme satisfies the IND-CPA security, then the underlying encryption scheme (ciphertexts of SRE) and the key assignment method (private keys of SRE) are also secure against $\mathcal{D}$. Therefore, the set T is not empty.

Next, we show that $T \subseteq E \setminus R_{\mathcal{D}}$. We can easily show that $T \subseteq \mathcal{N} \setminus R_{\mathcal{D}}$ since the subset tracing produce outputs a subset in the partition P such that $P = \mathcal{N} \setminus R_{\mathcal{D}}$. We also can show that if our PKTR scheme satisfies the IND-CPA security, then $T \subseteq E$ since the adversary can easily break the IND-CPA security of our PKTR scheme by using $\mathcal{D}$ if $T \not\subseteq E$. This completes our proof. $\square$

Lemma A.4 ([22]). *If the underlying encryption scheme is secure and the key assignment method of the underlying encryption scheme is also secure, then there exists an efficient subset tracing procedure with success probability of* $1-\varepsilon\log m$ *that requires at most* $O(m^2\log\frac{1}{\varepsilon}\log^3 m)$ *ciphertext queries to the pirate decoder.*

A.4 Discussions

LSD Scheme. Our PKTR scheme can use the LSD scheme instead of the SD scheme. The tracing algorithm of our PKTR that uses the LSD scheme is the same as that of our PKTR scheme that uses the SD scheme except the (b)-ii step that splits a subset into disjoint subsets. That is, the (b)-ii step in the tracing algorithm additionally splits subsets S_{i_1,j_1}, S_{i_2,i_2} that are already split by the bifurcation property of the SD scheme if the subset does not satisfy the subset condition of the LSD scheme. Therefore, the subset outputted by the subset tracing procedure is split to at most four subsets in the LSD scheme. Note that Halevy and Shamir showed that a subset in the SD scheme can be split at most two subsets to satisfy the conditions of the LSD scheme [18].

New Pirate Attacks. Pirate evolution attack and pirate 2.0 attack that can attack the trace and revoke schemes of Naor et al. [21] were introduced in [2, 19]. The trace and revoke schemes of Naor et al. are very effective to protect the schemes from a clone attacker who just copies all sub-keys of a user to make a pirate decoder, but they are ineffective against a new attacker who just uses a partial set of sub-keys to make a pirate decoder. Our PKTR scheme is also weak against these new attacks.