Server-Aided Revocable Attribute-Based Encryption Revised: Multi-User Setting and Fully Secure

Leixiao Cheng[1] and Fei Meng[2,3](✉)

[1] School of Mathematics, Shandong University, Jinan 250100, China
lxcheng@sdu.edu.cn

[2] Ding Lab, Beijing Institute of Mathematical Sciences and Applications, Beijing, China

[3] Yau Mathematical Center, Tsinghua University, Beijing, China

Abstract. Attribute-based encryption (ABE) is a promising cryptographic primitive achieving fine-grained access control on encrypted data. However, efficient user revocation is always essential to keep the system dynamic and protect data privacy. Cui et al. (ESORICS 2016) proposed the first server-aided revocable attribute-based encryption (SR-ABE) scheme, in which an untrusted server manages all the long-term transform keys and update keys generated by key generation center (KGC) in order to achieve efficient user revocation. So, there's no need for any user to communicate with KGC to update his/her decryption key regularly. In addition, the most part of computational overhead of decryption is outsourced to the server and user keeps a small size of private key to decrypt the final ciphertext. Then, Qin et al.'s (CANS 2017) extended Cui et al.s' work to be decryption key exposure resistant (DKER).

Unfortunately, current SR-ABE schemes could only be provably secure in one-user setting, which means there's only one "target user" id^* with an attribute set S_{id^*} satisfying the access structure $(\mathbb{M}^*, \rho)$ in the challenge ciphertext, i.e., $S_{id^*} \vDash (\mathbb{M}^*, \rho)$. However, a more reasonable security model, i.e., multi-user setting, requires that any user id in the system can be with an attribute set $S_{id} \vDash (\mathbb{M}^*, \rho)$, and the adversary is allowed to query on any user's private key SK_{id} and his/her long-term transform key $PK_{id,S_{id}}$ as long as his/her identity id is revoked at or before the challenge time t^*. How to construct a SR-ABE secure in multi-user setting is still an open problem.

In this paper, we propose the first SR-ABE scheme provably secure in multi-user setting. In addition, our SR-ABE is fully secure and decryption key exposure resistant. Our scheme is constructed based on dual system encryption methodology and novelly combines a variant of Lewko et al.'s work in EUROCRYPT 2010 and Lewko et al.'s work in TCC 2010. As a result, we solve the remaining open problem.

Keywords: Attribute-based encryption · Revocation · Server-aided · Multi-user setting · Fully secure

1 Introduction

Attribute-based Encryption (ABE)[22], as an extension of identity-based encryption (IBE), is a powerful cryptographic primitive achieving fine-grained access control on encrypted data. ABE schemes are usually divided into two types: Key-Policy ABE (KP-ABE) [9] and Ciphertext-Policy ABE (CP-ABE) [3]. In this paper, we only focus on CP-ABE. In CP-ABE scheme, the data owner is allowed to define a specific access policy in the ciphertext which can only be decrypted by users with attributes satisfying the policy. CP-ABE is very suitable for encrypted data sharing in public cloud storage scenarios.

In the IBE or ABE system, when users lose their secret keys or exit the system, efficient user revocation is very crucial for preserving data privacy and keeping the system dynamic. In 2001, Boneh et al. [6] proposed a simple identity revocation mechanism, in which the Key Generation Center (KGC) has to generate $O(N-r)$ new secret keys for all unrevoked users at time period t, where N is the total number of users and r is the number of revoked users. To reduce the workload of KGC, Boldyreva et al. [4] proposed a more efficient identity revocation mechanism based on the binary-tree structure of [15]. In [4], each user keeps $O(\log N)$ long-term secret keys and the KGC broadcasts $O(r\log(N/r))$ update keys at time period t. Only non-revoked users can obtain their corresponding update keys. However, there are two drawbacks in [4]: every user needs to keep at least $O(\log N)$ long-term secret keys; all non-revoked users are required to communicate with the KGC regularly. As a result, [4] is not suitable for users with limited resources or who cannot communicate with KGC in real-time.

To solve this problem, Qin et al. [17] proposed a novel system model i.e., server-aided revocation in IBE scenario (SR-IBE). In [17], user's original long-term secret keys and update keys are all managed by an untrusted server, which honestly follows the protocol but is curious about data encrypted in the ciphertext, and each user keeps only one short private key. Since the original long-term secret keys are stored in the server, those keys are renamed as long-term transform keys. In this case, user no longer needs to communication with KGC for key updating regularly. To extend server-aided revocation mechanism from IBE to ABE scenario, in 2016, Cui et al. [8] proposed the first server-aided revocable ABE (SR-ABE). [8] not only inherits the advantages of server-aided revocation mechanism, but also achieves the decryption outsourcing, i.e., user could decrypt the ciphertext with little computational overhead. However, the scheme fails to satisfy (local) decryption key exposure resistance (DKER). Specifically, in [8], user's decryption key is his/her private key, which does not change with time, so exposing the user's decryption key will make the scheme completely insecure. Seo and Emura [23] has shown that the exposure of decryption keys is a very realistic threat to many revocable cryptosystems. Then, Qin et al. [18,19] revisited the security model of [8] and enhanced it by capturing the decryption key exposure attacks on user's local decryption keys while allowing the adversary to fully corrupt the server. In [18,19], the user keeps just a short private key, and can delegate his/her decryption capacity to a decryption key with any specified time period. Even if the local decryption key of a certain time period is leaked, the security of the decryption key of other time periods will not be affected. Similarly, [18,19] maintain the properties of server-aided revocation and outsourced decryption.

In general, the system framework of SR-ABE is shown in Fig. 1. The ABE ciphertext generated by data owner is transformed by an untrusted server using a short-term transformation key which is generated by combining the long-term transformation key and the key update message. However, once a user is revoked, the server cannot assist him/her to accomplish the transformation. In [8], user's private key is the decryption key, so once the decryption key is exposed, user's privacy is totally leaked. However, in [18,19], user's decryption key is derived from the private key, so even a decryption key of time period t is exposed, it will not affect decryption keys in other time periods.

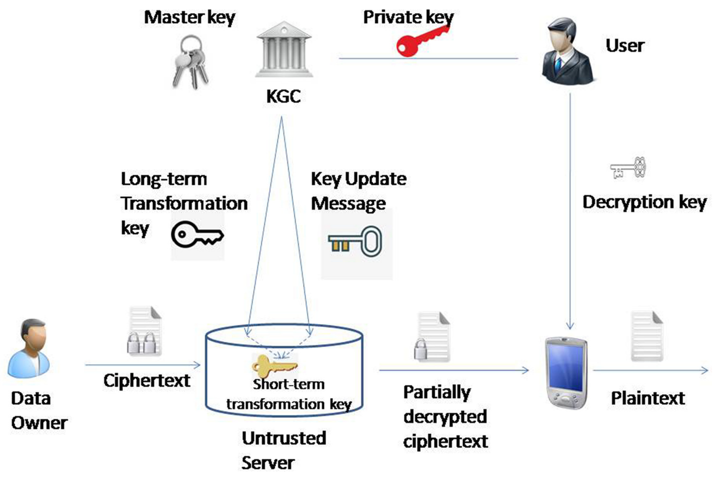

Fig. 1. System framework of SR-ABE

1.1 Motivation

The existing SR-ABE schemes [8,18,19] can only prove secure under "one-user setting", in which only one user id^* (called "target user") has the capacity to access the challenge ciphertext and the adversary can corrupt his private key. In [8,18,19], the adversary is divided into two distinct types: (1) the adversary is allowed to corrupt id^*'s private key, but id^* has to be revoked at or before the challenge time period t^*; (2) the adversary is not allowed to corrupt id^*'s private key, then id^* is not revoked but the adversary can obtain decryption keys for any time period except t^*. Note that [8,18,19] can only simulate the security game for these two types of adversaries separately, which do not cross with each other. However, this is too restrictive in a real world scenario, since the two different adversaries may exist simultaneously even for two target users.

But unfortunately, the restriction seems necessary for the proof technique used in [8,18,19].

Let us analyze why this is the case. In this paper, we focus on SR-ABE scheme with DKER. Note that [8] is improved by [18,19] which achieves DKER, so we take [18,19] as example. The analysis works similarly for [8] as well. First, we briefly recall some algorithms in the construction of [18,19]:

Setup(1^λ)**:** This algorithm outputs master secret key $msk = \alpha$, the public parameter $par = (g, w, v, u, h, u_0, h_0, e(g,g)^\alpha)$, along with a revocation list RL and a state st, where g is the generator of a group G, w, v, u, h, u_0, h_0 are randomly chosen from G, α is randomly chosen from $\mathbb{Z}_p$, st is set to be the binary tree BT (BT is introduced in Sect. 2.3).

UserKG(par, msk, id, S, st)**:** This algorithm randomly chooses $\beta_{id} \in \mathbb{Z}_p$ and set $sk_{id} = g^{\beta_{id}}$. Then, it chooses an undefined leaf node θ_{id} from BT, stores id in this node. For each $x \in \mathsf{Path}(\mathsf{BT}, \theta_{id})$, it runs as follows.

1. It fetches g_x from the node x. If x has not been defined, it randomly chooses $g_x \in G$, computes $g'_x = g^{\alpha - \beta_{id}}/g_x$ and stores g_x in the node x.
2. It randomly chooses $r_x, r_{x,1}, \dots, r_{x,k} \in \mathbb{Z}_p$, computes

$$P_{x,0} = g'_x \cdot w^{r_x}, \quad P_{x,1} = g^{r_x}, \quad P^{(i)}_{x,2} = g^{r_{x,i}}, \quad P^{(i)}_{x,3} = (u^{s_i} h)^{r_{x,i}} \cdot v^{-r_x}.$$

3. it outputs $pk_{id,S} = \{x, P_{x,0}, P_{x,1}, P^{(i)}_{x,2}, P^{(i)}_{x,3}\}_{x \in \mathsf{Path}(\mathsf{BT}, \theta_{id}), i \in [1,k]}$ as the long-term transformation key and $sk_{id} = g^{\beta_{id}}$ as the secret key.

TKeyUp$(par, msk, \mathsf{RL}, t, st)$**:** This algorithm inputs par, msk, a revocation list RL, a time period t and a state st. For each $x \in \mathsf{KUNodes}(\mathsf{BT}, \mathsf{RL}, t)$, it randomly chooses $s_x \in \mathbb{Z}_p$, fetches g_x from the node x, outputs a key update message $tku_t = \{x, Q_{x,1}, Q_{x,2}\}_{x \in \mathsf{KUNodes}(\mathsf{BT}, \mathsf{RL}, t)}$, where

$$Q_{x,0} = g_x \cdot (u^t h)^{s_x}, \quad Q_{x,1} = g^{s_x}.$$

Encrypt$(par, (\mathbb{M}, \rho), t, M)$**:** This algorithm inputs par, an LSSS access structure $(\mathbb{M}, \rho)$, a time period t and a message M, randomly chooses $\mathbf{v} = (s, y_2, \dots, y_n)^\perp \in \mathbb{Z}_p^n$ and $\mu_1, \dots, \mu_l \in \mathbb{Z}_p$, computes $\lambda_i = \mathbb{M}_i \cdot \mathbf{v}$, outputs the ciphertext $CT = \{C, C_0, \{C_{i,1}, C_{i,2}, C_{i,3}\}, C_4\}$, where

$$C = e(g,g)^{\alpha s} \cdot M, \quad C_0 = g^s, \quad C_{i,1} = w^{v_i} \cdot v^{\mu_i},$$
$$C_{i,2} = (u^{s_{\rho(i)}} h)^{-\mu_i}, \quad C_{i,3} = g^{\mu_i}, \quad C_4 = (u^t h)^s.$$

As we can see, Qin et al. [18,19] used the "random splitting technique" to divide a master secret key "α" into two parts. Specifically, "α" is split into "$\alpha - \beta_{id}$" and "β_{id}" for identity id, where β_{id} is randomly chosen and $g^{\beta_{id}}$ serves as the user's private key in **UserKG**. In order to achieve user revocation, $g^{\alpha - \beta_{id}}$ will be further divided into random g_x and $g'_{x,id}$ to generate key update message in **TKeyUp** and long-term transformation key in **UserKG** respectively (g_x is stored in the node x of BT, and does not change once stored; $g'_{x,id}$ changes with different identities), such that

$$g_x \cdot g'_{x,id} = g^{\alpha - \beta_{id}}, \quad x \in \mathsf{Path}(\mathsf{BT}, \theta_{id}). \tag{1}$$

The security of the SR-ABE scheme [18,19] is reduced to the Rouselakis-Waters CP-ABE scheme [20]. It seems that [18,19] cannot prove secure in multi-user setting. This is because if the adversary $\mathcal{A}$, who attacks the SR-ABE scheme [18,19], is allowed to simultaneously corrupt two separate users in two different types, as we mentioned at the beginning of this section, then the simulator $\mathcal{B}$, who was built using $\mathcal{A}$ to attack [20], can break [20] itself, which leads to the failure of the security reduction. The detail is as follows.

The simulator $\mathcal{B}$ is given the public parameters of Rouselakis-Waters, and the master key α is hidden from $\mathcal{B}$. Assume that there are two identities id_0^* and id_1^* with attribute sets S_0 and S_1 satisfying the challenge access structure $(\mathbb{M}^*, \rho)$ such that:

(1) $\mathcal{A}$ corrupts id_0^*'s private key and id_0^* is revoked before the time period t^*;
(2) $\mathcal{A}$ doesn't corrupts id_1^*'s private key and id_1^* is not revoked.

According to Eq. (1), we have

$$g_x \cdot g'_{x,id_0} = g^{\alpha - \beta_{id_0}}, \quad x \in \mathsf{Path}(\mathsf{BT}, \theta_{id_0}) \tag{2}$$

$$g_x \cdot g'_{x,id_1} = g^{\alpha - \beta_{id_1}}, \quad x \in \mathsf{Path}(\mathsf{BT}, \theta_{id_1}) \tag{3}$$

Since $S_0 \models (\mathbb{M}^*, \rho)$, g'_{x,id_0} for $x \in \mathsf{Path}(\mathsf{BT}, \theta_{id_0})$, are known to $\mathcal{B}$ in order to generate the long-term transformation key pk_{id_0,S_0} for $\mathcal{A}$; Since $\mathcal{A}$ corrupts id_0^*'s private key, β_{id_0} is a known value to $\mathcal{B}$; Since id_1 is non-revoked, $\mathcal{B}$ has to know g_x for $x \in \mathsf{Path}(\mathsf{BT}, \theta_{id_1})$ to generate the key update message at time period t^* for $\mathcal{A}$, especially the value g_{x^*} for $x^* \in \mathsf{Path}(\mathsf{BT}, \theta_{id_0}) \cap \mathsf{Path}(\mathsf{BT}, \theta_{id_1})$. According to Equation (2), $\mathcal{B}$ knows the value

$$g^{\alpha} = (g_{x^*} \cdot g'_{x^*,id_0}) \cdot g^{\beta_{id_0}},$$

which enables $\mathcal{B}$ to break the underlying Rouselakis-Waters CP-ABE scheme [20], and thus the security reduction fails. For the similar reason, [8] also cannot prove secure in multi-user setting and it is still a practical open problem to construct an SR-ABE scheme probably secure under such setting.

1.2 Our Approach

As we analyzed above, in previous SR-ABE schemes [8,18,19], if the adversary $\mathcal{A}$ is allowed to simultaneously corrupt two separate users in two different types, then the simulator $\mathcal{B}$ is able to compute g^{α} (α is the master secret key) and solve the underlying complexity assumption itself, thus the security reduction breaks down. This hints that we need to find a new construction and proof system so that the exposure of g^{α} will not lead to the failure of the security reduction. Fortunately, Waters et al. [11,13,24] introduced dual system methodology, which opens up a new way to prove security of IBE, ABE and other related encryption systems.

Briefly speaking, in the dual encryption system [11,13], both ciphertext and private key can be in one of two indistinguishable forms: normal and semi-functional. Unless both the key and ciphertext are semi-functional, the key will decrypt the ciphertext correctly. However, when a semi-functional key is used to decrypt a semi-functional ciphertext, the semi-functional components of the key and ciphertext will interact to generate an additional random term, and decryption will fail. In the real system, the normal keys and ciphertexts are used, while semi-functional objects are gradually presented in hybrid security proof: firstly in Game_0, the normal challenge ciphertext is switched to a semi-functional one; then, from Game_1 to Game_q, the secret keys given the adversary are changed from normal to semi-functional one by one and Game_q is a security game where the simulator only generates semi-functional objects; finally, we end up in $\mathsf{Game}_{\mathsf{Final}}$ where the challenge ciphertext is a semi-function encryption on a *random* group element and *all* of the private key queries result in semi-functional key, hence security can be proved straightforward.

When arguing that Game_{k-1} and Game_k are indistinguishable for $k \in [1, q]$, the simulator $\mathcal{B}$ who attacks the underlying assumptions (Assumption 1 and 2 in Sect. 2.1) always chooses the master secret key α by himself so that he is ready to make *any* key and *any* challenge ciphertext for adversary $\mathcal{A}$ who attacks the scheme. When claiming that Game_q and $\mathsf{Game}_{\mathsf{Final}}$ are indistinguishable, the simulator $\mathcal{B}$ who attacks Assumption 3 (Sect. 2.1) takes as input parameters $g, g^\alpha X_2, X_3, g^s Y_2, Z_2, T$ (T is either $e(g,g)^\alpha$ or a random element in G_T), then it makes use of the Assumption 3 parameter $g^\alpha X_2$ to generate semi-functional objections to answer *any* key query from $\mathcal{A}$.

Based on this observation, the dilemma encountered in proving multi-user security in the previous SR-ABE schemes could be overcome, because we no longer need to worry about the exposure of g^α or $g^\alpha X_2$. In other words, leveraging dual system methodology into SR-ABE may lead us down the right path to prove the security under multi-user setting. Therefore, we novelly combine the ABE scheme [11] and the IBE scheme [13] in the dual system to construct our SR-ABE scheme. Thanks to the powerful dual system encryption methodology, in our security proof, even the adversary $\mathcal{A}$ corrupts two separate users in two different types simultaneously, the simulator $\mathcal{B}$, who knows α or $g^\alpha X_2$, is able to answer *any* key query on these two users from $\mathcal{A}$, and thus the security reduction works. As a result, our SR-ABE is provably secure in multi-user setting. The only remaining question is how to combine those two schemes in the dual system organically to obtain a concrete SR-ABE scheme. We put this detail at the beginning of Sect. 4.

1.3 Our Contributions

In this paper, we construct the first (fully secure) server-aided revocable attribute-based encryption scheme with decryption key exposure resistance, achieving the security requirement in the multi-user setting. We solve the open problem of how to construct a SR-ABE scheme that is provably secure in the

Table 1. Comparison between our scheme and other indirect revocable ABE schemes.

	[4]	[1]	[21]	[8]	[18]	Ours
Revocation Mode	Indirect	Indirect	Indirect & Direct	Indirect	Indirect	Indirect
Type of ABE	KP-ABE	KP-ABE	KP-ABE & CP-ABE	CP-ABE	CP-ABE	CP-ABE
Server	–	–	–	Untrusted	Untrusted	Untrusted
Decryption Outsource	No	No	No	Yes	Yes	Yes
DKER	No	No	No	No	Yes	Yes
Fully Secure	No	No	No	No	No	Yes
Secure Channel	Yes	Yes	Yes	No	Yes/No	Yes/No
Multi-User Setting	Yes	Yes	Yes	No	No	Yes
Size of Key Updates	$O(r \log \frac{N}{r})$	$O(r \log \frac{N}{r})$ & –	$O(r \log \frac{N}{r})$	$O(r \log \frac{N}{r})$	$O(r \log \frac{N}{r})$	$O(r \log \frac{N}{r})$
Size of Key Stored by User	$O(l \log N)$	$O(l \log N)$	$O(l \log N)$ & $O(k \log N)$	$O(1)$	$O(1)$	$O(1)$

multi-user setting. Specifically, our scheme novelly combines a variant of a fully secure (H)IBE [13] and a fully secure ABE [11] in the dual encryption system.

In Table 1, we compare our SR-ABE scheme with several related indirect revocable ABE schemes [1,4,8,18,21]. Let N be the number of user in the system, r be the number of revoked users, l be the number of attributes presented in an access structure, and k be the size of the attribute set associated with an attribute-key. Also, let "-" denote not-applicable. As shown in Table 1, compared with [1,4,21], our scheme has inherited the wonderful merits of SR-ABE schemes [8] and [18], i.e., decryption outsourced and small size of key storage in the user side. There's no need for any user to communicate with KGC to update his/her decryption key regularly as well. In addition, compared with [8] and [18], our scheme is fully secure and provably secure in multi-user setting. Furthermore, different from [8], our SR-ABE satisfies DKER.

2 Preliminaries

In this section, we briefly introduce some basic cryptographic definitions.

2.1 Composite Order Bilinear Groups

We recall the definition of composite order bilinear groups in [13]. A group generator $\mathcal{G}$ is defined as an algorithm that takes a security parameter λ as input and outputs $(p_1, p_2, p_3, G, G_T, e)$, where p_1, p_2, p_3 are distinct primes, G and G_T are two cyclic groups of order $N = p_1 p_2 p_3$, and $e : G \times G \rightarrow G_T$ is a bilinear map with the following properties:

Bilinear: $\forall u, v \in G$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
Non-degenerate: $\exists g \in G$ such that $e(g, g) \in G_T$ is the generator of G_T.

The group operations in G and G_T as well as the bilinear map e are computable in polynomial time. Let $G_{p_1}, G_{p_2}, G_{p_3}$ denote the subgroups of order p_1, p_2, p_3 in G respectively, then when $h_i \in G_{p_i}$ and $h_j \in G_{p_j}$ for $i \neq j$, $e(h_i, h_j)$ is the identity element in G_T. This orthogonality property of $G_{p_1}, G_{p_2}, G_{p_3}$ will be used to implement semi-functionality in our SR-ABE.

We now introduce the complexity assumptions [11,12]. Let $G_{p_1 p_2}$ and $G_{p_1 p_3}$ denote the subgroup of order $p_1 p_2$ and $p_1 p_3$ in G, respectively.

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

$$\mathbb{G} = (N = p_1p_2p_3, G, G_T, e) \xleftarrow{R} \mathcal{G}, g \xleftarrow{R} G_{p_1}, X_3 \xleftarrow{R} G_{p_3}, D = (\mathbb{G}, g, X_3)$$
$$T_1 \xleftarrow{R} G_{p_1p_2}, T_2 \xleftarrow{R} G_{p_1}.$$

The advantage of an algorithm $\mathcal{A}$ in breaking Assumption 1 is defined as:

$$Adv1_{\mathcal{G},\mathcal{A}}(\lambda) :=\mid \Pr[\mathcal{A}(D, T_1)] - \Pr[\mathcal{A}(D, T_2)] \mid .$$

Definition 1. *We say that $\mathcal{G}$ satisfies Assumption 1 if $Adv1_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

Note that T_1 can be written uniquely as the product of an element of G_{p_1} and an element of G_{p_2}. We refer to these elements as the G_{p_i} part of T_1 for $i = 1, 2$.

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

$$\mathbb{G} = (N = p_1p_2p_3, G, G_T, e) \xleftarrow{R} \mathcal{G}, g, X_1 \xleftarrow{R} G_{p_1}, X_2, Y_2 \xleftarrow{R} G_{p_2},$$
$$X_3, Y_3 \xleftarrow{R} G_{p_3}, D = (\mathbb{G}, g, X_1X_2, X_3, Y_2Y_3), T_1 \xleftarrow{R} G, T_2 \xleftarrow{R} G_{p_1P_3}.$$

The advantage of an algorithm $\mathcal{A}$ in breaking Assumption 2 is defined as:

$$Adv2_{\mathcal{G},\mathcal{A}}(\lambda) :=\mid \Pr[\mathcal{A}(D, T_1)] - \Pr[\mathcal{A}(D, T_2)] \mid .$$

Definition 2. *We say that $\mathcal{G}$ satisfies Assumption 2 if $Adv2_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

Note that T_1 can be written uniquely as the product of an element of G_{p_1}, an element of G_{p_2} and an element of G_{p_3}. We refer to these elements as the G_{p_i} part of T_1 for $i = 1, 2, 3$. T_2 can be written as the product of an element of G_{p_1} and an element of G_{p_3} similarly.

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

$$\mathbb{G} = (N = p_1p_2p_3, G, G_T, e) \xleftarrow{R} \mathcal{G}, \alpha, s \xleftarrow{R} \mathbb{Z}_N, g \xleftarrow{R} G_{p_1}, X_2, Y_2, Z_2 \xleftarrow{R} G_{p_2},$$
$$X_3 \xleftarrow{R} G_{p_3}, D = (\mathbb{G}, g, g^{\alpha}X_2, X_3, g^sY_2, Z_2), T_1 = e(g, g)^{\alpha s}, T_2 \xleftarrow{R} G_T.$$

The advantage of an algorithm $\mathcal{A}$ in breaking Assumption 3 is defined as:

$$Adv3_{\mathcal{G},\mathcal{A}}(\lambda) :=\mid \Pr[\mathcal{A}(D, T_1)] - \Pr[\mathcal{A}(D, T_2)] \mid .$$

Definition 3. *We say that $\mathcal{G}$ satisfies Assumption 3 if $Adv3_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

2.2 Access Structures and Linear Secret Sharing

Definition 4 (Access structure[3]). *Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,P_2,\ldots,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \ldots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,P_2,\ldots,P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.*

Definition 5 (Linear Secret Sharing Schemes (LSSS)[3]). *A secret sharing scheme Π over a set of parties P is a linear secret-sharing scheme over Z_p if:*

- *The shares for each party form a vector over Z_p.*
- *There exists a matrix $\mathbb{M}$ with l rows and n columns, called the share generating matrix, for Π. For $i = 1, \ldots, l$, the i^{th} row of matrix $\mathbb{M}$, i.e., $\mathbb{M}_i$, is labelled by a party $\rho(i)$, where $\rho : \{1, \ldots, l\} \to P$ is a function that maps a row to a party for labelling. Considering that the column vector $\vec{v} = (s, r_2, \ldots, r_n)$, where $s \in Z_p$ is the secret to be shared and $r_2, \ldots, r_n \in Z_p$ are randomly chosen, then $\mathbb{M}\vec{v}$ is the vector of l shares of the secret s according to Π. The share $\mathbb{M}_i\vec{v}$ belongs to party $\rho(i)$.*

The linear reconstruction property states that there exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i\in I}$ such that, for any valid shares $\{\lambda_i\}_i$ of a secret s according to Π, we have: $\Sigma_{i\in I}\omega_i\lambda_i = s$, where $I = \{i \mid \rho(i) \in S\}$ for an authorized set S [2]. We note that for unauthorized sets, no such constants $\{\omega_i\}$ exist.

2.3 Binary Tree

We recall the definition of binary-tree data structure, as with [5,7,10,16,23]. This structure uses a node selection algorithm called KUNodes. In the algorithm, we use the following notations: BT denotes a binary-tree. root denotes the root node of BT. x denotes a node in the binary tree and θ emphasizes that the node x is a leaf node. The set $\mathsf{Path}(\mathsf{BT}, \theta)$ stands for the collection of nodes on the path from the leaf θ to the root (including θ and the root). If x is a non-leaf node, then x_ℓ, x_r denote the left and right child of x, respectively. The KUNodes algorithm takes as input a binary tree BT, a revocation list RL and a time t, and outputs the minimal set Y of nodes, such that the corresponding key update information can only be used by the non-revoked users to generate a valid short-term transformation key. Specifically, the KUNodes algorithm first marks all ancestor of users that were revoked by t as revoked nodes, then outputs all the non-revoked children of revoked nodes. The description of the KUNodes algorithm is as follows:

```
KUNodes(BT, RL, t):
  $X, Y \leftarrow \emptyset$;   $\forall(\theta_i, t_i) \in \mathsf{RL}$, and $t_i \le t$, add $\mathsf{Path}(\mathsf{BT}, \theta_i)$ to $X$;
  $\forall x \in X$: if $x_\ell \notin X$ then add $x_\ell$ to $Y$, if $x_r \notin X$ then add $x_r$ to $Y$;
  If $Y = \emptyset$ then add root to $Y$;   Return $Y$.
```

3 Framework and Security Model

Our SR-ABE scheme involves four types of entities: a key generation center (KGC), data owners, data users and an untrusted server.

Setup$(1^\lambda, U) \rightarrow (PK, MSK, \mathsf{RL}, st)$**:** Taking as input a security parameter λ and an attribute set U containing all possible attributes, KGC runs this algorithm to generate the public key PK, the master secret key MSK, an initially empty revocation list RL and a state st.

UserKG$(PK, MSK, id, S, st) \rightarrow (PK_{id,S}, SK_{id}, st)$**:** KGC runs the user key generation algorithm and outputs user's long-term transformation key $PK_{id,S}$ for the untrusted server and a private key SK_{id} for the user, then updates the state st.

TKeyUp$(PK, MSK, \mathsf{RL}, t, st) \rightarrow (tku_t, st)$**:** KGC runs the transformation key update algorithm and outputs a key update message tku_t for server and an updated state st.

TranKG$(PK, id, PK_{id,S}, tku_t) \rightarrow tk_{id,t}/\perp$**:** The server runs the transformation key generation algorithm and outputs a short-term transformation key $tk_{id,t}$ for id if id is not revoked at t. Otherwise, it outputs $\perp$.

DecKG$(PK, id, SK_{id}, t) \rightarrow dk_{id,t}$**:** The user runs the decryption key generation algorithm and outputs a decryption key $dk_{id,t}$ for id in time period t.

Enc$(PK, (\mathbb{M}, \rho), t, M) \rightarrow CT$**:** Taking the public key PK, an access structure $(\mathbb{M}, \rho)$, a time period t and a message M as the input, the data owner runs the encryption algorithm to generate a ciphertext CT and then submits CT to server.

Transform$(PK, id, S, tk_{id,t}, CT) \rightarrow CT'/\perp$**:** The server runs the ciphertext transformation algorithm to generate a partially decrypted ciphertext CT' for id if the attribute set S associated with the transformation key $tk_{id,t}$ satisfies the access structure of the ciphertext CT. Otherwise, it outputs $\perp$.

Decrypt$(PK, id, dk_{id,t}, CT') \rightarrow M/\perp$**:** The user runs the decryption algorithm and outputs the message M or a failure symbol $\perp$.

Revoke$(id, t, \mathsf{RL}, st) \rightarrow \mathsf{RL}$**:** KGC runs the revocation algorithm and outputs an updated revocation list RL.

3.1 Security Model

Now, we introduce the security definition of indistinguishability under chosen plaintext attacks (IND-CPA security) for SR-ABE between an adversary $\mathcal{A}$ and the challenger $\mathcal{B}$.

Setup: $\mathcal{B}$ runs the **Setup** algorithm, and returns the public key to $\mathcal{A}$, then keeps the master secret key MSK, an initially empty revocation list RL, a state st, and two empty sets T, T' by itself.

Phase 1: $\mathcal{A}$ adaptively issues a sequence of following queries to $\mathcal{B}$:

- Create(id,S). $\mathcal{B}$ runs **UserKG**(PK, MSK, id, S, st) to obtain the pair $(PK_{id,S}, SK_{id})$, stores in table T the entry $(id, S, PK_{id,S}, SK_{id})$ and returns $PK_{id,S}$ to $\mathcal{A}$.

- Corrupt(id). If there exists an entry indexed by id in table T, then $\mathcal{B}$ retrieves the entry $(id, S, PK_{id,S}, SK_{id})$, sets $T' = T' \cup \{(id, S)\}$, returns SK_{id}. If no such entry exists, then it returns $\perp$.
- TKeyUp(t). $\mathcal{B}$ runs **TKeyUp**$(PK, MSK, \mathsf{RL}, t, st)$ and returns tku_t.
- DecKG(id,t). If there exists an entry indexed by id in table T, then $\mathcal{B}$ retrieves the entry $(id, S, PK_{id,S}, SK_{id})$, runs DecKG$(PK, id, SK_{id}, t)$ and returns $dk_{id,t}$. If no such entry exists, then it returns $\perp$.
- Revocation(id,t). $\mathcal{B}$ runs **Revoke**(id, t, RL, st) and outputs an updated revocation list RL.

Challenge: $\mathcal{A}$ submits two messages (M_0, M_1) of the same size, an access structure $(\mathbb{M}^*, \rho)$ and a time period t^* with the following restrictions:
- If $(id, S) \in T'$ and $S \vDash (\mathbb{M}^*, \rho)$, then $\mathcal{A}$ must query the revocation oracle on (id, t) at or before t^*.
- If there exists a tuple $(id, S, PK_{id,S}, SK_{id}) \in T$, $S \vDash (\mathbb{M}^*, \rho)$ and id is not revoked at or before t^*, then $\mathcal{A}$ cannot query Corrupt(id) and DecKG(id,t^*).

$\mathcal{B}$ picks a random bit $\beta \in \{0, 1\}$, and returns the challenge ciphertext $CT^* \leftarrow$ **Enc**$(PK, (\mathbb{M}^*, \rho), t^*, M_\beta)$ to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ continues submits queries to $\mathcal{B}$ as in Phase 1, with the restrictions defined in the Challenge phase.

Guess:$\mathcal{A}$ outputs a guess β' of β, and it wins the game if $\beta' = \beta$. The advantage of $\mathcal{A}$ in this game is defined as $\mathsf{Adv}_{\mathcal{A}}(l) =\mid \Pr[\beta' = \beta] - 1/2 \mid$.

Definition 6. *An SR-ABE scheme is adaptively IND-CPA secure if the advantage* $\mathsf{Adv}_{\mathcal{A}}(l)$ *is negligible in* l *for all polynomial time adversary* $\mathcal{A}$.

4 Construction

In this section, we propose the construction of our SR-ABE with DKER, which is fully secure in multi-user setting. As we have discussed in Sect. 1.2, we try to construct an SR-ABE by the dual system encryption technique. However, we find that if we trivially follow the construction of SR-ABE with DKER [18,19] by just replacing their underlying ABE block [20] with dual ABE [11], then it will cause "authority abuse": (1) anyone can generate a valid private key, since it is computed without the system master secret key; (2) user id with his private key SK_{id} can easily change the identity embedded in his long-term transform key $PK_{id,S}$ from id to id'. Adding up these two points, user id can maliciously generate a new $SK_{id'}$ and $PK_{id',S'}$, where $S' \subseteq S$, for an unauthorized user id' by the key re-randomization technique, resulting in the authority abuse.

In our scheme, we novelly combine dual ABE [11] and dual IBE [13]. Firstly, we embed the system master key into user's private key to ensure that only the KGC can distribute users' private keys. Specifically, we view the system master key α as the master key of a variant of the 2-level HIBE [13] to generate 1-level user private key SK_{id} (id as identity), which is then used to delegate a 2-level decryption key $dk_{id,t}$ ($(id \parallel t)$ as identity). It should be noted that the exposure of decryption key $dk_{id,t}$ on time t will not affect the decryption key $dk_{id,t'}$ on time $t' \neq t$, so that our SR-ABE is decryption key exposure resistant (DKER). Secondly, we generate user's long-term transformation key $PK_{id,S}$ by combining the key generation algorithms of both [11] and the variant of [13]. The unique random r embedded in both SK_{id} and $PK_{id,S}$ guarantees that, without knowing r, anyone cannot change the identity embedded in SK_{id} and $PK_{id,S}$, so that the authority abuse is prevented. The detail of our scheme is shown as follows.

– **Setup**$(1^\lambda, U) \rightarrow (PK, MSK)$**:** The setup algorithm chooses a bilinear group G of order $N = p_1p_2p_3$, where p_1, p_2, p_3 are three distinct primes. We let G_{p_i} denote the subgroup of order p_i in G. It then chooses random exponents $\alpha, a \in \mathbb{Z}_N$, and random group elements $g, u, h, u_0, h_0 \in G_{p_1}$. For each attribute $i \in U$, it chooses a random value $s_i \in \mathbb{Z}_N$. Then, the algorithm outputs the public parameters PK and master secret key MSK as follows:

$$PK = \{N, g, g^a, u, h, u_0, h_0, e(g,g)^\alpha, \{T_i = g^{s_i}\}_{i \in U}\}, MSK = \{\alpha, X_3\} \quad (4)$$

where X_3 is a generator of G_{p_3}.

– **UserKG**$(PK, MSK, id, S, st) \rightarrow (PK_{id,S}, SK_{id}, st)$**:** The algorithm chooses an undefined leaf node θ from the binary tree BT, and stores id in this node. Then, it randomly chooses $r \in \mathbb{Z}_N$. For each node $x \in \mathsf{Path}(\mathsf{BT}, \theta)$, it runs as follows.
  - It fetches g_x from the node x. If x has not been defined, it randomly picks $g_x \in G_{p_1}$, then stores g_x in node x.
  - It randomly chooses $t_x \in \mathbb{Z}_N, R_{x,0}, \bar{R}_{x,0}, R_{x,i} \in G_{p_1}$, and computes

$$PK_{id,S,x} = \left\{ \begin{array}{l} K_x = g^{\alpha + a t_x r}((u^{id}h)^r / g_x) \cdot R_{x,0},\ L_x = g^{t_x r} \cdot \bar{R}_{x,0} \\ \{K_{x,i} = T_i^{t_x r} \cdot R_{x,i}\}_{i \in S} \end{array} \right\}. \quad (5)$$

Then, the algorithm picks a random element $R_3 \in G_{p_3}$ and computes

$$SK_{id} = g^\alpha (u^{id}h)^r \cdot R_3. \quad (6)$$

Finally, the algorithm outputs the long-term transformation key $PK_{id,S} = \{x, PK_{id,S,x}\}_{x \in \mathsf{Path}(\mathsf{BT},\theta)}$, the private key SK_{id} and updates the state st.

– **TKeyUp**$(PK, MSK, \mathsf{RL}, t, st) \rightarrow (tku_t, st)$**:** For each $x \in \mathsf{KUNodes}(\mathsf{BT}, \mathsf{RL}, t)$, the algorithm fetches g_x from x (g_x should always be predefined in the above UserKG algorithm), randomly chooses $\hat{R}_{x,3}, \bar{R}_{x,3} \in G_{p_3}$, $s_x \in \mathbb{Z}_N$, and computes

$$Q_{x,0,t} = g^\alpha g_x \cdot (u_0^t h_0)^{s_x} \hat{R}_{x,3}, \qquad Q_{x,1,t} = g^{s_x} \bar{R}_{x,3}. \quad (7)$$

Finally, the algorithm generates the transformation key update information $tku_t = \{x, Q_{x,0,t}, Q_{x,1,t}\}_{x \in \mathbf{KUNodes}(\mathsf{BT},\mathsf{RL},t)}$ and updates the state st.

– **TranKG**$(PK, id, PK_{id,S}, tku_t) \rightarrow tk_{id,t} / \perp$**:** Suppose θ is the leaf node corresponding with id. If $\mathsf{Path}(\mathsf{BT}, \theta) \bigcap \mathsf{KUNodes}(\mathsf{BT}, \mathsf{RL}, t) = \emptyset$, the algorithm returns $\perp$. Otherwise, there must exist one node x such that $x \in$

$\mathsf{Path}(\mathsf{BT},\theta)\bigcap\mathsf{KUNodes}(\mathsf{BT},\mathsf{RL},t)$. Then, it computes

$$\left\{\begin{array}{l} tk_0 = K_x \cdot Q_{x,0,t} = g^{2\alpha+at_xr}(u^{id}h)^r(u_0^th_0)^{s_x}\cdot R_{x,0}\hat{R}_{x,3} \\ tk_1 = L_x = g^{t_xr}\cdot \bar{R}_{x,0} \\ tk_{2,i} = K_{x,i} = T_i^{t_xr}R_{x,i} \qquad \forall i \in S \\ tk_3 = Q_{x,1,t} = g^{s_x}\bar{R}_{x,3} \end{array}\right\}. \tag{8}$$

Finally, it returns the transformation key $tk_{id,t} = \{tk_0, tk_1, \{tk_{2,i}\}_{i\in S}, tk_3\}$.

– **DecKG**$(PK, id, SK_{id}, t) \rightarrow dk_{id,t}$: It randomly chooses $r_t \in \mathbb{Z}_N$ and outputs a decryption key $dk_{id,t} = \{SK_{id}(u_0^th_0)^{r_t}, g^{r_t}\} = \{g^{\alpha}(u^{id}h)^r(u_0^th_0)^{r_t}R_3, g^{r_t}\}$.
– **Enc**$(PK,(\mathbb{M},\rho),t,M) \rightarrow CT$**:** Given an LSSS access structure $(\mathbb{M},\rho)$, $\mathbb{M}$ is an $l\times n$ matrix and ρ is a map from each row $\mathbb{M}_i$ of $\mathbb{M}$ to an attribute $\rho(i)$. The algorithm randomly chooses a random vector $\vec{v}\in\mathbb{Z}_N^n$ and denotes $\vec{v} = (s, y_2,\ldots,y_n)^{\perp}$. Then it computes the shares $\lambda = (\lambda_1,\ldots,\lambda_l)^{\perp} = \mathbb{M}\cdot\vec{v}$, where $\lambda_i = \mathbb{M}_i\cdot\vec{v}$. Finally, it randomly chooses $r_1,\ldots,r_l\in\mathbb{Z}_N$ and outputs the ciphertext CT as

$$CT = \left\{\begin{array}{l} C = M\cdot e(g,g)^{\alpha s}, C_0 = g^s, C_t = (u_0^th_0)^s \\ \{C_i = g^{a\lambda_i}T_{\rho(i)}^{-r_i}, D_i = g^{r_i}\}_{i\in[1,l]} \end{array}\right\}. \tag{9}$$

– **Transform**$(PK, id, S, tk_{id,t}, CT) \rightarrow CT'/\perp$**:** If the user has been revoked at time period t or the attribute set $S \nvDash (\mathbb{M},\rho)$, the algorithm returns $\perp$. Otherwise, the algorithm computes a set of constants $\{\omega_i\in\mathbb{Z}_N\}$ such that $\sum_{\rho(i)\in S}\omega_i\mathbb{M}_i = (1,0,...,0)$. Next, it computes

$$\begin{aligned} B_0 &= \prod_{\rho(i)\in S}(e(tk_{2,i},D_i)\cdot e(C_i,tk_1))^{\omega_i} \\ &= \prod_{\rho(i)\in S}(e(T_i^{t_xr},g^{r_i})\cdot e(g^{a\lambda_i}T_{\rho(i)}^{-r_i},g^{t_xr}\bar{R}_{x,0}))^{\omega_i} \\ &= e(g,g)^{at_xr\sum_{\rho(i)\in S}\lambda_i\omega_i} = e(g,g)^{at_xrs}, \end{aligned} \tag{10}$$

and $B_1 = e(tk_0, C_0) = e(g,g)^{2\alpha s}\cdot e(g,g)^{at_xrs}\cdot e((u^{id}h)^r, g^s)\cdot e((u_0^th_0)^{s_x}, g^s)$ and $B_2 = e(tk_3, C_t) = e(g^{s_x}\bar{R}_{x,3}, (u_0^th_0)^s) = e(g^{s_x}, (u_0^th_0)^s)$. Finally, the algorithm computes

$$D = \frac{B_1}{B_0\cdot B_2} = e(g,g)^{2\alpha s}\cdot e((u^{id}h)^r, g^s). \tag{11}$$

and returns the transformed ciphertext $CT' = (C, C_0, C_t, D)$
– **Decrypt**$(PK, id, dk_{id,t}, CT') \rightarrow M$**:** Set $dk_{id,t} = (dk_0, dk_1)$, it computes

$$\begin{aligned} D_0 &= e(dk_0, C_0) = e(g,g)^{\alpha s}\cdot e((u^{id}h)^r, g^s)\cdot e((u_0^th_0)^{r_t}, g^s) \\ D_1 &= e(dk_1, C_t) = e(g^{r_t}, (u_0^th_0)^s) \end{aligned} \tag{12}$$

and then computes

$$M = \frac{C\cdot D_0}{D\cdot D_1} = \frac{M\cdot e(g,g)^{2\alpha s}\cdot e((u^{id}h)^r, g^s)\cdot e((u_0^th_0)^{r_t}, g^s)}{e(g,g)^{2\alpha s}\cdot e((u^{id}h)^r, g^s)\cdot e(g^{r_t}, (u_0^th_0)^s)}. \tag{13}$$

– **Revoke**$(id, t, \mathsf{RL}, st) \rightarrow \mathsf{RL}$**:** If a user's identity id is revoked at time period t, the algorithm adds (x, t) into the revocation list RL for all nodes x associated with identity id.

5 Security Analysis

In this section, we provide a security analysis of our SR-ABE scheme in the multi-user setting. Following the basic technique of dual system encryption [11,13], we define two additional structures: semi-functional ciphertexts and semi-functional keys. These will not be used in the real system, but will be essential in the security proof.

Semi-functional Ciphertext: Let g_2 be a generator of G_{p_2}. Randomly pick $c, z_{t^*}, z_1, \ldots, z_l, \gamma_1, \ldots, \gamma_l \in \mathbb{Z}_N$ and $\vec{u} \in \mathbb{Z}_N^n$. Then:

$$\begin{aligned} &C_0 = g^s g_2^c, \qquad C_t = (u_0^t h_0)^s g_2^{c z_{t^*}}, \\ &C_i = g^{a\lambda_i} T_{\rho(i)}^{-r_i} g_2^{\mathbb{M}_i \vec{u} + \gamma_i z_{\rho(i)}}, \qquad D_i = g^{r_i} g_2^{-\gamma_i} \quad \forall i \in [1, l]. \end{aligned} \tag{14}$$

Semi-functional Key: There are two types of semi-functional key. A semi-functional key of type 1 is formed as follows. Chose random elements $z_{id}, b, d, z_t \in \mathbb{Z}_N$, set

$$SK_{id} = g^{\alpha}(u^{id}h)^r \cdot R_3 \cdot g_2^{b z_{id}}; \tag{15}$$

for each node $x \in \mathsf{Path}(\mathsf{BT}, \theta)$, set

$$\begin{aligned} &K_x = g^{\alpha + a t_x r}((u^{id}h)^r / g_x) \cdot R_{x,0} \cdot g_2^{d t_x} \cdot g_2^{b z_{id}}, \quad L_x = g^{t_x r} \cdot \bar{R}_{x,0} \cdot g_2^{b t_x}, \\ &K_{x,i} = T_i^{t_x r} R_{x,i} \cdot g_2^{b t_x z_{\rho(i)}} \quad \forall i \in S; \end{aligned} \tag{16}$$

for each node $x \in \mathsf{KUNodes}(\mathsf{BT}, \mathsf{RL}, t)$, pick random element $\gamma_x \in \mathbb{Z}_N$, set:

$$Q_{x,0,t} = g^{\alpha} g_x \cdot (u_0^t h_0)^{s_x} \hat{R}_{x,3} \cdot g_2^{\gamma_x z_t}, \quad Q_{x,1,t} = g^{s_x} \bar{R}_{x,3} \cdot g_2^{\gamma_x}. \tag{17}$$

A semi-functional key of type 2 is formed as:

$$SK_{id} = g^{\alpha}(u^{id}h)^r \cdot R_3 \cdot g_2^{b z_{id}}; \tag{18}$$

for $\forall x \in \mathsf{Path}(\mathsf{BT}, \theta)$,

$$\begin{aligned} &K_x = g^{\alpha + a t_x r}((u^{id}h)^r / g_x) \cdot R_{x,0} \cdot g_2^{d t_x} \cdot g_2^{b z_{id}}, \qquad L_x = g^{t_x r} \cdot \bar{R}_{x,0}, \\ &K_{x,i} = T_i^{t_x r} R_{x,i} \quad \forall i \in S; \end{aligned} \tag{19}$$

for $\forall x \in \mathsf{KUNodes}(\mathsf{BT}, \mathsf{RL}, t)$,

$$Q_{x,0,t} = g^{\alpha} g_x \cdot (u_0^t h_0)^{s_x} \hat{R}_{x,3} \cdot g_2^{\gamma_x z_t}, \quad Q_{x,1,t} = g^{s_x} \bar{R}_{x,3} \cdot g_2^{\gamma_x}. \tag{20}$$

It should be noted that if we use a semi-functional key to decrypt a semi-functional ciphertext, we get the following additional term:

$$e(g_2, g_2)^{(cd-bu_1)t_x + c\gamma_x(z_t - z_{t^*})}, \tag{21}$$

where u_1 denotes the first coordinate of $\vec{u}$ (i.e. $(1, 0, \ldots, 0) \cdot \vec{u}$). Note that the values z_i, which are used to hide u_1 from an attacker, are common to semi-functional ciphertexts and semi-functional keys of type 1. Similar to [11], our security proof will depend on the restriction that each attribute can only be used once in the row labeling of an access matrix. Based on this restriction, the attacker gains very limited information-theoretic knowledge of z_i.

We call a semi-functional key of type 1 "nominally" semi-functional if $(cd - bu_1)t_x + c\gamma_x(z_t - z_{t^*}) = 0$. If we use such a key to decrypt a corresponding semi-functional ciphertext, the decryption still succeeds.

We will prove the security of our system from Assumptions 1, 2, and 3 using a hybrid argument over a sequence of games. The first game, $\mathbf{Game}_{Real}$, is the real security game (the ciphertext and all the keys are normal). The next game $\mathbf{Game}_{Restricted}$ will be like the real security game except that the attacker cannot ask for the transformation key update information and decryption keys for times which are equal to the challenge time t^* modulo p_2. Also, the attacker cannot ask for long-term transformation keys, private keys and decryption keys for identities $id_i \neq id_j$ modulo N such that $id_i = id_j$ modulo p_2. In the next game, $\mathbf{Game}_0$, all of the keys will be normal, but the challenge ciphertext will be semi-functional. We let q denote the number of key queries made by the attacker. For k from 1 to q, we define:

$\mathbf{Game}_{k,1}$ In this game, the challenge ciphertext is semi-functional, the first $k-1$ keys are semi-functional of type 2, the k^{th} key is semi-functional of type 1, and the remaining keys are normal.

$\mathbf{Game}_{k,2}$ In this game, the challenge ciphertext is semi-functional, the first k keys are semi-functional of type 2, and the remaining keys are normal.

In $\mathbf{Game}_{q,2}$, all of the keys are semi-functional of type 2. In the final game, $\mathbf{Game}_{Final}$, all keys are semi-functional of type 2 and the ciphertext is a semi-functional encryption of a random message, independent of the two messages provided by the attacker. Thus, the adversary's advantage in winning the final game is 0. Now, we will prove that these games are indistinguishable. The proof of Lemmas 2, 4, 5 is given in Appendix.

Lemma 1. *Supposed that a PPT adversary $\mathcal{A}$ can distinguish the $\mathbf{Game}_{Real}$ and $\mathbf{Game}_{Restricted}$ with a non-negligible advantage $\epsilon > 0$, then there exists a PPT simulator $\mathcal{B}$ that can break either Assumption 1 or Assumption 2 with advantage $\geq \frac{\epsilon}{2}$.*

Proof. This proof is similar with Lemma 1 of [13], so we omit it here.

Lemma 2. *Supposed that a PPT adversary $\mathcal{A}$ can distinguish $\mathbf{Game}_{Restricted}$ and $\mathbf{Game}_0$ with a non-negligible advantage $\epsilon > 0$, then there exists a PPT simulator $\mathcal{B}$ that can break Assumption 1 with advantage ϵ.*

Lemma 3. *Supposed that a PPT adversary $\mathcal{A}$ can distinguish the* $\mathbf{Game}_{k-1,2}$ *and* $\mathbf{Game}_{k,1}$ *with a non-negligible advantage $\epsilon > 0$, then there exists a PPT simulator $\mathcal{B}$ that can break Assumption 2 with advantage ϵ.*

Proof. $\mathcal{B}$ is given $(g, X_1X_2, X_3, Y_2Y_3, T)$ and simulates $\mathbf{Game}_{k-1,2}$ or $\mathbf{Game}_{k,1}$ with $\mathcal{A}$. It randomly picks $a, \alpha, a_0, a_1, b_0, b_1 \in \mathbb{Z}_N$ and $s_i \in \mathbb{Z}_N$ for each attribute i in the system, then sets $u = g^{a_1}, h = g^{b_1}, u_0 = g^{a_0}, h_0 = g^{b_0}$ and returns the public parameters $PK = \{N, g, g^a, u, h, u_0, h_0, e(g,g)^\alpha, \{T_i = g^{s_i}, \forall i\}\}$ to $\mathcal{A}$.

To make the first $k-1$ keys semi-functional of type 2, $\mathcal{B}$ responds to each key request as follows. Choose an undefined leaf node θ from BT and store id in this node. Randomly choose $f, r, z_{id}, d', z_t \in \mathbb{Z}_N$, $R'_3 \in G_{p_3}$, set

- $SK_{id} = g^\alpha (u^{id}h)^r \cdot R'_3 \cdot (Y_2Y_3)^{fz_{id}}$;
- For each $x \in \mathsf{Path}(\mathsf{BT}, \theta)$, fetch g_x from the node x (if g_x has not been defined, randomly pick $g_x \in G_{p_1}$ and store in node x), randomly choose $t_x \in \mathbb{Z}_N$, $R'_{x,0}, \bar{R}_{x,0}, \{R_{x,i}\}_{i \in S} \in G_{p_3}$, set

$$\begin{aligned} &K_x = g^{\alpha + at_xr}((u^{id}h)^r/g_x) \cdot R'_{x,0} \cdot (Y_2Y_3)^{d't_x + fz_{id}}, \quad L_x = g^{t_xr} \cdot \bar{R}_{x,0}, \\ &K_{x,i} = T_i^{t_xr} R_{x,i} \quad \forall i \in S; \end{aligned} \tag{22}$$

- For each $x \in \mathsf{KUNodes}(\mathsf{BT}, \mathsf{RL}, t)$, fetch g_x from the node x, randomly choose $s_x, \gamma'_x \in \mathbb{Z}_N$ and $\hat{R}'_{x,3}, \bar{R}'_{x,3} \in G_{p_3}$, set

$$Q_{x,0,t} = g^\alpha g_x (u_0^t h_0)^{s_x} \cdot \hat{R}'_{x,3} (Y_2Y_3)^{\gamma'_x z_t}, \; Q_{x,1,t} = g^{s_x} \cdot \bar{R}'_{x,3} (Y_2Y_3)^{\gamma'_x}. \tag{23}$$

The above keys are properly distributed. To make normal keys for requests $> k$, $\mathcal{B}$ can simply run the key generation algorithm by using the MSK.

To answer the k^{th} key request, $\mathcal{B}$ will implicity set g^r be the G_{p_1} part of T. Choose an undefined leaf node θ from BT and store id in this node. Randomly choose $R'_3 \in G_{p_3}$, set

- $SK_{id} = g^\alpha T^{a_1 id + b_1} \cdot R'_3$;
- For each $x \in \mathsf{Path}(\mathsf{BT}, \theta)$, fetch g_x from the node x, choose random elements $t_x \in \mathbb{Z}_N$, $R'_{x,0}, \bar{R}'_{x,0}, \{R'_{x,i}\}_{i \in S} \in G_{p_3}$, set

$$\begin{aligned} &K_x = g^\alpha T^{at_x} (T^{a_1 id + b_1}/g_x) \cdot R'_{x,0}, \quad L_x = T^{t_x} \cdot \bar{R}'_{x,0}, \\ &K_{x,i} = T^{s_i t_x} R'_{x,i} \quad \forall i \in S; \end{aligned} \tag{24}$$

- For each $x \in \mathsf{KUNodes}(\mathsf{BT}, \mathsf{RL}, t)$, fetch g_x from the node x, choose random elements $\hat{R}'_{x,3}, \bar{R}'_{x,3} \in G_{p_3}$ and $s'_x \in \mathbb{Z}_N$, set

$$Q_{x,0,t} = g^\alpha g_x \cdot (T^{a_0 t + b_0})^{s'_x} \hat{R}'_{x,3}, \qquad Q_{x,1,t} = T^{s'_x} \bar{R}'_{x,3}. \tag{25}$$

If $T \in G_{p_1p_3}$, we set $s_x = rs'_x$ and this is a properly distributed normal key. Otherwise $T \in G$, this is a semi-functional key of type 1. In this case, we implicitly set $z_i = s_i$. If we set g_2^b as the G_{p_2} part of T, then we have that $d = ba$, $z_{id} = a_1 id + b_1$, $z_t = a_0 t + b_0$, $\gamma_x = bs'_x$, $s_x = rs'_x$. Note that the value of s'_x, s_i modulo p_2 are uncorrelated from these values modulo p_1. Since $\phi(id) = a_1 id + b_1$ ($\varphi(t) = a_0 t + b_0$) is pairwise independent function modulo p_2, as long as $id_i \neq id_j (\mathsf{mod}\, p_2)$ ($t \neq t^* (\mathsf{mod}\, p_2)$), z_{id_i} and z_{id_j} (z_t and z_{t^*}) will seem randomly distributed to $\mathcal{A}$. If $id_i = id_j (\mathsf{mod}\, p_2)$ or $t = t^* (\mathsf{mod}\, p_2)$, then $\mathcal{A}$ has made invalid key requests, this is why we use additional restrictions.

$\mathcal{A}$ sends $\mathcal{B}$ two messages (M_0, M_1), a challenge access matrix $(\mathbb{M}^*, \rho)$ and a challenge time t^*. To generate the semi-functional challenge ciphertext CT^*, $\mathcal{B}$ will implicitly set $g^s = X_1$ and $g_2^c = X_2$. It randomly chooses $u_2, \ldots, u_n \in \mathbb{Z}_N$, $r'_i \in \mathbb{Z}_N$ for $i \in [1, l]$ and a random bit $\beta \in \{0, 1\}$ and sets $\vec{u}' = (a, u_2, \ldots, u_n)$. Finally, $\mathcal{B}$ generates the challenge ciphertext CT^* as:

$$CT^* = \left\{ \begin{matrix} C = M_\beta \cdot e(g^\alpha, X_1X_2), C_0 = X_1X_2, C_t = (X_1X_2)^{a_0 t^* + b_0} \\ C_i = (X_1X_2)^{\mathbb{M}_i^* \vec{u}'} T^{-r'_i s_{\rho(i)}}, D_i = (X_1X_2)^{r'_i} \quad \forall i \end{matrix} \right\}. \tag{26}$$

We set $\vec{v} = sa^{-1}\vec{u}'$ and $\vec{u} = c\vec{u}'$ (i.e., $u_1 = ca$). Then, s is shared in the G_{p_1} while ca is shared in the G_{p_2}. We also implicitly set $r_i = sr'_i$ and $\gamma_i = -cr'_i$. The values $z_{\rho(i)} = s_{\rho(i)}$ match those in the k^{th} key if it is semi-functional key of type 1, as required.

The k^{th} key and challenge ciphertext are almost properly distributed. However, the first coordinate of $\vec{u}$ (i.e., u_1) is correlated with the value of a modulo p_2, since a also appears in the k^{th} key if it is semi-functional. We argue that this is information theoretically hidden from the adversary $\mathcal{A}$, who cannot request any keys that can decrypt the challenge ciphertext. This argument has been carefully proved in Lemma 8 of [11], so we omit it here.

In addition, with the setting of $z_{id} = a_1 id + b_1$ and $z_t = a_0 t + b_0$, if $\mathcal{B}$ use the k^{th} key to decrypt a semi-functional ciphertext, which is embedded with the same time period with k^{th} key query, we would have $(cd - bu_1)t_x + c\gamma_x(z_t - z_t) = (cba - bca)t_x + c\gamma_x(z_t - z_t) = 0 \mod p_2$, so the k^{th} key is either normal key or nominally semi-functional key.

Thus, if $T \in G_{p_1 p_3}$, then $\mathcal{B}$ has properly simulated $\mathbf{Game}_{k-1,2}$. Otherwise, $T \in G$ and $\mathcal{B}$ has properly simulated $\mathbf{Game}_{k,1}$. Therefore, $\mathcal{B}$ can use the output of $\mathcal{A}$ to gain advantage negligibly close to ϵ in breaking Assumption 2. □

Lemma 4. *Supposed that a PPT adversary $\mathcal{A}$ can distinguish the* $\mathbf{Game}_{k,1}$ *and* $\mathbf{Game}_{k,2}$ *with a non-negligible advantage $\epsilon > 0$, then there exists a PPT simulator $\mathcal{B}$ that can break Assumption 2 with advantage ϵ.*

Lemma 5. *Supposed that a PPT adversary $\mathcal{A}$ can distinguish the* $\mathbf{Game}_{q,2}$ *and* $\mathbf{Game}_{Final}$ *with a non-negligible advantage $\epsilon > 0$, then there exists a PPT simulator $\mathcal{B}$ that can break Assumption 3 with advantage ϵ.*

6 Conclusion

In this paper, we propose the first (fully secure) SR-ABE scheme with DKER, which is provably secure in multi-user setting. The construction of our scheme relies on the basic technique of dual encryption system in composite order bilinear groups. In our security proof, we inherited the shortcomings of the one-use restriction in the fully secure ABE scheme [11], meaning that a single attribute could only be used once in a policy. Later, [14] relaxed the restriction and allowed unrestricted use of attributes while still proving full security in the standard model. However, these frameworks only work in composite-order bilinear groups, where the computations (especially pairing operation) are very slow. In practice, prime-order bilinear groups are preferable because they provide more efficient and compact instantiations. Our SR-ABE can be further improved by using the techniques of above works. Due to the limited space, we leave it as future work.

Acknowledgments. We thank anonymous reviewers for helpful feedback.

A Proof of Lemma 2

Proof. $\mathcal{B}$ is given (g, X_3, T) and simulates $\mathbf{Game}_{Restricted}$ or $\mathbf{Game}_0$ with $\mathcal{A}$. It sets the public parameters as follows. It randomly picks $a, \alpha, a_0, a_1, b_0, b_1 \in \mathbb{Z}_N$ and $s_i \in \mathbb{Z}_N$ for each attribute i in the system, then sets $u = g^{a_1}, h = g^{b_1}, u_0 = g^{a_0}, h_0 = g^{b_0}$, returns the public parameters to $\mathcal{A}$ as:

$$PK = \{N, g, g^a, u, h, u_0, h_0, e(g,g)^\alpha, \{T_i = g^{s_i}, \forall i\}\}, \tag{27}$$

and keeps $MSK = \{\alpha, X_3\}$ as secret. In this case, $\mathcal{B}$ can answer any normal key query (including Create(id,S), Corrupt(id), TKeyUp(t), DecKG(id,t)) from $\mathcal{A}$ by running the corresponding key generation algorithm with MSK.

$\mathcal{A}$ sends $\mathcal{B}$ two messages (M_0, M_1), a challenge access matrix $(\mathbb{M}^*, \rho)$ and a challenge time t^*. To generate the challenge ciphertext CT^*, $\mathcal{B}$ will implicitly set g^s to be the G_{p_1} part of T (T is the product of g^s and possible an element of G_{p_2}). It randomly chooses $v'_2, \ldots, v'_n \in \mathbb{Z}_N$, $r'_i \in \mathbb{Z}_N$ for $i \in [1, l]$, $\beta \in \{0, 1\}$ and sets $\vec{v}' = (1, v'_2, \ldots, v'_n)^\perp$. Finally, $\mathcal{B}$ generates the challenge ciphertext CT^* as:

$$CT^* = \left\{ \begin{array}{ll} C = M_\beta \cdot e(g^\alpha, T), \quad C_0 = T, \quad C_t = T^{a_0 t^* + b_0} \\ C_i = T^{a\mathbb{M}^*_i \vec{v}'} T^{-r'_i s_{\rho(i)}}, \quad D_i = g^{r'_i} \quad \forall i \end{array} \right\}. \tag{28}$$

We note that this implicitly sets $\vec{v} = (s, sv'_2, \ldots, sv'_n)$ and $r_i = sr'_i$. Modulo p_1, v is a random vector with first coordinate s and r_i is a random value. Thus, if $T \in G_{p_1}$, CT^* is a properly distributed normal ciphertext. Otherwise, $T \in G_{p_1 p_2}$, we let g_2^c as the G_{p_2} part of T (i.e. $T = g^s g_2^c$). We then have a semi-functional ciphertext with $z_{t^*} = a_0 t^* + b_0$, $u = ca\vec{v}'$, $\gamma_i = -cr'_i$, and $z_{\rho(i)} = s_{\rho(i)}$. By the Chinese Remainder Theorem, $a_0, b_0, a, v'_2, \ldots, v'_n, r'_i, s_{\rho(i)}$ modulo p_2 are uncorrelated from these values modulo p_1, so CT^* is a properly distributed semi-functional ciphertext. Therefore, $\mathcal{B}$ can break Assumption 1 with advantage ϵ by the output of $\mathcal{A}$. $\square$

B Proof of Lemma 4

Proof. $\mathcal{B}$ is given $(g, X_1X_2, X_3, Y_2Y_3, T)$ and simulates $\mathbf{Game}_{k,1}$ or $\mathbf{Game}_{k,2}$ with $\mathcal{A}$. It randomly picks $a, \alpha, a_0, a_1, b_0, b_1 \in \mathbb{Z}_N$ and $s_i \in \mathbb{Z}_N$ for each attribute i in the system, then sets $u = g^{a_1}, h = g^{b_1}, u_0 = g^{a_0}, h_0 = g^{b_0}$ and returns the public parameters $PK = \{N, g, g^a, u, h, u_0, h_0, e(g,g)^\alpha, \{T_i = g^{s_i}, \forall i\}\}$ to $\mathcal{A}$.

The first $k-1$ semi-functional keys of type 2, the normal keys $> k$, and the challenge ciphertext are all constructed the same as the above lemma. Hence, the ciphertext is sharing the value ac in the G_{p_2} subgroup. However, this will not be correlated with the k^{th} key any way, so the value is random modulo p_2. To answer the k^{th} key request, $\mathcal{B}$ choose a random element $R'_3 \in G_{p_3}$ and set

- $SK_{id} = g^\alpha T^{a_1 id + b_1} \cdot R'_3$;
- For each $x \in \mathsf{Path}(\mathsf{BT}, \theta)$, fetch g_x from the node x, choose random elements $t_x \in \mathbb{Z}_N$, $R'_{x,0}, \bar{R}'_{x,0}, \{R'_{x,i}\}_{i \in S} \in G_{p_3}$, and an additional $h_x \in \mathbb{Z}_N$, set

$$\begin{aligned} &K_x = g^\alpha T^{at_x}(T^{a_1 id + b_1}/g_x) \cdot R'_{x,0} \cdot (Y_2Y_3)^{h_x}, \qquad L_x = T^{t_x} \cdot \bar{R}'_{x,0}, \\ &K_{x,i} = T^{s_i t_x} R'_{x,i} \quad \forall i \in S; \end{aligned} \tag{29}$$

- For each $x \in \mathsf{KUNodes}(\mathsf{BT}, \mathsf{RL}, t)$, fetch g_x from the node x, choose random elements $\hat{R}_{x,3}, \bar{R}_{x,3} \in G_{p_3}$ and $s'_x \in \mathbb{Z}_N$, set

$$Q_{x,0,t} = g^\alpha g_x \cdot (T^{a_0 t + b_0})^{s'_x} \hat{R}_{x,3}, \quad Q_{x,1,t} = T^{s'_x} \bar{R}_{x,3}. \tag{30}$$

Note that we add the $(Y_2Y_3)^{h_x}$ term. This randomizes the G_{p_2} part of K_x, so the key is no longer nominally semi-functional. If we use the k^{th} key to decrypt the semi-functional ciphertext, the decryption would fail.

Thus, if $T \in G_{p_1p_3}$, then $\mathcal{B}$ has properly simulated $\mathbf{Game}_{k,2}$. Otherwise, $T \in G$, then $\mathcal{B}$ has properly simulated $\mathbf{Game}_{k,1}$. Therefore, $\mathcal{B}$ can use the output of $\mathcal{A}$ to gain advantage to ϵ in breaking Assumption 2. $\square$

C Proof of Lemma 5

Proof. $\mathcal{B}$ is given $(g, g^\alpha X_2, X_3, g^s Y_2, Z_2, T)$ and simulates $\mathbf{Game}_{q,2}$ or $\mathbf{Game}_{Final}$ with $\mathcal{A}$. It randomly picks $a, , a_0, a_1, b_0, b_1 \in \mathbb{Z}_N$ and $s_i \in \mathbb{Z}_N$ for each attribute i in the system, then sets $u = g^{a_1}, h = g^{b_1}, u_0 = g^{a_0}, h_0 = g^{b_0}$ and returns $PK = \{N, g, g^a, u, h, u_0, h_0, e(g, g^\alpha X_2) = e(g,g)^\alpha, \{T_i = g^{s_i}, \forall i\}\}$ to $\mathcal{A}$.

To make semi-functional keys of type 2, randomly choose $f, r, z_{id}, d', z_t \in \mathbb{Z}_N$, $R'_3 \in G_{p_3}$ and set

- $SK_{id} = g^\alpha (u^{id} h)^r \cdot R'_3 \cdot Z_2^{f z_{id}}$;
- For each $x \in \mathsf{Path}(\mathsf{BT}, \theta)$, fetch g_x from the node x, randomly choose $t_x \in \mathbb{Z}_N$, $R'_{x,0}, \bar{R}_{x,0}, \{R_{x,i}\}_{i \in S} \in G_{p_3}$, set

$$\begin{aligned} &K_x = g^{\alpha + at_x r}((u^{id}h)^r/g_x) \cdot R'_{x,0} \cdot Z_2^{d' t_x + f z_{id}}, \quad L_x = g^{t_x r} \cdot \bar{R}'_{x,0}, \\ &K_{x,i} = T_i^{t_x r} R'_{x,i} \quad \forall i \in S; \end{aligned} \tag{31}$$

– For each $x \in \mathsf{KUNodes}(\mathsf{BT}, \mathsf{RL}, t)$, fetch g_x from the node x, randomly choose $s_x, \gamma'_x \in \mathbb{Z}_N$ and $\hat{R}'_{x,3}, \bar{R}'_{x,3} \in G_{p_3}$, set

$$Q_{x,0,t} = g^{\alpha} g_x \cdot (u_0^t h_0)^{s_x} \hat{R}'_{x,3} \cdot Z_2^{\gamma'_x z_t}, \quad Q_{x,1,t} = g^{s_x} \bar{R}'_{x,3} \cdot Z_2^{\gamma'_x}. \tag{32}$$

$\mathcal{A}$ sends $\mathcal{B}$ two messages (M_0, M_1), a challenge access matrix $(\mathbb{M}^*, \rho)$ and a challenge time t^*. $\mathcal{B}$ chooses $u_2, \ldots, u_n, r'_i \in \mathbb{Z}_N$, a random bit $\beta \in \{0, 1\}$ and sets $\vec{u}' = (a, u_2, \ldots, u_n)$. Finally, $\mathcal{B}$ generates the challenge ciphertext CT^* as:

$$CT^* = \left\{ \begin{array}{l} C = M_\beta \cdot T, C_0 = g^s Y_2, C_t = (g^s Y_2)^{a_0 t^* + b_0} \\ C_i = (g^s Y_2)^{\mathbb{M}_i^* \vec{u}'} (g^s Y_2)^{-r'_i s_{\rho(i)}}, D_i = (g^s Y_2)^{r'_i} \quad \forall i \end{array} \right\}. \tag{33}$$

We set $Y_2 = g_2^c$, $\vec{v} = s a^{-1} \vec{u}'$ and $\vec{u} = c\vec{u}'$ (i.e., $u_1 = ac$), so s is shared in the G_{p_1} and ca is shared in the G_{p_2}. This implicitly sets $u_1 = ca$, $r_i = sr'_i$ and $\gamma_i = -cr'_i$.

Thus, if $T = e(g, g)^{\alpha s}$, then $\mathcal{B}$ has properly simulated $\mathbf{Game}_{q,2}$ and CT^* is a semi-functional ciphertext with encryption of M_β. Otherwise, $T \in G_T$, then $\mathcal{B}$ has properly simulated $\mathbf{Game}_{Final}$ and CT^* is a semi-functional ciphertext with encryption of a random message in G_T. Therefore, $\mathcal{B}$ can use the output of $\mathcal{A}$ to gain advantage to ϵ in breaking Assumption 3. □

References

1. Attrapadung, N., Imai, H.: Attribute-based encryption supporting direct/indirect revocation modes. In: Parker, M.G. (ed.) IMACC 2009. LNCS, vol. 5921, pp. 278–300. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10868-6_17
2. Beimel, A.: Secure schemes for secret sharing and key distribution. PhD thesis Israel institute of technology Technion (1996)
3. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy 2007, pp. 321–334 (2007)
4. Boldyreva, A., Goyal, V., Kumar, V.: Identity-based encryption with efficient revocation. In: CCS 2008, pp. 417–426 (2008)
5. Boldyreva, A., Goyal, V., Kumar, V.: Identity-based encryption with efficient revocation. IACR Cryptology ePrint Archive **2012**, 52 (2012)
6. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
7. Chen, J., Lim, H.W., Ling, S., Wang, H., Nguyen, K.: Revocable identity-based encryption from lattices. In: Susilo, W., Mu, Y., Seberry, J. (eds.) ACISP 2012. LNCS, vol. 7372, pp. 390–403. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31448-3_29
8. Cui, H., Deng, R.H., Li, Y., Qin, B.: Server-aided revocable attribute-based encryption. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9879, pp. 570–587. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45741-3_29
9. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: CCS 2006, pp. 89–98 (2006)

10. Katsumata, S., Matsuda, T., Takayasu, A.: Lattice-based revocable (hierarchical) IBE with decryption key exposure resistance. In: PKC 2019, pp. 441–471 (2019)
11. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_4
12. Lewko, A.B., Sahai, A., Waters, B.: Revocation systems with very small private keys. In: IEEE Symposium on Security and Privacy, S&P 2010, pp. 273–285. IEEE Computer Society (2010)
13. Lewko, A., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_27
14. Lewko, A., Waters, B.: New proof methods for attribute-based encryption: achieving full security through selective techniques. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 180–198. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_12
15. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: CRYPTO, pp. 41–62 (2001)
16. González-Nieto, J.M., Manulis, M., Sun, D.: Fully private revocable predicate encryption. In: Susilo, W., Mu, Y., Seberry, J. (eds.) ACISP 2012. LNCS, vol. 7372, pp. 350–363. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31448-3_26
17. Qin, B., Deng, R.H., Li, Y., Liu, S.: Server-aided revocable identity-based encryption. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9326, pp. 286–304. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24174-6_15
18. Qin, B., Zhao, Q., Zheng, D., Cui, H.: Server-aided revocable attribute-based encryption resilient to decryption key exposure. In: Capkun, S., Chow, S.S.M. (eds.) CANS 2017. LNCS, vol. 11261, pp. 504–514. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02641-7_25
19. Qin, B., Zhao, Q., Zheng, D., Cui, H.: (Dual) server-aided revocable attribute-based encryption with decryption key exposure resistance. Inf. Sci. **490**, 74–92 (2019)
20. Rouselakis, Y., Waters, B.: Practical constructions and new proof methods for large universe attribute-based encryption. In: Sadeghi, A., Gligor, V.D., Yung, M. (eds.) CCS 2013, pp. 463–474. ACM (2013)
21. Sahai, A., Seyalioglu, H., Waters, B.: Dynamic credentials and ciphertext delegation for attribute-based encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 199–217. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_13
22. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27
23. Seo, J.H., Emura, K.: Revocable identity-based encryption revisited: security model and construction. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 216–234. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_14
24. Waters, B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_36