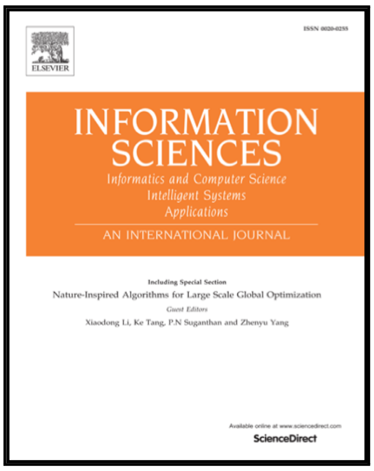

Revocable Attribute-Based Encryption with Decryption Key Exposure Resistance and Ciphertext Delegation

Shengmin Xu, Guomin Yang, Yi Mu

PII: S0020-0255(18)30921-6
DOI: https://doi.org/10.1016/j.ins.2018.11.031
Reference: INS 14075

To appear in: *Information Sciences*

Received date: 17 March 2018
Revised date: 5 September 2018
Accepted date: 17 November 2018

Revocable Attribute-Based Encryption with Decryption Key Exposure Resistance and Ciphertext Delegation

Shengmin Xu[a], Guomin Yang*[a], Yi Mu[b]

[a] *Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Australia*
[b] *Fujian Provincial Key Laboratory of Network Security and Cryptology, Fujian Normal University, Fuzhou, Fujian, China*

Abstract

Attribute-based encryption (ABE) enables fine-grained access control over encrypted data. A practical and popular approach for handing revocation in ABE is to use the indirect revocation mechanism, in which a key generation centre (KGC) periodically broadcasts key update information for all data users over a public channel. Unfortunately, existing RABE schemes are vulnerable to decryption key exposure attack which has been well studied in the identity-based setting. In this paper, we introduce a new notion for RABE called re-randomizable piecewise key generation by allowing a data user to re-randmomize the combined secret key and the key update to obtain the decryption key, and the secret key is unrecoverable even both the decryption key and the key update are known by the attacker. We then propose a new primitive called re-randomizable attribute-based encryption (RRABE) that can achieve both re-randomizable piecewise key generation and ciphertext delegation. We also refine the existing security model for RABE to capture decryption key exposure resistance and present a generic construction of RABE from RRABE. Finally, by applying our generic transformation, we give a concrete RABE scheme achieving decryption key exposure resistance and ciphertext delegation simultaneously.

Key words: Access control, attribute-based encryption, revocable storage,

*Corresponding author
Email addresses: `gyang@uow.edu.au` (Guomin Yang*)

decryption key exposure

1. Introduction

User revocation is a critical issue that must be addressed properly in any security systems, e.g., due to expiration or change of the user membership and user credentials being stolen/compromised/misused. Without a secure revocation mechanism, public key cryptosystems are hardly useful in practice. Providing efficient user revocation has been the subject of attention in cryptographic research under different key management and distribution settings such the traditional Public Key Infrastructure (PKI) [1, 2, 3, 4, 5], the identity-based setting [6, 7, 8, 9, 10, 11], as well as the attribute-based setting [12, 13, 14, 15, 16]. The existing revocation mechanisms can be categorized into direct revocation, indirect revocation and server-aid revocation. The details are given below.

Direct Revocation. In the conventional public key management systems, revocation can be done via certificate revocation list (CRL), certificate revocation system (CRS) and certificate revocation tree (CRT). These are referred to as direct revocation. Unfortunately, such an approach breaks the implied anonymity when being applied to the identity-based or the attribute-based settings.

Indirect Revocation. In IBE and ABE settings, indirect revocation is commonly used. It enforces revocation by letting the KGC publish the key update periodically in such a way that only non-revoked users can update their keys, and revoked users' keys are implicitly rendered useless. Specifically, indirect revocation also consists of two different types.

Type-I. The first type of indirect revocation was proposed by Boneh and Franklin [6]. The KGC keeps a revocation list, and revocation is performed by transmitting new private keys to all non-revoked users at each revocation epoch. Such a method puts a very heavy burden on the KGC since it needs to compute new private keys frequently for all non-revoked users and keeps secure channels with all non-revoked users for transmitting new private keys each time.

Type-II. To reduce the overhead of the KGC, Boldyreva et al. [7] proposed the second type of indirect revocation as in Figure 1, which significantly improves the efficiency of previous solutions. The decryption key of each user is split into a secret key and an update key based on fuzzy IBE scheme [17]. With this method, the KGC just publishes the key update over a public channel, and the overhead of the KGC reduces from linear to logarithmic in the number of users due to the tree-based data structure [4]. This revocation approach is one of formal treatments for revocable schemes since the receivers' privacy is preserved but others are not, and the user privacy is one of major concerns in many real applications. Hence, in this paper, we focus on this revocation method.

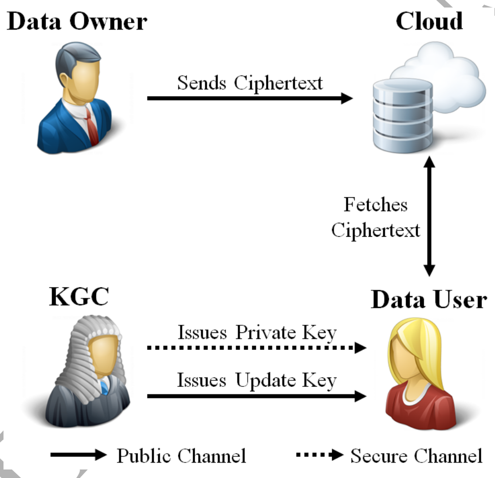

Figure 1: Type-II Indirect Revocation

Server-Aided Revocation. Indirect revocation fails to provide immediate revocation since the revocation only happens at the beginning of each revocation epoch. To address this problem, the server-aid revocation as in Figure 2 was proposed in IBE setting [11], and then extended to ABE setting [15]. The cloud service provider (CSP) needs to decrypt every requested ciphertext partially and then transfers them to the related non-revoked data user. Unfortunately, this method is cannot maintain the implied anonymity in IBE and ABE settings since the CSP must know the identity of the receiver before conducting the partial decryption. The only way to achieve the user privacy is the CSP partially decrypt the ciphertext for each non-revoked users, which significantly increases the cost in partially decryption phase. This revocation method is one of reasonable treatments for revocable schemes when the receivers' privacy is not essential in the system.

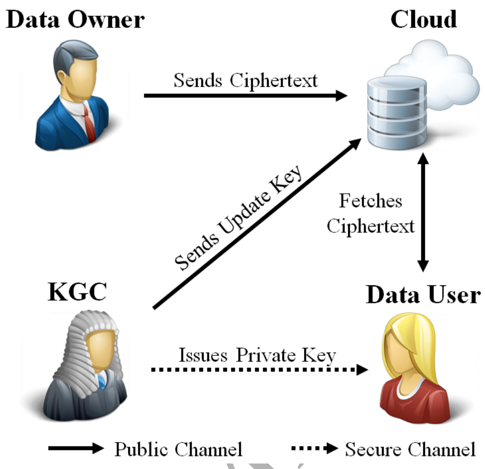

Figure 2: Server-Aided Revocation

When applying the aforementioned revocation mechanisms, some additional security issues/concerns would occur and below we revisit two important security issues related to user revocation.

Decryption Key Exposure Resistance. In each revocation epoch, the data user derives a short-term decryption key by combing the long-term secret key and the key update published by the KGC periodically. In the real applications, the decryption key is frequently used to decrypt data and may be leaked due to various attacks, e.g., key leakage attack [18] and side-channel attack [19]. Meanwhile, the increasing use of mobile devices with Internet connections also makes the frequently used decryption keys vulnerable to other threats such as malwares. If the user's long-term key can be calculated from the compromised decryption key and the key update, which is delivered through a public channel as illustrated in Figure 1, then the attacker is able to derive all the subsequent decryption keys. To overcome this issue, the concept of decryption key exposure resistance [10] was proposed in the identity-based setting to prevent the secret key exposure when the decryption key is leaked. However, such a problem has not been well addressed in the ABE setting. If the short-term decryption key is leaked, the user secret key can be recovered in many RABE systems [7, 8, 13, 14].

Ciphertext Delegation. The concept of ciphertext delegation [14] was introduced to prevent revoked users from accessing the data encrypted before the user is revoked. It allows the ciphertext to be updated by the CSP without accessing any secret information. Following the indirect revocation, the decryption key of each user is divided into an attribute-based secret key and a time-based key update, and the latter is updated in each revocation epoch for non-revoked users. Similarly, the time-related information in ciphertext is also updated by the CSP. Hence, revoked users cannot access messages encrypted before revocation.

In this paper, we introduce a new notion called re-randomizable piecewise key generation to resist decryption key exposure attack and a new cryptographic primitive named re-randomizable attribute-based encryption (RRABE) to achieve decryption key exposure resistance and ciphertext delegation simultaneously. We also provide a generic construction for RABE from RRABE. The comparison of our RABE scheme with some other RABE schemes [7, 13, 14, 15] is given in Table 1, where “ – ” denotes not-applicable. In server-aid revocable schemes, the CSP will first partially decrypt the ciphertext based on the receiver's information, thus ciphertext delegation is not considered in this revocation method. Note that the key exposure resistance considered in [15] is different from that in this work. In [15], the key exposure is against the partial decryption key (or transformation key) possessed by the CSP while in this paper we consider the exposure of the decryption key possessed by the user. We should also note that key exposure against the transformation key is unnecessary when the key updated is sent through via a public channel since the transformation key can be derived by anyone. On the other hand, since the user decryption key is static, it obviously cannot resist the decryption key exposure attack we consider here.

1.1. Related Work

Sahai and Waters [17] introduced ABE that allows users to selectively share their encrypted data at a fine-grained level. To enrich expressiveness of access control policies, Goyal et al. [20] and Bethencourt et al. [21] then proposed key-policy and ciphertext-policy ABE schemes, respectively. In key-policy ABE (KP-ABE) schemes, attribute sets are used to annotate ciphertexts, and private keys are associated with access structures that specify which ciphertexts the user will be entitled to decrypt. Ciphertext-policy ABE (CP-ABE) proceeds in a dual way, by assigning attribute sets to private keys and letting senders specify an access policy that receivers' attribute sets should comply with. However, the seminal works [17, 20, 21] of ABE schemes suffer problems of the size of the key and the ciphertext are linear to the attribute set and security proofs are under the selective model. Attrapadung et al. [22] proposed the first constant-size ABE and Lewko et al. [23] provided first fully secure ABE with dual encryption system [24], respectively. Unfortunately, the above schemes must define the attribute universe at setup phase or sacrifice the security by deploying the random oracle to scale up the attribute universe, Rouselakis and Waters [25] proposed large universe ABE schemes to overcome this problem. After that, many ABE schemes [26, 27, 28, 29, 30] have been proposed to improve the efficiency, security and functionality.

RIBE and RABE are the extensions of IBE and ABE settings by proving an efficient revocation mechanism. The issue of revocation of IBE setting was pointed by Boneh and Franklin [6] and they suggested that private keys can be renewed by appending current date at the end of identities as $id\|t$ (refers to type-I indirect revocation), where id is the user's identity and t is current date. However, such an approach is inefficient and unscalable since heavy workloads of the KGC to generate all non-revoked users' private keys and to keep secure channels each time. Then, many revocable schemes were proposed to solve this problem, but there is no efficient solution until Boldyreva et al. [7] proposed

Table 1: Comparison between our RABE scheme and existing RABE schemes

	BGK [7]	*AI* [13]	*SSW* [14]	*CDLQ* [15]	*Our RABE*
Revocation Mode	Indirect	Direct & Indirect	Indirect	Server-Aided	Indirect
Key Exposure Resistance	$\times$	$\times$	$\times$	$\times$	$\surd$
Ciphertext Delegation	$\times$	$\times$	$\surd$	–	$\surd$
Generic Construction	$\times$	$\times$	$\surd$	$\times$	$\surd$
User Privacy	$\surd$	$\times$	$\surd$	$\times$	$\surd$
Size of Public Parameter	$O(\Omega)$	$O(\Omega)$	$O(1)$	$O(1)$	$O(1)$
Size of Key Update	$O(\log N)$	$O(\log N)$	$O(\log N \log \mathcal{T})$	$O(\log N)$	$O(\log N \log \mathcal{T})$
Size of Ciphertext	$O(\Omega)$	$O(\Omega)$	$O(\log \mathcal{T} \cdot \Omega + \log \mathcal{T} \log \mathcal{T})$	$O(\Omega)$	$O(\log \mathcal{T} \cdot \Omega + \log \mathcal{T} \log \mathcal{T})$
Computation Cost in Key Update	$O(\log N)$	$O(\log N)$	$O(\log N \log \mathcal{T})$	$O(\log N)$	$O(\log N \log \mathcal{T})$
Computation Cost in Encryption	$O(\Omega)$	$O(\Omega)$	$O(\Omega + \log \mathcal{T})$	$O(\Omega)$	$O(\Omega + \log \mathcal{T})$
Computation Cost in Decryption	$O(\Omega)$	$O(\Omega)$	$O(\Omega + \log \mathcal{T})$	$O(\Omega)$	$O(\Omega + \log \mathcal{T})$

the first practical RIBE scheme (refers to type-II indirect revocation). After that, subsequent works [8, 9, 10, 13, 14, 31, 16, 32, 33, 34] in both IBE and ABE settings were proposed to improve the functionality, efficiency and security of revocation mechanisms. Unfortunately, the type-II indirect revocation cannot provide the immediate revocation since the revocation only happens at the beginning of each revocation epoch. To address this problem, the server-aid revocation was proposed in IBE [11] and ABE [15] settings, respectively. However, this revocation method cannot preserve the receivers' privacy since the CSP needs to know the receivers' identities before conducting the partial decryption.

Some revocable schemes [10, 31, 15] considered the security threat called key exposure attack. The non-revoked users required to non-trivially combining their secret keys and related key updates to generate decryption keys, where secret keys are given to users when they join the system and key updates will be published periodically. However, the decryption key is vulnerable due to various security threats in the real application, and the problem of how to keep the secret key secure when the decryption key comprised has been one of important requirements when designing the revocable schemes. Seo and Emura [10] and Watanabe et al. [31] proposed the solution for IBE setting. Unfortunately, there is no formal solution in ABE setting. Some other works relates to revocation and revocable storages, for example revocable predicate encryption [35] and self-update encryption [36, 37], also do not consider the decryption exposure attack.

1.2. Our Contributions

Sahai et al. [14] introduced a novel concept called piecewise key generation and used it to construct RABE schemes. However, this approach by default cannot resist decryption key exposure attack [10]. To address this problem, we introduce a new notion named re-randomizable piecewise key generation.

We then introduce a primitive called re-randomizable attribute-based encryption (RRABE) to remove the relationship among the short-term decryption key, the long-term secret key and the public key-updating material via re-randomization and support ciphertext delegation. We provide a concrete construction for RRABE with re-randomizable piecewise key generation and ciphertext delegation by utilizing the Rouselakis–Waters ABE [25] as the underlying building block.

We propose a strong security model for RABE schemes to capture the decryption key exposure attack by refining the security model in [14], and provide a generic construction of RABE based on RRABE. We prove that the resulting RABE scheme is secure under our refined security model and present a concrete instantiation of the generic construction that achieves secure revocation, decryption key exposure resistance as well as ciphertext delegation.

1.3. Paper Organization

Some preliminaries are introduced in the Section 2. In Section 3, we provide definitions for re-randomizable piecewise key generation, RRABE and RABE, and their security models. We then present the generic transformation for RRABE to RABE and formal security proof in Section 4. We also demonstrate instantiations of RRABE and RABE, and the formal security proof in Section 5. Finally, we summarize our result in Section 6.

2. Preliminaries and Notations

Let $\mathbb{N}$ denote the set of all natural numbers, and for $n \in \mathbb{N}$, we define $[n] := \{1, ..., n\}$. $x \leftarrow y$ denotes that x is output from y if y is a function or an algorithm, or y is assigned to x otherwise. If x and y are strings, then $|x|$ denotes the bit-length of x, and $x\|y$ denotes the concatenation of x and y. For a finite set S, $|S|$ denotes its size and S_i denote the i^{th} value in the set S. If $\mathcal{A}$ is a probabilistic algorithm, then $y \leftarrow \mathcal{A}(x; r)$ denotes that $\mathcal{A}$ computes y as output by taking x as input and using r as randomness, and we just write $y \leftarrow \mathcal{A}(x)$ if we do not need to make the randomness used by $\mathcal{A}$ explicit. Throughout this paper, we use λ to denote a security parameter. A function $\epsilon(\lambda) : \mathbb{N} \rightarrow [0, 1]$ is said to be *negligible* if for all positive polynomials $p(\lambda)$ and all sufficiently large $\lambda \in \mathbb{N}$, we have $\epsilon(\lambda) < 1/p(\lambda)$. Let $\mathcal{M}, \mathcal{I}, \mathcal{T}, \mathcal{P}$ and Ω denote a message space, an identity space, the time bound, policies and an attribute set, respectively. Ω_0 and Ω_1 are two attribute sets derived from Ω, s.t. $\Omega_0 \cup \Omega_1 = \Omega$ and $\Omega_0 \cap \Omega_1 = \varnothing$.

2.1. Bilinear Map

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

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

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

2.2. Assumption

Rouselakis and Waters [25] introduced a q-type assumption on prime order bilinear groups, denoted q-1, which is similar to the decisional parallel bilinear Diffie-Hellman exponent assumption. It is defined via the following game between a challenger and an attacker: Initially the challenger calls the group generation algorithm with input the security parameter, picks a random group element $g \in \mathbb{G}$, and $q + 2$ random exponents $a, s, b_1, b_2, ..., b_q \in \mathbb{Z}_p^{q+2}$. Then the challenger sends to the attacker the group description and all of the following terms:

$$\begin{array}{ll}
g, g^s & \\
g^{a^i}, g^{b_j}, g^{sb_j}, g^{a^i b_j}, g^{a^i/b_j^2} & \forall(i,j) \in [q,q] \\
g^{a^i b/b_{j'}^2} & \forall(i,j,j') \in [2q,q,q] \text{ with } j \neq j' \\
g^{a^i/b_j} & \forall(i,j) \in [2q,q] \text{ with } i \neq q+1 \\
g^{sa^i b_j/b_{j'}}, g^{sa^i b_j/b_{j'}^2} & \forall(i,j,j') \in [q,q,q] \text{ with } j \neq j'
\end{array}$$

The challenger also flips a random coin $b \leftarrow \{0, 1\}$ and if $b = 0$, it gives the attacker the term $e(g, g)^{sa^{q+1}}$. Otherwise, it gives a random term $R \in \mathbb{G}_T$. Finally, the attacker outputs a guess $b' \in \{0, 1\}$.

Definition 1. *We say that the q-1 assumption holds if all polynomial probabilistic time attackers have at most a negligible advantage in λ in the above security game, where the advantage is defined as*

$$\mathbf{Adv} = \Big| \Pr[b = b'] - 1/2 \Big|.$$

2.3. Linear Secret Sharing Scheme

We recall the definition of linear secret sharing scheme (LSSS), as defined in [14]. A LSSS policy is of the type $(\mathbb{M}, \rho)$ where $\mathbb{M}$ is an $n \times l$ matrix over the base field $\mathbb{F}$ and ρ is a map from $[n]$ to Ω. A policy $(\mathbb{M}, \rho)$ satisfies an attribute set $S \subseteq \Omega$ if $1 = (1, 0, ..., 0) \in \mathbb{F}^l$ is contained in $\mathsf{Span}_{\mathbb{F}}(\mathbb{M}_i : \rho(i) \in S)$, where $\mathbb{M}_i$ is the i^{th} row of $\mathbb{M}$.

2.4. Attribute-Based Encryption

ABE schemes are generally divided into two types depending on if the access policy is embedded in keys or ciphertexts. We revisit the definition of CP-ABE, where ciphertexts have an access policies incorporated within while keys are associated with set of attributes. In the rest of paper, unless otherwise specified, let ABE denote CP-ABE.

Definition 2 (ABE). *An ABE scheme Π with the attribute set Ω that supports policies $\mathcal{P}$ with the message space $\mathcal{M}$ involves three types of entities: a KGC, senders and receivers, and consists of five algorithms given below.*

$\Pi.\mathsf{Init}(\lambda) \rightarrow pp$: *The probabilistic initialization algorithm is run by the KGC. It takes a security parameter $\lambda \in \mathbb{N}$ as input, and outputs the public parameter pp.*

$\Pi.\mathsf{Setup}(pp) \rightarrow (pk, msk)$: *The probabilistic setup algorithm is run by the KGC. It takes the public parameter pp as input, and outputs the public key pk and the master secret key msk.*

$\Pi.\mathsf{KeyGen}(msk, S) \to sk_S$*: The probabilistic key generation algorithm is run by the KGC. It takes the master secret key* msk *and an attribute set* $S \subseteq \Omega$ *as input, and outputs the secret key* sk_S*.*

$\Pi.\mathsf{Enc}(pk, m, \mathbb{A}) \to ct_{\mathbb{A}}$*: The probabilistic encryption algorithm is run by senders. It takes the public key* pk*, a message* $m \in \mathcal{M}$ *and an access structure* $\mathbb{A} \in \mathcal{P}$ *as input, and outputs the ciphertext* $ct_{\mathbb{A}}$*.*

$\Pi.\mathsf{Dec}(sk_S, ct_{\mathbb{A}}) \to m/\perp$*: The deterministic decryption algorithm is run by receivers. It takes the secret key* sk_S *and the ciphertext* $ct_{\mathbb{A}}$ *as input, and outputs a message* $m \in \mathcal{M}$ *or a failure symbol* $\perp$*.*

The consistency condition requires that for all $\lambda \in \mathbb{N}$, all pp output by the initialization algorithm, all pk and msk output by setup algorithm, and all $m \in \mathcal{M}$, we then have

$$\Pi.\mathsf{Dec}(sk_S, ct_{\mathbb{A}}) = m$$

with probability 1.

Next, we describe the security of indistinguishable under chosen plaintext attack ($\mathsf{IND\text{-}CPA}$ security) for ABE setting. Throughout this paper, we provide adaptive models in detail, and they are allowed to modify to be selective models by the adversary commits the challenge information in advance.

Definition 3 ($\mathsf{IND\text{-}CPA}$ in ABE). *An ABE scheme* Π *with the attribute set* Ω *that supports policies* $\mathcal{P}$ *with the message space* $\mathcal{M}$ *consists of five algorithms* $\Pi = (\mathsf{Init}, \mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*. For an adversary* $\mathcal{A}$*, we define the following* *experiment:*

$$
\begin{aligned}
&\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\Pi,\mathcal{A}}(\lambda)\\
&\quad pp \leftarrow \Pi.\mathsf{Init}(\lambda);\\
&\quad (pk, msk) \leftarrow \Pi.\mathsf{Setup}(pp);\\
&\quad (m_0, m_1, \mathbb{A}^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\Pi.\mathsf{KeyGen}(\cdot)}}(pp, pk);\\
&\quad b \leftarrow \{0,1\};\\
&\quad ct_{\mathbb{A}*} \leftarrow \Pi.\mathsf{Enc}(pk, \mathbb{A}^*, m_b);\\
&\quad b' \leftarrow \mathcal{A}^{\mathcal{O}_{\Pi.\mathsf{KeyGen}(\cdot)}}(ct_{\mathbb{A}*});\\
&\quad \textit{If } b = b' \textit{ return } 1 \textit{ else return } 0.
\end{aligned}
$$

$\mathcal{O}_{\Pi.\mathsf{KeyGen}}(\cdot)$ *is key generation oracle that allows* $\mathcal{A}$ *to query an attribute set* $S \subseteq \Omega$ *except* $\mathbb{A}^*(S) = 1$, *and it runs* $\Pi.\mathsf{KeyGen}(msk, S)$ *to return the secret key* sk_S.

An ABE scheme is said to be IND-CPA *secure if for any probabilistic polynomial time adversary* $\mathcal{A}$, *the following advantage is negligible:*

$$\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\Pi,\mathcal{A}}(\lambda) = \Big|\Pr[\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\Pi,\mathcal{A}}(\lambda) = 1] - 1/2\Big|.$$

2.5. Rouselakis-Waters Attribute-Based Encryption

We recall the large universe ABE scheme proposed by Rouselakis and Waters [25] since it satisfies prerequisites to transfer the RRABE scheme (refers to Section 3.1 and 2.6), and it has been proved selectively secure if the q-1 assumption holds. The details are given below.

$\Pi.\mathsf{Init}(\lambda) \to pp$: The initialization algorithm takes a security parameter $\lambda \in \mathbb{N}$ as input. It chooses a bilinear group of order p according to the bilinear group parameter generator $(\mathbb{G}, p, g) \leftarrow \mathcal{G}(\lambda)$, and outputs the public parameters $pp = (\mathbb{G}, p, g)$.

$\Pi.\mathsf{Setup}(pp) \to (pk, msk)$: The setup algorithm takes the public parameter pp as input. It picks random terms $g, u, h, w, v \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p$, and outputs the public key pk and the master secret key msk:

$$pp = (u, h, w, v, e(g, g)^{\alpha}), \quad msk = \alpha.$$

$\Pi.\mathsf{KeyGen}(msk, S) \to sk_S$: The key generation algorithm takes the master secret key msk and an attribute set $S = \{A_1, A_2, ..., A_k\} \subseteq \Omega$ as input. It picks $k+1$ random exponents $r, r_1, r_2, ..., r_k \in \mathbb{Z}_p$, and outputs the secret key sk:

$$sk_S = (g^{\alpha} w^r, g^r, \{g^{r_i}, (u^{A_i} h)^{r_i} v^{-r}\}_{i\in[k]}).$$

$\Pi.\mathsf{Enc}(pk, m, \mathbb{A}) \to ct_{\mathbb{A}}$: The encryption algorithm takes the public key pk, a message $m \in \mathcal{M}$ and an access structure $\mathbb{A} \in \mathcal{P}$ encoded in an LSSS policy with $\mathbb{M} \in \mathbb{Z}_p^{n\times l}$ and $\rho : [n] \to \mathbb{Z}_p$ as input. It picks the vector $\vec{y} = (s, y_2, ..., y_l)^{\top} \in \mathbb{Z}_p^{l\times 1}$ and computes the vector $\vec{u} = (u_1, u_2, ..., u_n)^{\top} = \mathbb{M}\vec{y}$. After that, it chooses n random exponents $\mu_1, \mu_2, ..., \mu_n \in \mathbb{Z}_p$ and for $i \in [n]$, it calculates

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

Finally, it outputs the ciphertext $ct_{\mathbb{A}}$:

$$ct_{\mathbb{A}} = (C, C_0, \{C_{1,i}, C_{2,i}, C_{3,i}\}_{i\in[n]}).$$

$\Pi.\mathsf{Dec}(sk_S, ct_{\mathbb{A}}) \to m/\perp$: The decryption algorithm takes the secret key sk_S and the ciphertext $ct_{\mathbb{A}}$ as input. Parse the secret key sk_S and the ciphertext $ct_{\mathbb{A}}$:

$$sk_S = (K_0, K_1, \{K_{2,i}, K_{3,i}\}_{i\in[k]}), \quad ct_{\mathbb{A}} = (C, C_0, \{C_{1,i}, C_{2,i}, C_{3,i}\}_{i\in[n]}).$$

If $A(S) = 0$, it returns a failure symbol $\perp$, otherwise, it calculates the set of rows in $\mathbb{M}$ that provides a share to attributes in S, i.e. $S = \{i : \rho(i) \in S\}$. Then, it computes the constants $\{w_i \in \mathbb{Z}_p\}_{i\in S}$ s.t. $\sum_{i\in S} w_i \mathbb{M}_i = (1, 0, ..., 0)$. Thus, we have $\sum_{i\in S} w_i u_i = s$ and the message hiding component P can be revoked:

$$P = \frac{e(C_0, K_0)}{\prod_{i\in S}(e(C_{1,i}, K_1) e(C_{2,i}, K_{2,i}) e(C_{3,i}, K_{3,i}))^{w_i}}.$$

Finally, it outputs a message $m = C/P$.

2.6. Ciphertext Delegation

We recall the definition of ciphertext delegation in ABE setting, as defined in [14], where the ABE scheme based on LSSS that allows for elementary ciphertext manipulations and ciphertext re-randomization.

Definition 4 (Ciphertext Delegation). *An ABE scheme* Π *with an attribute set* Ω *that supports policies* $\mathcal{P}$ *with a message space* $\mathcal{M}$ *is said to have ciphertext delegation if it has an ciphertext delegation algorithm* $\Pi.\mathsf{CTDelegate}$ *with the following guarantee: For all* $\lambda \in \mathbb{N}$*, all pp output by initialization algorithm, all pk and msk output by setup algorithm, and all* $m \in \mathcal{M}$*, we have*

$$\Pi.\mathsf{CTDelegate}(\Pi.\mathsf{Enc}(pk, m, \mathbb{A}), \mathbb{A}') \equiv \Pi.\mathsf{Enc}(pk, m, \mathbb{A}')$$

if the access policy $\mathbb{A}' \in \mathcal{P}$ *can be derived from the access policy* $\mathbb{A} \in \mathcal{P}$*, where* $\equiv$ *denotes equality in distribution.*

2.7. Tree-Based Revocation Approach

Tree-based data structure is widely used to reduce the cost of generating and transmitting key updates from linear to logarithmic. To revoke a user, the subset-cover algorithm $\mathsf{KUNode}(st, rl, t)$ [4] can be used, where st is the state representing the tree based data structure, rl is the revocation list recording identities of revoked users and t is the time representing the current revocation epoch. When a user wants to join the system, who will be assigned an random identifier $id \in \mathcal{I}$ and an undefined leaf node in st will be labelled this identifier id. The revocation method only requires the user id to store the keys in $\mathsf{Path}(id)$, where $\mathsf{Path}(id)$ denotes nodes from the root node to the leaf node id. More details refer to [4].

3. Formal Definitions and Security Models

3.1. Re-Randomizable Piecewise Key Generation

Piecewise key generation [14] was proposed to support a generic transformation from ABE schemes to RABE schemes. However, this transformation only works in the basic model without considering key exposure attack [10]. To address this problem, we introduce a new notion called re-randomizable piecewise key generation. This transformation requires the underlying ABE to have the following features: (1) the master secret key is embedded in one of secret key components and (2) the secret key has re-randomizable property. The former enables the data user to derive the decryption key by combining the secret key and the key update and the latter allows the decryption key to be re-randomizable to prevent key exposure attack. Some ABE schemes used to construct the RABE schemes in [7, 13, 14] cannot meet the above requirements. The major reason is those ABE schemes split their master secret key based on the LSSS and the decryption key only trivially combines the secret key and the key update together without re-randomization.

Definition 5 (Re-Randomizable Piecewise Key Generation). *An ABE scheme* Π *with the attribute set* Ω *that support policies* $\mathcal{P}$ *with a message space* $\mathcal{M}$ *and an identifier space* $\mathcal{I}$ *is said to have re-randomizable piecewise key generation if the attribute set* Ω *can be split into two attribute sets* Ω_0 *and* Ω_1 *s.t.* $\Omega_0 \cup \Omega_1 = \Omega$ *and* $\Omega_0 \cap \Omega_1 = \emptyset$*, key generation algorithm and decryption algorithm are modified, and has an algorithm* $\Pi.\mathsf{DKGen}$ *with the following guarantees:*

$\Pi.\mathsf{KeyGen}(msk, S_b, b, U_b) \rightarrow sk^{(b)}_{S_b,U_b}$*: The probabilistic key generation algorithm takes the master secret key* msk*, an attribute set* $S_b \subseteq \Omega_b$*, a bit* $b \in \{0,1\}$ *and an identifier* $U_b \in \mathcal{I}$ *as input, and outputs the secret key* $sk^{(b)}_{S_b,U_b}$*.*

$\Pi.\mathsf{DKGen}(sk^{(0)}_{S_0,U_0}, sk^{(1)}_{S_1,U_1}) \rightarrow dk_{S_0,S_1}/\perp$*: The probabilistic decryption key generation algorithm takes two secret keys* $sk^{(0)}_{S_0,U_0}$ *and* $sk^{(1)}_{S_1,U_1}$ *as input, and outputs the decryption key* dk_{S_0,S_1} *or a failure symbol* $\perp$*. Note that decryption key generation is required to be probabilistic for re-randomizing the decryption key.*

$\Pi.\mathsf{Dec}(dk_{S_0,S_1}, ct_{\mathbb{A}}) \rightarrow m/\perp$*: The deterministic decryption algorithm takes the decryption key* dk_{S_0,S_1} *and the ciphertext* $ct_{\mathbb{A}}$ *as input, and outputs a message* $m \in \mathcal{M}$ *or a failure symbol* $\perp$*.*

The consistency condition requires that for all $\lambda \in \mathbb{N}$, all pp output by initialization algorithm, all pk and msk output by setup algorithm, and all $m \in \mathcal{M}$, we have

$$\Pi.\mathsf{Dec}(dk_{S_0,S_1}, ct_{\mathbb{A}}) = m$$

with probability 1.

3.2. Re-Randomizable Attribute-Based Encryption

We give the definition of the syntax and the security model for RRABE. Our RRABE combines building blocks of re-randomizable piecewise key generation and ciphertext delegation, and it will be treated as the basic building block to construct RABE schemes.

Definition 6 (RRABE). *An RRABE scheme Φ with attribute sets Ω_0 and Ω_1 that support policies $\mathcal{P}$ with the message space $\mathcal{M}$ and an identity space $\mathcal{I}$ involves four types of entities: a KGC, senders, receivers and a CSP, and consists of seven algorithms given below.*

$\Phi.\mathsf{Init}(\lambda) \to pp$*: The probabilistic initialization algorithm is run by the KGC. It takes a security parameter $\lambda \in \mathbb{N}$ as input, and outputs the public parameter pp.*

$\Phi.\mathsf{Setup}(pp) \to (pk, msk)$*: The probabilistic setup algorithm is run by the KGC. It takes the public parameter pp as input, and outputs the public key pk and the master secret key msk.*

$\Phi.\mathsf{KeyGen}(msk, S_b, b, U_b) \to sk^{(b)}_{S_b,U_b}$*: The probabilistic key generation algorithm is run by the KGC. It takes the master secret key msk, an attribute set $S_b \subseteq \Omega_b$, a bit $b \in \{0,1\}$ and an identifier $U_b \in \mathcal{I}$ as input, and outputs the secret key $sk^{(b)}_{S_b,U_b}$.*

$\Phi.\mathsf{DKGen}(sk^{(0)}_{S_0,U_0}, sk^{(1)}_{S_1,U_1}) \to dk_{S_0,S_1}/\perp$*: The probabilistic decryption key generation algorithm is run by receivers. It takes two secret keys $sk^{(0)}_{S_0,U_0}$ and $sk^{(1)}_{S_1,U_1}$ as input, and outputs the decryption key dk_{S_0,S_1} or a failure symbol $\perp$.*

$\Phi.\mathsf{Enc}(pk, m, \mathbb{A}) \to ct_{\mathbb{A}}$*: The probabilistic encryption algorithm is run by senders. It takes the public key pk, a message $m \in \mathcal{M}$ and an access structure $\mathbb{A} \in \mathcal{P}$ as input, and outputs the ciphertext $ct_{\mathbb{A}}$.*

$\Phi.\mathsf{CTDelegate}(ct_{\mathbb{A}}, \mathbb{A}') \to ct_{\mathbb{A}'}/\perp$: *The probabilistic ciphertext delegation algorithm is run by the CSP. It takes the ciphertext* $ct_{\mathbb{A}}$ *and an access structure* $\mathbb{A}' \in \mathcal{P}$[1] *as input, and outputs the ciphertext* $ct_{\mathbb{A}'}$ *or a failure symbol* $\perp$.

$\Phi.\mathsf{Dec}(dk_{S_0,S_1}, ct_{\mathbb{A}}) \to m/\perp$: *The deterministic decryption algorithm is run by receivers. It takes the decryption key* dk_{S_0,S_1} *and the ciphertext* $ct_{\mathbb{A}}$ *as input, and outputs a message* $m \in \mathcal{M}$ *if* $\mathbb{A}(S_0) = \mathbb{A}(S_1) = 1$; *otherwise, returns a failure symbol* $\perp$.

Remark. The access structure $\mathbb{A}$ includes two disjointed access structures $\mathbb{A}_0$ and $\mathbb{A}_1$. For simplicity, we omit the subscripts of $\mathbb{A}_0$ and $\mathbb{A}_1$. We use $\mathbb{A}(S_b) = 1$ to represent the attribute set S_b satisfying the access structure $\mathbb{A}_b$.

The consistency condition requires that for all $\lambda \in \mathbb{N}$, all pp output by initialization algorithm, all pk and msk output by setup algorithm, all $m \in \mathcal{M}$, $U \in \mathcal{I}$, and $\mathbb{A}(S_0) = \mathbb{A}(S_1) = 1$, we have

$$\Phi.\mathsf{Dec}(dk_{S_0,S_1}, ct_{\mathbb{A}}) = m$$

with probability 1.

Next, we describe the security definition of $\mathsf{IND\text{-}CPA}$ security for the RRABE scheme. In this model, it captures the security requirement of decryption key exposure resistance by allowing the adversary to query the decryption oracle to obtain the short-term decryption keys.

Definition 7 ($\mathsf{IND\text{-}CPA}$ **in RRABE).** *An RRABE* Φ *with attribute sets* Ω_0 *and* Ω_1 *that supports policies* $\mathcal{P}$ *with the message space* $\mathcal{M}$ *consists of seven algorithms* $\Phi = (\mathsf{Init}, \mathsf{Setup}, \mathsf{KeyGen}, \mathsf{DKGen}, \mathsf{Enc}, \mathsf{CTDelegate}, \mathsf{Dec})$. *For an ad-*

[1] $\mathbb{A}'$ is a more restrict access structure derived from the access structure $\mathbb{A}$. Specifically, for any attribute set S, if $\mathbb{A}'(S) = 1$, then $\mathbb{A}(S) = 1$.

versary $\mathcal{A}$, we define the following experiment:

```
$\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\Phi,\mathcal{A}}(\lambda)$
    $pp \leftarrow \Phi.\mathsf{Init}(\lambda)$;
    $(pk, msk) \leftarrow \Phi.\mathsf{Setup}(pp)$;
    $(m_0, m_1, \mathbb{A}^*) \leftarrow \mathcal{A}^{\mathcal{O}}(pp, pk)$;
    $b \leftarrow \{0,1\}$;
    $ct_{\mathbb{A}^*} \leftarrow \Phi.\mathsf{Enc}(pk, \mathbb{A}^*, m_b)$;
    $b' \leftarrow \mathcal{A}^{\mathcal{O}}(ct_{\mathbb{A}^*})$;
    If $b = b'$ return 1 else return 0.
```

$\mathcal{O}$ represents a set of oracles, $\{\mathcal{O}_{\Phi.\mathsf{KeyGen}}(\cdot,\cdot,\cdot), \mathcal{O}_{\Phi.\mathsf{DKGen}}(\cdot,\cdot,\cdot)\}$, and the details are given in below:

- *$\mathcal{O}_{\Phi.\mathsf{KeyGen}}(\cdot,\cdot,\cdot)$ is key generation oracle that allows $\mathcal{A}$ to query an attribute set $S \subseteq \Omega_b$, a bit $b \in \{0,1\}$ and an identifier $U \in \mathcal{I}$, and it runs $\Phi.\mathsf{KeyGen}(msk, S, b, U)$ to return the secret key $sk^{(b)}_{S_b,U}$.*
- *$\mathcal{O}_{\Phi.\mathsf{DKGen}}(\cdot,\cdot,\cdot)$ is decryption key generation oracle that allows $\mathcal{A}$ to query two attribute sets $S_0 \in \Omega_0$ and $S_1 \in \Omega_1$, and an identifier $U \in \mathcal{I}$, and it runs $\Phi.\mathsf{DKGen}(sk^{(0)}_{S_0,U}, sk^{(1)}_{S_1,U})$ to return the decryption key dk_{S_0,S_1} if secret keys $sk^{(0)}_{S_0,U}$ and $sk^{(1)}_{S_1,U}$ are available. Otherwise, it first runs $\Phi.\mathsf{KeyGen}(msk, S, b, U)$ to obtain the secret key $sk^{(b)}_{S_b,U}$.*

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

1. *If $\mathcal{O}_{\Phi.\mathsf{KeyGen}}(\cdot,\cdot,\cdot)$ was queried on a bit $b \in \{0,1\}$ and an identifier $U \in \mathcal{I}$ with an attribute set $S_b \subseteq \Omega_b$ s.t. $\mathbb{A}^*(S_b) = 1$, then $\mathcal{O}_{\Phi.\mathsf{KeyGen}}(\cdot,\cdot,\cdot)$ cannot be queried for the bit $1-b$ for the identifier U with the attribute set $S_{1-b} \subseteq \Omega_{1-b}$ s.t. $\mathbb{A}^*(S_{1-b}) = 1$.*
2. *$\mathcal{O}_{\Phi.\mathsf{DKGen}}(\cdot,\cdot,\cdot)$ cannot be queried on an identifier $U \in \mathcal{I}$ with two attribute sets $S_0 \subseteq \Omega_0$ and $S_1 \subseteq \Omega_1$ s.t. $\mathbb{A}^*(S_0) = \mathbb{A}^*(S_1) = 1$.*

The RRABE scheme is said to be IND-CPA *secure if for any probabilistic polynomial time adversary $\mathcal{A}$, the following advantage is negligible:*

$$\mathbf{Adv}_{\Phi,\mathcal{A}}^{\mathsf{IND\text{-}CPA}}(\lambda) = \Big|\Pr[\mathbf{Exp}_{\Phi,\mathcal{A}}^{\mathsf{IND\text{-}CPA}}(\lambda) = 1] - 1/2\Big|.$$

3.3. Revocable Attribute-Based Encryption

In this section, we give the formal definition of the syntax of RABE and the related security model. Our definition refines the definitions in the identity-based setting [31] and attribute-based setting [14].

Definition 8 (RABE). *An RABE scheme Ψ with attribute sets Ω_0 and Ω_1 that support policies $\mathcal{P}$ with a message space $\mathcal{M}$, the time bound $\mathcal{T}$ and an identity space $\mathcal{I}$ involves four types of entities: a KGC, senders, receivers and a CSP, and consists of nine algorithms given below.*

$\Psi.\mathsf{Init}(\lambda) \rightarrow pp$: *The probabilistic initialization algorithm is run by the KGC. It takes a security parameter $\lambda \in \mathbb{N}$ as input, and outputs the public parameter pp.*

$\Psi.\mathsf{Setup}(pp, N) \rightarrow (pk, msk, rl, st)$: *The probabilistic setup algorithm is run by the KGC. It takes the public parameter pp and the number of system users $N \in \mathbb{N}$ as input, and outputs the public key pk, the master secret key msk, the revocation list rl and the state st.*

$\Psi.\mathsf{KeyGen}(msk, st, S, id) \rightarrow (sk_{S,id}, st)$: *The probabilistic key generation algorithm is run by the KGC. It takes the master secret key msk, the state st, an attribute set $S \subseteq \Omega_0$ and an identity $id \in \mathcal{I}$ as input, and outputs the secret key $sk_{S,id}$ and the state st.*

$\Psi.\mathsf{KeyUpdate}(msk, rl, st, S_t) \rightarrow ku_{S_t}$: *The probabilistic key update algorithm is run by the KGC. It takes the master secret key msk, the revocation list rl, the state st and a time-related attribute set $S_t \in \Omega_1$ as input, and outputs the key update ku_{S_t}.*

$\Psi.\mathsf{DKGen}(sk_{S,id}, ku_{S_t}) \rightarrow dk_{S,S_t,id}/\perp$: *The probabilistic decryption key generation algorithm is run by receivers. It takes the secret key $sk_{S,id}$ and the key update ku_{S_t} as input, and outputs the decryption key $dk_{S,S_t,id}$ or a failure symbol $\perp$.*

$\Psi.\mathsf{Enc}(pk, m, \mathbb{A}) \to ct_{\mathbb{A}}$: *The probabilistic encryption algorithm is run by senders. It takes the public key pk, a message $m \in \mathcal{M}$ and an access structure $\mathbb{A} \in \mathcal{P}$, which contains both an attribute related policy and a time related policy, as input, and outputs the ciphertext $ct_{\mathbb{A}}$.*

$\Psi.\mathsf{CTUpdate}(ct_{\mathbb{A}}, \mathbb{A}') \to ct_{\mathbb{A}'}/\perp$: *The probabilistic ciphertext update algorithm is run by the KGC. It takes the ciphertext $ct_{\mathbb{A}}$ and an access structure $\mathbb{A}' \in \mathcal{P}$*[2] *as input, and outputs the ciphertext $ct_{\mathbb{A}'}$ or a failure symbol $\perp$.*

$\Phi.\mathsf{Dec}(dk_{S,S_t,id}, ct_{\mathbb{A}}) \to m/\perp$: *The deterministic decryption algorithm is run by receivers. It takes the decryption key $dk_{S,S_t,id}$ and the ciphertext $ct_{\mathbb{A}}$ as input, and outputs a message $m \in \mathcal{M}$ or a failure symbol $\perp$.*

$\Phi.\mathsf{Rev}(rl, id, t) \to rl$: *The deterministic revocation algorithm is run by the KGC. It takes the revocation list rl, an identity $id \in \mathcal{I}$ to be revoked and the time $t \in \mathcal{T}$ as input, and outputs the revocation list rl.*

The consistency condition requires that for all $\lambda \in \mathbb{N}$ and a polynomial N, all pp output by initialization algorithm, all pk and msk output by setup algorithm, all $m \in \mathcal{M}, id \in \mathcal{I}, t \in \mathcal{T}$, all possible valid states st and revocation lists rl, and $\mathbb{A}(S) = \mathbb{A}(S_t) = 1$, we have

$$\Phi.\mathsf{Dec}(dk_{S,S_t,id}, ct_{\mathbb{A}}) = m$$

with probability 1.

Next, we describe the security definition of $\mathsf{IND\text{-}CPA}$ for the RABE scheme. The difference between the following model and the traditional RABE model is that our model provides an additional oracle called decryption key generation oracle, which allows the adversary to query the short-term decryption key. We should note that the previous model for RRABE also has the decryption key generation oracle but it does not consider revocation. In the following model, we will consider the adversary to be either revoked user or non-revoked user

[2]Similar as before, $\mathbb{A}'$ is a more restrict access structure derived from the access structure $\mathbb{A}$.

depending on the challenge access structure and the queries to the revocation oracle. Specifically, the adversary is a revoked user if the long-term secret key satisfying the challenge access structure is revoked before or at the challenge time; otherwise, the adversary is a non-revoked user. The details are described below.

Definition 9 (IND-CPA **in RABE).** *An RABE scheme Ψ with the attribute sets Ω_0 and Ω_1 that support policies $\mathcal{P}$ with a message space $\mathcal{M}$, the time bound $\mathcal{T}$ and an identity space $\mathcal{I}$ consists of nine algorithms* $\Psi = (\mathsf{Init}, \mathsf{Setup}, \mathsf{KeyGen}, \mathsf{KeyUpdate}, \mathsf{DKGen}, \mathsf{Enc}, \mathsf{CTUpdate}, \mathsf{Dec}, \mathsf{Rev})$. *For any adversary $\mathcal{A}$, we define the following experiment:*

```
$\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\Psi,\mathcal{A}}(\lambda, N)$
    $pp \leftarrow \Psi.\mathsf{Init}(\lambda)$;
    $(pk, msk, rl, st) \leftarrow \Psi.\mathsf{Setup}(pp, N)$;
    $(m_0, m_1, \mathbb{A}^*) \leftarrow \mathcal{A}^{\mathcal{O}}(pp, pk)$;
    $b \leftarrow \{0, 1\}$;
    $ct_{\mathbb{A}*} \leftarrow \Psi.\mathsf{Enc}(pk, m_b, \mathbb{A}^*)$;
    $b' \leftarrow \mathcal{A}^{\mathcal{O}}(ct_{\mathbb{A}*})$;
    If $b = b'$ return 1 else return 0.
```

$\mathcal{O}$ is a set of oracle, $\{\mathcal{O}_{\Psi.\mathsf{KeyGen}}(\cdot,\cdot), \mathcal{O}_{\Psi.\mathsf{KeyUpdate}}(\cdot), \mathcal{O}_{\Psi.\mathsf{Rev}}(\cdot,\cdot), \mathcal{O}_{\Psi.\mathsf{DKGen}}(\cdot,\cdot)\}$, and the details are given in below:

- *$\mathcal{O}_{\Psi.\mathsf{KeyGen}}(\cdot,\cdot)$ is key generation oracle that allows $\mathcal{A}$ to query an attribute set $S \subseteq \Omega_0$ and an identity $id \in \mathcal{I}$, and it runs $\Psi.\mathsf{KeyGen}(msk, st, S, id)$ to return the secret key $sk_{S,id}$.*
- *$\mathcal{O}_{\Psi.\mathsf{KeyUpdate}}(\cdot)$ is key update oracle that allows $\mathcal{A}$ to query the time $t \in \mathcal{T}$, and it runs $\Psi.\mathsf{KeyUpdate}(msk, rl, st, S_t)$ to return the key update ku_{S_t}.*
- *$\mathcal{O}_{\Psi.\mathsf{Rev}}(\cdot,\cdot)$ is revocation oracle that allows $\mathcal{A}$ to query an identity $id \in \mathcal{I}$ and the time $t \in \mathcal{T}$, and it runs $\Psi.\mathsf{Rev}(rl, id, t)$ to update the revocation list rl.*

- $\mathcal{O}_{\Psi.\mathsf{DKGen}}(\cdot,\cdot,\cdot)$ *is decryption key generation oracle that allows* $\mathcal{A}$ *to query an attribute set* $S \in \Omega_0$*, the time* $t \in \mathcal{T}$ *and an identity* $id \in \mathcal{I}$*, and it runs* $\Psi.\mathsf{DKGen}(sk_{S,id}, ku_{S_t})$ *to return the decryption key* $dk_{S,S_t,id}$ *if the secret key* $sk_{S,id}$ *and the key update* ku_{S_t} *are available. Otherwise, it first runs* $\Psi.\mathsf{KeyGen}(msk, st, S, id)$ *and* $\Psi.\mathsf{KeyUpdate}(msk, rl, st, S_t)$ *to obtain the secret key* $sk_{S,id}$ *and the key update* ku_{S_t}*.*

$\mathcal{A}$ *is allowed to issue the above oracles with the following restriction:*

1. $\mathcal{O}_{\Psi.\mathsf{KeyUpdate}}(\cdot)$ *and* $\mathcal{O}_{\Psi.\mathsf{Rev}}(\cdot,\cdot)$ *can be queried at the time* t *which is greater than or equal to that of all previous queries.*
2. $\mathcal{O}_{\Psi.\mathsf{Rev}}(\cdot,\cdot)$ *cannot be queried at the time* t *if* $\mathcal{O}_{\Psi.\mathsf{KeyUpdate}}(\cdot)$ *was queried at the time* t*.*
3. *If* $\mathcal{O}_{\Psi.\mathsf{KeyGen}}(\cdot,\cdot)$ *was queried on an identity* $id \in \mathcal{I}$ *with the attribute set* $S \subseteq \Omega_0$ *s.t.* $\mathbb{A}^*(S) = 1$*, then* $\mathcal{O}_{\Psi.\mathsf{Rev}}(\cdot,\cdot)$ *must be queried on this identity* id *at the time* $t \leqslant t^*$*.*
4. $\mathcal{O}_{\Psi.\mathsf{DKGen}}(\cdot,\cdot,\cdot)$ *cannot be queried on any identity* $id \in \mathcal{I}$ *with the attribute set* S *s.t.* $\mathbb{A}^*(S) = 1$ *at the challenge time* t^**.*

An RABE is said to be IND-CPA *secure if for any probabilistic polynomial time adversary* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\Psi,\mathcal{A}}(\lambda, N) = \left|\Pr[\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\Psi,\mathcal{A}}(\lambda, N) = 1] - 1/2\right|.$$

4. Generic Construction of Revocable Attribute-Based Encryption

We provide a framework of the RABE scheme based on RRABE (Section 3.2). Let $\Phi = (\mathsf{Init}, \mathsf{Setup}, \mathsf{KeyGen}, \mathsf{DKGen}, \mathsf{Enc}, \mathsf{CTDelegate}, \mathsf{Dec})$ denote an RRABE scheme with the attribute set Ω_0 and Ω_1 supporting policies $\mathcal{P}$ with a message space $\mathcal{M}$, the time bound $\mathcal{T}$ and an identity space $\mathcal{I}$. The generic construction of the RABE scheme $\Psi = (\mathsf{Init}, \mathsf{Setup}, \mathsf{KeyGen}, \mathsf{KeyUpdate}, \mathsf{DKGen}, \mathsf{Enc}, \mathsf{CTUpdate}, \mathsf{Dec}, \mathsf{Rev})$ is derived as follows:

$\Psi.\mathsf{Init}(\lambda) \to pp$: The initialization algorithm takes a security parameter $\lambda \in \mathbb{N}$ as input, and it runs $\Phi.\mathsf{Init}(\lambda)$ to return the public parameter pp.

$\Psi.\mathsf{Setup}(pp, N) \to (pk, msk, rl, st)$: The setup algorithm takes the public parameter pp and the number of system users $N \in \mathbb{N}$ as input. It runs $\Phi.\mathsf{Setup}(pp)$ to obtain the public key pk and the master secret key msk, and initializes an empty revocation list $rl \leftarrow \varnothing$ and the state $st \leftarrow \mathsf{BT}$, where BT is a binary tree with at least N leaves. Finally, it returns the public key pk, the master secret key msk, the revocation list rl and the state st.

$\Psi.\mathsf{KeyGen}(msk, st, S, id) \to (sk_{S,id}, st)$: The key generation algorithm takes the master secret key msk, the state st, an attribute set $S \subseteq \Omega_0$ and an identity $id \in \mathcal{I}$ as input. For all $\theta \in \mathsf{Path}(id)$, it fetches the state st_{ϵ_θ} if it is available, otherwise, it picks a random state st_{ϵ_θ} in the key domain. It then runs $\Phi.\mathsf{KeyGen}(msk, S, 0, \theta)$ to obtain the secret key $sk_{S,id,\theta}$. Finally, it returns a set of the secret key $sk_{S,id} = \{sk_{S,id,\theta}\}_{\theta \in \mathsf{Path}(id)}$.

$\Psi.\mathsf{KeyUpdate}(msk, rl, st, S_t) \to ku_{S_t}$: The key update algorithm takes the master secret key msk, the revocation list rl, the state st and a time-related attribute set $S_t \subseteq \Omega_1$ as input. For all $\theta \in \mathsf{KUNodes}(st, rl, t)$, it runs $\Phi.\mathsf{KeyGen}(msk, S_t, 1, \theta)$ to obtain the key update $ku_{t,\theta}$. Finally, it returns a set of the key update $ku_{S_t} = \{ku_{S_t,\theta}\}_{\theta \in \mathsf{KUNodes}(st,rl,t)}$.

$\Psi.\mathsf{DKGen}(sk_{S,id}, ku_{S_t}) \to dk_{S,S_t,id}/\perp$: The decryption key generation algorithm takes the secret key $sk_{S,id}$ and the key update $ku_{S,t}$ as input. It returns a failure symbol $\perp$ if $\varnothing = \mathsf{Path}(id) \cap \mathsf{KUNodes}(st, rl, t)$, otherwise, we have the node $\theta = \mathsf{Path}(id) \cap \mathsf{KUNodes}(st, rl, t)$, and it runs $\Phi.\mathsf{DKGen}(sk_{S,id,\theta}, ku_{S_t,\theta})$ to return the decryption key $dk_{S,S_t,id}$.

$\Psi.\mathsf{Enc}(pk, m, \mathbb{A}) \to ct_{\mathbb{A}}$: The encryption algorithm takes the public key pk, a message $m \in \mathcal{M}$ and an access structure $\mathbb{A} \in \mathcal{P}$ as input, and it runs $\Phi.\mathsf{Enc}(pk, m, \mathbb{A})$ to return the ciphertext $ct_{\mathbb{A}}$.

$\Psi.\mathsf{CTUpdate}(ct_{\mathbb{A}}, \mathbb{A}') \to ct_{\mathbb{A}'}/\perp$: The ciphertext update algorithm takes the ciphertext $ct_{\mathbb{A}}$ and an access structure $\mathbb{A}' \in \mathcal{P}$ as input. It runs $\Phi.\mathsf{CTDelegate}(ct_{\mathbb{A}}, \mathbb{A}')$ to return the ciphertext $ct_{\mathbb{A}'}$ or a failure symbol $\perp$.

$\Psi.\mathsf{Dec}(dk_{S,S_t,id}, ct_{\mathbb{A}}) \to m/\perp$: The decryption algorithm takes the decryption key $dk_{S,S_t,id}$ and the ciphertext $ct_{\mathbb{A}}$ as input, and it runs $\Phi.\mathsf{Dec}(dk_{S,S_t,id}, ct_{\mathbb{A}})$

to return a message $m \in \mathcal{M}$ or a failure symbol $\perp$.

$\Psi.\mathsf{Rev}(rl, id, t) \rightarrow rl$: The revocation algorithm takes an identity $id \in \mathcal{I}$ to be revoked and the time $t \in \mathcal{T}$ as input. It updates the revocation list rl as $rl \leftarrow rl \cup (id, t)$.

We prove the security of our RABE scheme described above under the security model in Section 3.3, assuming that the underlying RRABE scheme is secure. The following security proof is in the adaptive model assuming the underlying RRABE is adaptively secure. If the basic scheme is selectively secure, the following security proof needs to be modified by committing the challenge access structure $\mathbb{A}^*$ in advance.

Theorem 1. *If the underlying RRABE scheme is secure, the proposed RABE scheme is secure.*

Proof:. Assume there exist an adversary $\mathcal{A}$ that can break the RABE scheme, we can build the algorithm $\mathcal{B}$ to break the security of the underlying RRABE scheme $\mathcal{C}$.

Setup. $\mathcal{B}$ received the public parameter pp and the public key pk from $\mathcal{C}$, and then initializes an empty revocation list $rl \leftarrow \emptyset$ and the state $st = \mathsf{BT}$, where BT is a binary tree with at least N leaves. Finally, $\mathcal{B}$ sends the public parameter pp and the public key pk to $\mathcal{A}$.

Queries. $\mathcal{A}$ is allowed to make the following queries to $\mathcal{B}$ adaptively.

- $\mathcal{O}_{\Psi.\mathsf{KeyGen}}(S, id)$: $\mathcal{A}$ queries key generation oracle for an attribute set $S \subseteq \Omega_0$ and an identity $id \in \mathcal{I}$ to $\mathcal{B}$. $\mathcal{B}$ randomly picks an unassigned leaf mode and sets this node as id. $\mathcal{B}$ then sends queries $\mathcal{O}_{\Phi.\mathsf{KeyGen}}(S, 0, U)$ for $U \in \mathsf{Path}(id)$ to $\mathcal{C}$, then $\mathcal{C}$ returns a set of secret keys $\{sk_{id,S,U}\}_{U \in \mathsf{Path}(id)}$. Finally, $\mathcal{B}$ returns the secret key $sk_{id,S} = \{sk_{id,S,U}\}_{U \in \mathsf{Path}(id)}$.

- $\mathcal{O}_{\Psi.\mathsf{KeyUpdate}}(t)$: $\mathcal{A}$ queries key update oracle for the time $t \in \mathcal{T}$ to $\mathcal{B}$. $\mathcal{B}$ then sends queries of $\mathcal{O}_{\Phi.\mathsf{KeyGen}}(S_t, 1, U)$ for $U \in \mathsf{KUNodes}(st, rl, t)$ to $\mathcal{C}$. $\mathcal{C}$ returns a set of key updates $\{ku_{S_t,U}\}_{U \in \mathsf{KUNodes}(st,rl,t)}$. Finally, $\mathcal{B}$ returns the key update $ku_{S_t} = \{ku_{S_t,U}\}_{U \in \mathsf{KUNodes}(st,rl,t)}$.

- $\mathcal{O}_{\Psi.\mathsf{Rev}}(id,t)$: $\mathcal{A}$ queries revocation oracle for an identity $id \in \mathcal{I}$ and the time $t \in \mathcal{T}$ to $\mathcal{B}$. $\mathcal{B}$ runs $\Phi.\mathsf{Rev}(rl,id,t)$ to update the revocation list rl.

- $\mathcal{O}_{\Psi.\mathsf{DKGen}}(S,t,id)$: $\mathcal{A}$ queries decryption key generation oracle for an attribute $S \subseteq \Omega_0$, the time $t \in \mathcal{T}$ and an identity $id \in \mathcal{I}$ to $\mathcal{B}$. $\mathcal{B}$ then sends the query of $\mathcal{O}_{\Phi.\mathsf{DKGen}}(S,S_t,U)$ for $U \in \mathsf{Path}(id) \cap \mathsf{KUNodes}(st,rl,t)$ to $\mathcal{C}$, then $\mathcal{C}$ returns the decryption key $dk_{S,S_t,U}$. Finally, $\mathcal{B}$ returns the decryption key $dk_{S,S_t,U}$ to $\mathcal{A}$ as the decryption key $dk_{S,S_t,id}$.

Output. $\mathcal{A}$ outputs two messages m_0 and m_1 s.t. $|m_0| = |m_1|$, and a challenge access structure $\mathbb{A}^*$ (the challenge access structure $\mathbb{A}^*$ is committed in advance in the selective model). $\mathcal{B}$ sends the challenge information $(m_0, m_1, \mathbb{A}^*)$ to $\mathcal{C}$. $\mathcal{C}$ returns the challenge ciphertext $ct_{\mathbb{A}*}$ to $\mathcal{B}$. $\mathcal{B}$ then returns this challenge ciphertext $ct_{\mathbb{A}*}$ to $\mathcal{A}$. $\mathcal{A}$ outputs a bit b' and $\mathcal{B}$ returns b' to $\mathcal{C}$.

There is no abort in above simulation. Therefore, if $\mathcal{A}$ breaks the RABE scheme with advantage ϵ, then we can build an algorithm $\mathcal{B}$ to break the security of the RRABE with advantage ϵ. □

5. Instantiations of Proposed Schemes

In this section, we provide an instantiation of the RRABE scheme and give the formal security proof, and then present an instantiation of RABE and related efficiency analysis compared to other RABE schemes from standard parings in the prime-order groups.

5.1. An Instantiation of Re-Randomizable Attribute-Based Encryption

The RRABE scheme is the basic building block to construct the instantiations of RABE, and the instantiation of RRABE are given below.

$\Phi.\mathsf{Init}(\lambda) \to pp$: The initialization algorithm takes a security parameter $\lambda \in \mathbb{N}$ as input. It chooses a bilinear group of order p according to the bilinear group parameter generator $(\mathbb{G}, p, g) \leftarrow \mathcal{G}(\lambda)$, and outputs the public parameters $pp = (\mathbb{G}, p, g)$.

$\Phi.\mathsf{Setup}(pp) \rightarrow (pk, msk)$: The setup algorithm takes the public parameter pp as input. It picks random terms $u, h, w, v \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p$, then outputs the public key pk and the master secret key msk:

$$pk = (pp, u, h, w, v, e(g, g)^{\alpha}), \quad msk = (pk, \alpha).$$

$\Phi.\mathsf{KeyGen}(msk, S_b, b, U_b) \rightarrow sk^{(b)}_{S_b,U_b}$: The secret key generation algorithm takes the master secret key msk, an attribute set $S_b = \{A_1, A_2, ..., A_k\} \subseteq \Omega_b$, a bit $b \in \{0, 1\}$ and an identifier $U_b \in \mathcal{I}$ as input, where we assume the attribute string has been hashed from the domain $\{0, 1\}^*$ to the attribute universal $\mathbb{Z}_p$ and the $S_0 \in \Omega_0$ and $S_1 \in \Omega_1$ do not have overlap (e.g., $\Omega_0 \cup \Omega_1 = \Omega$ and $\Omega_0 \cap \Omega_1 = \varnothing$). It fetches α_U if it is available, otherwise, it randomly chooses and stores $\alpha_U \in \mathbb{Z}_p$. Then, it picks $k + 1$ random exponents $r_b, r_{b,1}, r_{b,2}, ..., r_{b,k} \in \mathbb{Z}_p^{k+1}$. If $b = 0$, it outputs the secret key $sk^{(0)}_{S_0,U_0}$:

$$sk^{(0)}_{S_0,U_0} = (g^{\alpha_U} w^{r_0}, g^{r_0}, \{g^{r_{0,i}}, (u^{A_i} h)^{r_{0,i}} v^{-r_0}\}_{i\in[k]}).$$

Otherwise, it outputs the secret key $sk^{(1)}_{S_1,U_1}$:

$$sk^{(1)}_{S_1,U_1} = (g^{\alpha-\alpha_U} w^{r_1}, g^{r_1}, \{g^{r_{1,i}}, (u^{A_i} h)^{r_{1,i}} v^{-r_1}\}_{i\in[k]}).$$

$\Phi.\mathsf{DKGen}(sk^{(0)}_{S_0,U_0}, sk^{(1)}_{S_1,U_1}) \rightarrow dk_{S_0,S_1}/\perp$: The decryption key generation algorithm takes two secret keys $sk^{(0)}_{S_0,U_0}$ and $sk^{(1)}_{S_1,U_1}$ as input. Parse two secret keys $sk^{(0)}_{S_0,U_0}$ and $sk^{(1)}_{S_1,U_1}$:

$$sk^{(0)}_{S_0,U_0} = (K^{(0)}_0, K^{(0)}_1, \{K^{(0)}_{2,i}, K^{(0)}_{3,i}\}_{i\in[I]}),$$
$$sk^{(1)}_{S_1,U_1} = (K^{(1)}_0, K^{(1)}_1, \{K^{(1)}_{2,j}, K^{(1)}_{3,j}\}_{j\in[J]}),$$

550 where $|S_0| = I$ and $|S_1| = J$. If $U_0 \neq U_1$, it outputs a failure symbol $\perp$. Otherwise, it picks $I+J+2$ random exponents $r'_0, r'_{0,1}, r'_{0,2}, ..., r'_{0,I}, r'_1, r'_{1,1}, r'_{1,2}, ..., r'_{1,J} \in$

$\mathbb{Z}_p^{I+J+2}$ and calculates:

$$\begin{aligned}
D_0 &= K_0^{(0)} \cdot K_0^{(1)} \cdot w^{r_0'} w^{r_1'} = g^{\alpha} w^{r_0+r_0'+r_1+r_1'},\\
D_1 &= K_1^{(0)} \cdot g^{r_0'} = g^{r_0+r_0'},\\
D_2 &= K_1^{(1)} \cdot g^{r_1'} = g^{r_1+r_1'},\\
D_{3,i} &= K_{2,i}^{(0)} \cdot g^{r_{0,i}'} = g^{r_{0,i}+r_{0,i}'},\\
D_{4,i} &= K_{3,i}^{(0)} \cdot (u^{A_i} h)^{r_{0,i}'} v^{-r_0'} = (u^{A_i} h)^{r_{0,i}+r_{0,i}'} v^{-(r_0+r_0')},\\
D_{5,j} &= K_{2,j}^{(1)} \cdot g^{r_{1,j}'} = g^{r_{1,j}+r_{1,j}'},\\
D_{6,j} &= K_{3,j}^{(1)} \cdot (u^{A_j} h)^{r_{1,j}'} v^{-r_1'} = (u^{A_j} h)^{r_{1,j}+r_{1,j}'} v^{-(r_1+r_1')}.
\end{aligned}$$

Finally, it outputs the decryption key dk_{S_0,S_1}:

$$dk_{S_0,S_1} = (D_0, D_1, D_2, \{D_{3,i}, D_{4,i}\}_{i\in[I]}, \{D_{5,j}, D_{6,j}\}_{j\in[J]}).$$

$\Phi.\mathsf{Enc}(pk, m, \mathbb{A}) \to ct_{\mathbb{A}}$: The encryption algorithm takes the public key pk, a message $m \in \mathcal{M}$ and an access structure $\mathbb{A}$ encoded in an LSSS policy with $\mathbb{M} \in \mathbb{Z}_p^{n\times l}$ and $\rho : [n] \to \mathbb{Z}_p$. It picks the vector $\vec{y} = (s, y_2, ..., y_l)^\top \in \mathbb{Z}_p^{l\times 1}$ randomly and computes the vector $\vec{u} = (u_1, u_2, ..., u_n)^\top = \mathbb{M}\vec{y}$. It then chooses n random exponents $\mu_1, \mu_2, ..., \mu_n \in \mathbb{Z}_p^n$ and for $i \in [n]$, it calculates:

$$C = m \cdot e(g,g)^{\alpha s}, C_0 = g^s, C_{1,i} = w^{u_i} v^{\mu_i}, C_{2,i} = (u^{\rho(i)} h)^{-\mu_i}, C_{3,i} = g^{\mu_i}.$$

Finally, it outputs the ciphertext $ct_{\mathbb{A}}$:

$$ct_{\mathbb{A}} = (C, C_0, \{C_{1,i}, C_{2,i}, C_{3,i}\}_{i\in[n]}).$$

$\Phi.\mathsf{CTDelegate}(ct_{\mathbb{A}}, \mathbb{A}') \to ct_{\mathbb{A}'}/\bot$: The ciphertext delegation algorithm takes the ciphertext $ct_{\mathbb{A}}$ under the access structure $\mathbb{A} = (\mathbb{M}, \rho)$ and an access structure $\mathbb{A}' = (\mathbb{M}', \rho')$, where matrices are $\mathbb{M} \in \mathbb{Z}_p^{n\times l}$, $\mathbb{M}' \in \mathbb{Z}_p^{n'\times l}$ and mapping functions are $\rho : [n] \to \mathbb{Z}_p$, $\rho' : [n'] \to \mathbb{Z}_p$. Parse the ciphertext $ct_{\mathbb{A}}$:

$$ct_{\mathbb{A}} = (C, C_0, \{C_{1,i}, C_{2,i}, C_{3,i}\}_{i\in[n]}).$$

It returns a failure symbol $\bot$ if the access structure $\mathbb{A}'$ cannot be derived from the access structure $\mathbb{A}$, i.e. we have two sets S_I and S_J, s.t. $S_I = \{i' \mid i' =$

$\rho(i), i \in [n]\}$, $S_J = \{j' \mid j' = \rho(j), j \in [n']\}$ and $S_J \nsubseteq S_I$, otherwise, it picks a random vector $\vec{y}' = (s', y'_2, ..., y'_{n'})^\top \in \mathbb{Z}_p^{l \times 1}$ and computes the vector $\vec{u}' = (u'_1, u'_2, ..., u'_{n'})^\top = \mathbb{M}'\vec{y}'$. Then, it chooses n' random exponents $\mu'_1, \mu'_2, ..., \mu'_{n'} \in \mathbb{Z}_p^{n'}$. For all $j \in [n']$, we have $\rho'(j) = \rho(i)$ for some i. After that, it calculates:

$$\begin{aligned}
C' &= C \cdot e(g,g)^{\alpha s'} = m \cdot e(g,g)^{\alpha(s+s')},\\
C'_0 &= C_0 \cdot g^{s'} = g^{s+s'},\\
C'_{1,j} &= C_{1,i} \cdot w^{u'_j} v^{\mu'_j} = w^{u_i+u'_j} v^{\mu_i+\mu'_j},\\
C'_{2,j} &= C_{2,i} \cdot (u^{\rho'(j)}h)^{-\mu'_j} = (u^{\rho'(j)}h)^{-(\mu_i+\mu'_j)},\\
C'_{3,j} &= C_{3,i} \cdot g^{\mu'_j} = g^{\mu_i+\mu'_j}.
\end{aligned}$$

Finally, it outputs the ciphertext $ct_{\mathbb{A}'}$:

$$ct_{\mathbb{A}'} = (C', C'_0, \{C'_{1,j}, C'_{2,j}, C'_{3,j}\}_{j \in [n']}).$$

$\Phi.\mathsf{Dec}(dk_{S_0,S_1}, ct_{\mathbb{A}}) \to m/\perp$: The decryption algorithm takes the decryption key 560 dk_{S_0,S_1} and the ciphertext $ct_{\mathbb{A}}$ as input. Let I denote the size of S_0 and J denote the size of S_1. Parse the decryption key dk_{S_0,S_1} and the ciphertext $ct_{\mathbb{A}}$:

$$\begin{aligned}
dk_{S_0,S_1} &= (D_0, D_1, D_2, \{D_{3,i}, D_{4,i}\}_{i \in [I]}, \{D_{5,j}, D_{6,j}\}_{j \in [J]}),\\
ct_{\mathbb{A}} &= (C, C_0, \{C_{1,i}, C_{2,i}, C_{3,i}\}_{i \in [n]}).
\end{aligned}$$

It calculates the set of rows in $\mathbb{M}$ that provides a share to attributes in S_0 and S_1, i.e. $S_I = \{i : \rho(i) \in S_0\}$ and $S_J = \{j : \rho(j) \in S_1\}$, where S_I and S_J are two sets storing the attribute variables in the attribute domain $\mathbb{Z}_p$ for each attributes in attribute sets S_0 and S_1, respectively. Then, it computes constants $\{w_i \in \mathbb{Z}_p\}_{i \in S_I}$ and $\{w_j \in \mathbb{Z}_p\}_{j \in S_J}$ s.t.

$$\sum_{i \in S_I} w_i \mathbb{M}_i = (1, 0, ..., 0), \quad \sum_{j \in S_J} w_j \mathbb{M}_j = (1, 0, ..., 0).$$

If $\mathbb{A}(S_0) = 0$, it returns a failure symbol $\perp$, otherwise, we have $\sum_{i \in S_I} w_i u_i = s$

and calculates partial message hiding component P_1:

$$\begin{aligned} P_1 &= \prod_{i\in S_I}(e(C_{1,i},D_1)e(C_{2,i},D_{3,i})e(C_{3,i},D_{4,i}))^{w_i} \\ &= \prod_{i\in S_I}(e(w^{u_i}v^{\mu_i},g^{r_0+r'_0})e((u^{\rho(i)}h)^{-\mu_i},g^{r_{0,i}+r'_{0,i}})\cdot \\ &\quad e(g^{\mu_i},(u^{A_i}h)^{r_{0,i}+r'_{0,i}}v^{-(r_0+r'_0)}))^{w_i} \\ &= \prod_{i\in S_I} e(g,w)^{u_iw_i(r_0+r'_0)}e(g,v)^{\mu_iw_i(r_0+r'_0)}\cdot \\ &\quad e(g,u^{\rho(i)}h)^{-\mu w_i(r_{0,i}+r'_{0,i})}e(g,u^{\rho(i)}h)^{\mu w_i(r_{0,i}+r'_{0,i})}\cdot \\ &\quad e(g,v)^{-\mu_iw_i(r_0+r'_0)} \\ &= e(g,w)^{\sum_{i\in S_I}u_iw_i(r_0+r'_0)} \\ &= e(g,w)^{s(r_0+r'_0)}. \end{aligned}$$

If $\mathbb{A}(S_1)=0$, it returns a failure symbol $\perp$, otherwise, we have $\sum_{j\in S_J}w_ju_j=s$
565 and calculate the other partial message hiding component P_2:

$$\begin{aligned} P_2 &= \prod_{j\in S_J}(e(C_{1,j},D_1)e(C_{2,j},D_{5,j})e(C_{3,j},D_{6,j}))^{w_j} \\ &= \prod_{j\in S_J}(e(w^{u_j}v^{\mu_j},g^{r_1+r'_1})e((u^{\rho(j)}h)^{-\mu_j},g^{r_{1,j}+r'_{1,j}})\cdot \\ &\quad e(g^{\mu_j},(u^{A_j}h)^{r_{1,j}+r'_{1,j}}v^{-(r_1+r'_1)}))^{w_j} \\ &= \prod_{j\in S_J} e(g,w)^{u_jw_j(r_1+r'_1)}e(g,v)^{\mu_jw_j(r_1+r'_1)}\cdot \\ &\quad e(g,u^{\rho(j)}h)^{-\mu w_j(r_{1,j}+r'_{1,j})}e(g,u^{\rho(j)}h)^{\mu w_j(r_{1,j}+r'_{1,j})}\cdot \\ &\quad e(g,v)^{-\mu_jw_j(r_1+r'_1)} \\ &= e(g,w)^{\sum_{j\in S_J}u_jw_j(r_1+r'_1)} \\ &= e(g,w)^{s(r_1+r'_1)}. \end{aligned}$$

If $\mathbb{A}(S_0)=\mathbb{A}(S_1)=1$, we have two partial message hiding components P_1 and P_2, then we calculate P_3:

$$\begin{aligned} P_3 &= P_1\cdot P_2 \\ &= e(g,w)^{s(r_0+r'_0)}\cdot e(g,w)^{s(r_1+r'_1)} \\ &= e(g,w)^{s(r_0+r'_0+r_1+r'_1)}. \end{aligned}$$

We recover the message hiding component P_4:

$$\begin{aligned} P_4 &= e(C_0, D_0)/P_3 \\ &= e(g^s, g^\alpha w^{r_0+r_0'+r_1+r_1'})e(g,w)^{-s(r_0+r_0'+r_1+r_1')} \\ &= e(g,g)^{\alpha s} e(g,w)^{s(r_0+r_0'+r_1+r_1')} e(g,w)^{-s(r_0+r_0'+r_1+r_1')} \\ &= e(g,g)^{\alpha s}. \end{aligned}$$

Finally, the message m can be recovered:

$$\begin{aligned} C/P_4 &= m \cdot e(g,g)^{\alpha s}/e(g,g)^{\alpha s} \\ &= m. \end{aligned}$$

5.2. Formal Security Proof of Re-Randomizable Attribute-Based Encryption

We prove the security of our RRABE scheme described above under the selective security model in Section 3.2. Note that the underlying Rouselakis-Waters ABE scheme has been proved selectively secure if the q-1 assumption holds in standard model.

Theorem 2. *If the Rouselakis-Waters scheme is secure, the proposed RRABE scheme is secure.*

Proof:. Assume there exist an adversary $\mathcal{A}$ that can break the RRABE scheme, we can build the algorithm $\mathcal{B}$ to break the security of Rouselakis-Waters scheme $\mathcal{C}$.

Setup. $\mathcal{A}$ sends the challenge access structure $\mathbb{A}^*$ to $\mathcal{B}$. $\mathcal{B}$ then forwards this challenge access structure $\mathbb{A}^*$ to $\mathcal{C}$. $\mathcal{B}$ received the public parameter $pp = (\mathbb{G}, p, g)$ and public key $pk = (u, h, w, v, e(g,g)^\alpha)$ from $\mathcal{C}$. $\mathcal{B}$ then initializes an empty state $st \leftarrow \varnothing$ for recording random keys. Finally, $\mathcal{B}$ returns the public parameter pp and public key pk to $\mathcal{A}$.

Queries. $\mathcal{A}$ is allowed to make the following queries to $\mathcal{B}$ adaptively.

- $\mathcal{O}_{\Phi.\mathsf{KeyGen}}(\cdot,\cdot,\cdot)$: $\mathcal{A}$ queries key generation oracle for an attribute set $S = (A_1, A_2, ..., A_k) \in \Omega_b$, a bit $b \in \{0,1\}$ and an identifier $U \in \mathcal{I}$. It fetches the key α_U from the state st if it is available, otherwise, it chooses $\alpha_U \in \mathbb{Z}_p$ randomly and updates the state st as $st \leftarrow st \cup (U, \alpha_U)$. If $b = 0$, $\mathcal{B}$ returns the secret key $sk^{(0)}_{S,U}$:

$$sk^{(0)}_{S,U} = (g^{\alpha_U} w^r, g^r, \{g^{r_i}, (u^{A_i} h)^{r_i} v^{-r}\}_{i \in [k]}).$$

If $b = 1$, $\mathcal{B}$ queries key generation oracle $\mathcal{O}_{\Pi.\mathsf{KeyGen}(\cdot)}(S)$ to obtain the secret key sk_S:

$$\begin{aligned} sk_S &= (K_0, K_1, \{K_{2,i}, K_{3,i}\}_{i \in [k]}) \\ &= (g^{\alpha} w^r, g^r, \{g^{r_i}, (u^{A_i} h)^{r_i} v^{-r}\} \end{aligned}$$

and it returns the secret key $sk^{(1)}_{S,U}$:

$$sk^{(1)}_{S,U} = (K_0 \cdot g^{-\alpha_U}, K_1, \{K_{2,i}, K_{3,i}\}_{i \in [k]}).$$

- $\mathcal{O}_{\Phi.\mathsf{DKGen}}(\cdot,\cdot,\cdot)$: $\mathcal{A}$ queries decryption key generation oracle for two attribute sets $S_0 \in \Omega_0$ and $S_1 \in \Omega_1$, and an identifier $U \in \mathcal{I}$. If both $\mathcal{O}_{\Phi.\mathsf{KeyGen}}(S_0, 0, U)$ and $\mathcal{O}_{\Phi.\mathsf{KeyGen}}(S_1, 1, U)$ have been queried, $\mathcal{B}$ executes $\Phi.\mathsf{DKGen}(sk^{(0)}_{S_0,U}, S^{(1)}_{S_1,U})$ to return the decryption key dk_{S_0,S_1}, otherwise, it first follows key generation oracle $\mathcal{O}_{\Phi.\mathsf{KeyGen}}(\cdot,\cdot,\cdot)$ to obtain secret keys $sk^{(0)}_{S_0,U}$ and $sk^{(1)}_{S_1,U}$.

Output. $\mathcal{A}$ outputs two message m_0 and m_1 s.t. $|m_0| = |m_1|$ to $\mathcal{B}$. $\mathcal{B}$ sends the challenge information (m_0, m_1) to $\mathcal{C}$. $\mathcal{C}$ returns the challenger ciphertext $ct_{\mathbb{A}*}$ to $\mathcal{B}$. $\mathcal{B}$ then forwards this challenge ciphertext $ct_{\mathbb{A}*}$ to $\mathcal{A}$. $\mathcal{A}$ outputs a bit b' and $\mathcal{B}$ returns b' to $\mathcal{C}$.

If $\mathcal{A}$ breaks the RRABE scheme with advantage ϵ, then we can build an algorithm $\mathcal{B}$ to break the security of Rouselakis-Waters ABE scheme. □

5.3. An Instantiation of Revocable Attribute-Based Encryption

We propose an RABE scheme by ultimating the RRABE scheme in Section 5.1 and the generic transformation in Section 4. The details are given below.

$\Psi.\mathsf{Init}(\lambda) \rightarrow pp$: The initialization algorithm takes a security parameter $\lambda \in \mathbb{N}$ as input. It chooses a bilinear group of order p according to the bilinear group parameter generator $(\mathbb{G}, p, g) \leftarrow \mathcal{G}(\lambda)$, and outputs the public parameter $pp = (\mathbb{G}, p, g)$.

$\Psi.\mathsf{Setup}(pp, N) \rightarrow (pk, msk, rl, st)$: The setup algorithm takes the public parameter pp and the number of system users $N \in \mathbb{N}$ as input. It picks random terms $u, h, w, v \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p$. Then, it initializes a binary tree BT with at least N leaves. It outputs the public key pk, the master secret key msk, the revocation list rl and the state st:

$$pk = (pp, u, h, w, v, e(g, g)^{\alpha}), \quad msk = (pk, \alpha), \quad rl \leftarrow \emptyset, \quad st = \mathsf{BT}.$$

$\Psi.\mathsf{KeyGen}(st, S, id) \rightarrow (sk_{S,id}, st)$: The key generation algorithm takes the state st, an attribute set $S = \{A_1, A_2, ..., A_I\} \subseteq \Omega_0$ and an identity $id \in \mathcal{I}$ as input, where the identity id is a random string in the identity domain $\mathcal{I}$ and it is not the real identity of the user. It randomly chooses an unsigned leaf node in st and stores id in this node. For all $\theta \in \mathsf{Path}(id)$, it fetches α_θ if it is available, otherwise, it randomly picks $\alpha_\theta \in \mathbb{Z}_p$ and updates the state st by storing α_θ in the node θ. It then picks $I + 1$ random exponents $r, r_1, r_2, ..., r_I \in \mathbb{Z}_p^{I+1}$ and calculates the secret key $sk_{S,id,\theta}$:

$$sk_{S,id,\theta} = (g^{\alpha_\theta} w^r, g^r, \{g^{r_i}, (u^{A_i} h)^{r_i} v^{-r}\}_{i \in [I]}).$$

Finally, it outputs the secret key $sk_{S,id}$ and the updated state st:

$$sk_{S,id} = \{sk_{S,id,\theta}\}_{\theta \in \mathsf{Path}(id)}, \quad st \leftarrow st \cup (\theta, \alpha_\theta)_{\theta \in \mathsf{Path}(id)}.$$

$\Psi.\mathsf{KeyUpdate}(msk, rl, st, S_t) \rightarrow ku_{S_t}$: The key update algorithm takes the master secret key msk, the revocation list rl, the state st and an attribute set $S_t \subseteq \Omega_1$ as input, where the time $t = (t_1, t_2, ..., t_J)$ in bit representation with wildcard symbol has been converted to the attribute set $S_t \subseteq \Omega_1$ as in [14] for ciphertext delegation. For all $\theta \in \mathsf{KUNodes}(st, rl, t)$, it fetches α_θ and picks $J+1$ random exponents $R, R_1, R_2, ..., R_J \in \mathbb{Z}_p^{J+1}$, then calculates the key update $ku_{S_t,\theta}$:

$$ku_{S_t,\theta} = (g^{\alpha - \alpha_\theta} w^R, g^R, \{g^{R_j}, (u^{A_{t_j}} h)^{R_j} v^{-R}\}_{j \in [J]}).$$

Finally, it outputs the key update ku_{S_t}:

$$ku_{S_t} = \{ku_{S_t,\theta}\}_{\theta \in \mathsf{KUNodes}(st,rl,t)}.$$

$\Psi.\mathsf{DKGen}(sk_{S,id}, ku_{S_t}) \to dk_{S,S_t,id}/\perp$: The decryption key generation algorithm takes the secret key $sk_{S,id}$ and the key update ku_{S_t} as input. Parse the secret key $sk_{S,id}$ and the key update ku_{S_t}:

$$sk_{S,id} = \{sk_{S,id,\theta}\}_{\theta \in \mathsf{Path}(id)}, ku_{S_t} = \{ku_{S_t,\theta}\}_{\theta \in \mathsf{KUNodes}(st,rl,t)}.$$

If $\mathsf{Path}(id) \cap \mathsf{KUNodes}(st, rl, t) = \varnothing$, it returns a failure symbol $\perp$ which indicates the user was revoked. Otherwise, we have only one node $\theta \in \mathsf{Path}(id) \cap \mathsf{KUNodes}(st, rl, t)$. Parse the secret key $sk_{S,id,\theta}$ and the key update $ku_{S_t,\theta}$:

$$\begin{aligned} sk_{S,id,\theta} &= (K_0^{(0)}, K_1^{(0)}, \{K_{2,i}^{(0)}, K_{3,i}^{(0)}\}_{i\in[I]}), \\ ku_{S_t,\theta} &= (K_0^{(1)}, K_1^{(1)}, \{K_{2,j}^{(1)}, K_{3,j}^{(1)}\}_{j\in[J]}). \end{aligned}$$

It picks $I + J + 2$ random exponents $r', r'_1, r'_2, ..., r'_I, R', R'_1, R'_2, ..., R'_J \in \mathbb{Z}_p^{I+J+2}$ and computes:

$$\begin{aligned} D_0 &= K_0^{(0)} \cdot K_0^{(1)} \cdot w^{r'} w^{R'} = g^{\alpha} w^{r+r'+R+R'}, \\ D_1 &= K_1^{(0)} \cdot g^{r'} = g^{r+r'}, \\ D_2 &= K_1^{(1)} \cdot g^{R'} = g^{R+R'}, \\ D_{3,i} &= K_{2,i}^{(0)} \cdot g^{r'_i} = g^{r_i + r'_i}, \\ D_{4,i} &= K_{3,i}^{(0)} \cdot (u^{A_i} h)^{r'_i} v^{-r'} = (u^{A_i} h)^{r_i + r'_i} v^{-(r+r')}, \\ D_{5,j} &= K_{2,j}^{(1)} \cdot g^{R'_j} = g^{R_j + R'_j}, \\ D_{6,j} &= K_{3,j}^{(1)} \cdot (u^{A_{t_j}} h)^{R'_j} v^{-R'} = (u^{A_{t_j}} h)^{R_j + R'_j} v^{-(R+R')}. \end{aligned}$$

Finally, it outputs the decryption key $dk_{S,S_t,id}$:

$$dk_{S,S_t,id} = (D_0, D_1, D_2, \{D_{3,i}, D_{4,i}\}_{i\in[I]}, \{D_{5,j}, D_{6,j}\}_{j\in[J]}).$$

$\Psi.\mathsf{Enc}(pk, m, \mathbb{A}) \to ct_{\mathbb{A}}$: The encryption algorithm takes the public key pk, the message $m \in \mathcal{M}$ and the access structure $\mathbb{A} \in \mathcal{P}$ encoded in an LSSS policy with

$\mathbb{M} \in \mathbb{Z}_p^{n\times l}$ and $\rho : [n] \to \mathbb{Z}_p$ as input, where the access structure $\mathbb{A}$ can be seen as $\mathbb{A}_{att} \wedge \mathbb{A}_t$, and $\mathbb{A}_{att}$ is the access structure for the attributes and $\mathbb{A}_t$ is the access structure for the time period. It picks a random vector $\vec{y} = (s, y_2, ..., y_l)^\top \in \mathbb{Z}_p^{l\times 1}$ and computes the vector $\vec{u} = (u_1, u_2, ..., u_n)^\top = \mathbb{M}\vec{y}$. It then picks n random exponents $\mu_1, \mu_2, ..., \mu_n \in \mathbb{Z}_p^n$ and calculates:

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

It outputs the ciphertext $ct_{\mathbb{A}}$:

$$ct_{\mathbb{A}} = (C, C_0, \{C_{1,i}, C_{2,i}, C_{3,i}\}_{i\in[n]}).$$

$\Psi.\mathsf{CTUpdate}(ct_{\mathbb{A}}, \mathbb{A}') \to ct_{\mathbb{A}'}/\perp$: The ciphertext update algorithm takes the ciphertext $ct_{\mathbb{A}}$ under the access structure $\mathbb{A} = (\mathbb{M}, \rho) \in \mathcal{P}$ and an access structure $\mathbb{A}' = (\mathbb{M}', \rho') \in \mathcal{P}$ as input, where matrices are $\mathbb{M} \in \mathbb{Z}_p^{n\times l}$, $\mathbb{M}' \in \mathbb{Z}_p^{n'\times l}$ and mapping functions $\rho : [n] \to \mathbb{Z}_p$, $\rho' : [n'] \to \mathbb{Z}_p$. Parse the ciphertext $ct_{\mathbb{A}}$:

$$ct_{\mathbb{A}} = (C, C_0, \{C_{1,i}, C_{2,i}, C_{3,i}\}_{i\in[n]}).$$

If the access structure $\mathbb{A}'$ cannot be derived from the access structure $\mathbb{A}$, it returns a failure symbol $\perp$, otherwise, it picks a random vector $\vec{y}' = (s', y'_2, ..., y'_{n'})^\top \in \mathbb{Z}_p^{l\times 1}$ and computes the vector $\vec{u}' = (u'_1, u'_2, ..., u'_{n'})^\top = \mathbb{M}'\vec{y}'$. After that, it picks n' random exponents $\mu'_1, \mu'_2, ..., \mu'_{n'} \in \mathbb{Z}_p^{n'}$. For all $j \in [n']$, we have $\rho'(j) = \rho(i)$ for some i, then it calculates:

$$\begin{aligned}
C' &= C \cdot e(g, g)^{\alpha s'} = m \cdot e(g, g)^{\alpha(s+s')}, \\
C'_0 &= C_0 \cdot g^{s'} = g^{s+s'}, \\
C'_{1,j} &= C_{1,i} \cdot w^{u'_j} v^{\mu'_j} = w^{u_i+u'_j} v^{\mu_i+\mu'_j}, \\
C'_{2,j} &= C_{2,i} \cdot (u^{\rho'(j)} h)^{-\mu'_j} = (u^{\rho'(j)} h)^{-(\mu_i+\mu'_j)}, \\
C'_{3,j} &= C_{3,i} \cdot g^{\mu'_j} = g^{\mu_i+\mu'_j}.
\end{aligned}$$

It outputs the updated ciphertext $ct_{\mathbb{A}'}$:

$$ct_{\mathbb{A}'} = (C', C'_0, \{C'_{1,j}, C'_{2,j}, C'_{3,j}\}_{j\in[n']}).$$

$\Psi.\mathsf{Dec}(dk_{S,S_t,id}, ct_{\mathbb{A}}) \to m/\perp$: The decryption algorithm takes the decryption key $dk_{S,S_t,id}$ and ciphertext $ct_{\mathbb{A}}$ as input. Parse the decryption key $dk_{S,S_t,id}$ and the ciphertext $ct_{\mathbb{A}}$:

$$\begin{aligned} dk_{S,S_t,id} &= (D_0, D_1, D_2, \{D_{3,i}, D_{4,i}\}_{i\in[I]}, \{D_{5,j}, D_{6,j}\}_{j\in[J]}), \\ ct_{\mathbb{A}} &= (C, C_0, \{C_{1,i}, C_{2,i}, C_{3,i}\}_{i\in[n]}). \end{aligned}$$

It calculates the set of rows in $\mathbb{M}$ that provides a share to attributes in S_I and S_J, i.e. $S_I = \{i : \rho(i) \in S\}$ and $S_J = \{j : \rho(j) \in S_t\}$. Then, it computes constants $\{w_i \in \mathbb{Z}_p\}_{i\in S_I}$ and $\{w_j \in \mathbb{Z}_p\}_{j\in S_J}$ s.t.

$$\sum_{i\in S_I} w_i \mathbb{M}_i = (1, 0, ..., 0), \sum_{j\in S_J} w_j \mathbb{M}_j = (1, 0, ..., 0).$$

If $\mathbb{A}(S) = 0$, it returns a failure symbol $\perp$, otherwise, we have $\sum_{i\in S_I} w_i u_i = s$ and calculate partial message hiding component P_1:

$$\begin{aligned} P_1 &= \prod_{i\in S_I} (e(C_{1,i}, D_1)e(C_{2,i}, D_{3,i})e(C_{3,i}, D_{4,i}))^{w_i} \\ &= e(g, w)^{s(r_0+r_0')}. \end{aligned}$$

If $\mathbb{A}(S_t) = 0$, it returns a failure symbol $\perp$, otherwise, we have $\sum_{j\in S_J} w_j u_j = s$ and calculate the other partial message hiding component P_2:

$$\begin{aligned} P_2 &= \prod_{j\in S_J} (e(C_{1,j}, D_1)e(C_{2,j}, D_{5,j})e(C_{3,j}, D_{6,j}))^{w_j} \\ &= e(g, w)^{s(r_1+r_1')}. \end{aligned}$$

If $\mathbb{A}(S) = \mathbb{A}(S_t) = 1$, we have two partial message hiding components P_1 and P_2, then we have P_3:

$$P_3 = P_1 \cdot P_2 = e(g, w)^{s(r_0+r_0'+r_1+r_1')}.$$

We calculate the message hiding component P_4:

$$P_4 = e(C_0, D_0)/P_3 = e(g, g)^{\alpha s}.$$

Finally, the message m can be recovered:

$$C/P_4 = m.$$

$\Psi.\mathsf{Rev}(rl, id, t) \rightarrow rl$: The revocation algorithm takes the revocation list rl, an identity $id \in \mathcal{I}$ to be revoked and the time $t \in \mathcal{T}$, and updates revocation list rl:

$$rl \leftarrow rl \cup (id, t).$$

5.4. Efficiency Analysis

Our RABE scheme has the constant size public key $O(1)$ and we deploy the subset-cover method [4] for revoking corrupted users to reduce the cost from linear to logarithmic. Compared to previous RABE schemes [7, 13, 14, 15], our RABE scheme is proved secure in the strong security model, and achieves additional features in terms of security and functionality simultaneously, e.g., decryption key exposure resistance and ciphertext delegation, and the efficiency of our proposed scheme is comparable with previous RABE schemes.

We provide the experimental results comparing the efficient ciphertext-policy RABE [14] and our proposed RABE scheme. We have implemented the above schemes in Java using jPBC library with the Type A elliptic curve and the symmetric paring setting. In particularly, we used the parameters from "a.properties" in jPBC library, which produces an elliptic curve bilinear group with 160-bit group order, 512-bit base field and embedded degree 2. Hence, p is a 160-bit prime number, and elements in $\mathbb{G}$ and $\mathbb{G}_T$ are 512-bit and 1024-bit, respectively. The software implementation was performed on a PC running 64-bit Windows 10 with Dual 2.8GHz Intel(R) Core(TM) i7-7700HQ CPU and 16GB memory.

The experimental result of the computational time of the key generation algorithm is in Figure 3. We limit the size of the attribute universe $|\Omega| = 30$ in this experiment result and the rest of simulations, and to evaluate the scalability of the scheme, the number of system users is from $2^4 = 16$ to $2^{12} = 4096$. Note that the computational time of the key generation algorithm only relates to the number of system users.

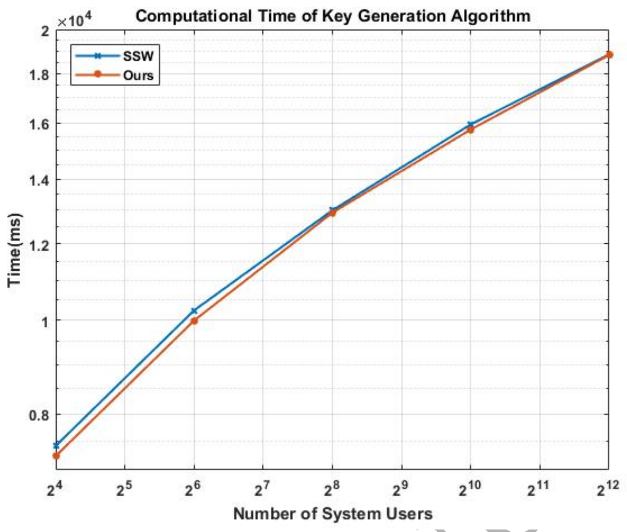

Figure 3: Computational Time of Key Generation Algorithm ($|\Omega| = 30, N = 2^4$ to 2^{12})

The comparisons of the computational time in the key update algorithm are given in Figure 4 and 5. In Figure 4, we consider a lightweight system. In Figure 5, we then consider a large systems with $N = 2^{12} = 4096$ and $R = 2^7 = 128$ in the time from 2^{10} to 2^{50}.

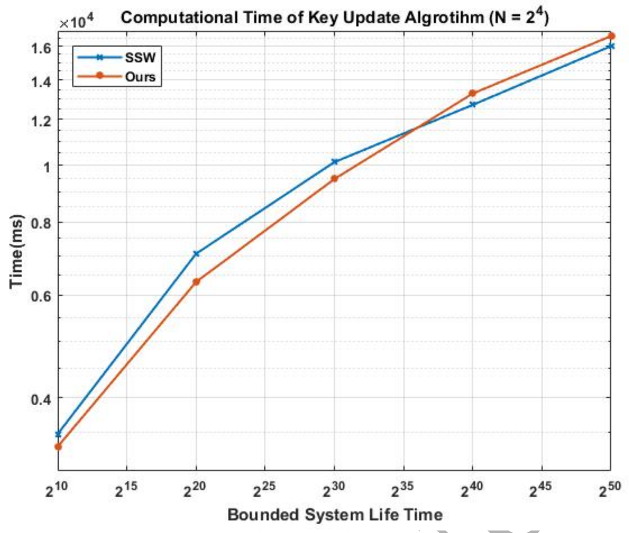

Figure 4: Computational Time of Key Update Algorithm ($|\Omega| = 30, N = 2^4,\ R = 2^2$)

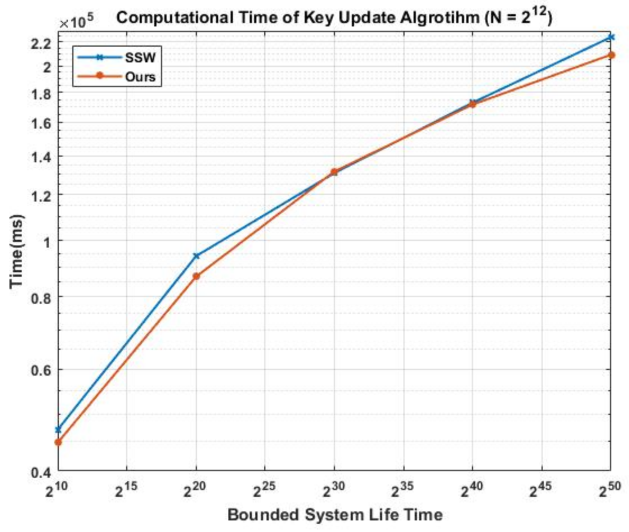

Figure 5: Computational Time of Key Update Algorithm ($|\Omega| = 30, N = 2^{12},\ R = 2^7$)

Figure 6 shows the computational time of the encryption algorithm with different bounded system life time from 2^{10} to 2^{50}. Note that the computational time of encryption algorithm only relates to the access policy and the bounded system life time. 655

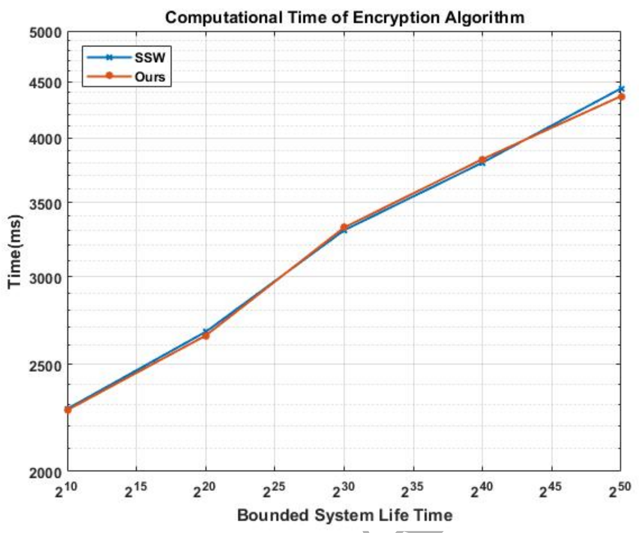

Figure 6: Computational Time of Encryption Algorithm ($|\Omega| = 30$)

From Figure 3 to 6, we can see that the experimental results are consistent with what we excepted in Table 1. Therefore, our construction has comparable performance to the state-of-the-art ciphertext-policy RABE scheme supporting ciphertext delegation, and our scheme also provides decryption key exposure resistance. 660

6. Conclusion

In this paper, we propose a novel notion called re-randomizable piecewise key generation and use it to construct re-randomizable attribute-based encryption (RRABE), which is a new cryptographic primitive to construct revocable ABE with decryption key exposure resistance. We also define the formal security models against decryption key exposure attack, and provide instantiations of RRABE and RABE schemes under our proposed models. Our proposed 665

RABE scheme allows not only efficient revocation but also capture many functionalities, e.g., decryption key exposure resistance and ciphertext delegation simultaneously. Moreover, the efficiency of our scheme is not worse than previous RABE schemes without decryption key exposure resistance and ciphertext delegation.

Acknowledgment

We thank the anonymous reviewers for their invaluable comments and suggestions.

References

[1] S. Micali, Efficient certificate revocation, Tech. rep. (1996).

[2] W. Aiello, S. Lodha, R. Ostrovsky, Fast digital identity revocation (extended abstract), in: CRYPTO, 1998, pp. 137–152.

[3] K. Nissim, M. Naor, Certificate revocation and certificate update, in: USENIX, 1998.

[4] D. Naor, M. Naor, J. Lotspiech, Revocation and tracing schemes for stateless receivers, in: CRYPTO, 2001, pp. 41–62.

[5] D. Boneh, X. Ding, G. Tsudik, C. Wong, A method for fast revocation of public key certificates and security capabilities, in: USENIX, 2001.

[6] D. Boneh, M. K. Franklin, Identity-based encryption from the weil pairing, in: CRYPTO, 2001, pp. 213–229.

[7] A. Boldyreva, V. Goyal, V. Kumar, Identity-based encryption with efficient revocation, in: ACM CCS, 2008, pp. 417–426.

[8] B. Libert, D. Vergnaud, Adaptive-id secure revocable identity-based encryption, in: CT-RSA, 2009, pp. 1–15.

[9] L. Su, H. W. Lim, S. Ling, H. Wang, Revocable IBE systems with almost constant-size key update, in: Pairing, 2013, pp. 168–185.

[10] J. H. Seo, K. Emura, Revocable identity-based encryption revisited: Security model and construction, in: PKC, 2013, pp. 216–234.

[11] B. Qin, R. H. Deng, Y. Li, S. Liu, Server-aided revocable identity-based encryption, in: ESORICS, 2015, pp. 286–304.

[12] M. Pirretti, P. Traynor, P. D. McDaniel, B. Waters, Secure attribute-based systems, in: ACM CCS, 2006, pp. 99–112.

[13] N. Attrapadung, H. Imai, Attribute-based encryption supporting direct/indirect revocation modes, in: IMA, 2009, pp. 278–300.

[14] A. Sahai, H. Seyalioglu, B. Waters, Dynamic credentials and ciphertext delegation for attribute-based encryption, in: CRYPTO, 2012, pp. 199–217.

[15] H. Cui, R. H. Deng, Y. Li, B. Qin, Server-aided revocable attribute-based encryption, in: ESORICS, 2016, pp. 570–587.

[16] S. Xu, G. Yang, Y. Mu, R. H. Deng, Secure fine-grained access control and data sharing for dynamic groups in the cloud, IEEE Trans. Information Forensics and Security 13 (8) (2018) 2101–2113.

[17] A. Sahai, B. Waters, Fuzzy identity-based encryption, in: EUROCRYPT, 2005, pp. 457–473.

[18] X. Liu, R. H. Deng, K. R. Choo, J. Weng, An efficient privacy-preserving outsourced calculation toolkit with multiple keys, IEEE Trans. Information Forensics and Security 11 (11) (2016) 2401–2414.

[19] J. Lee, M. Tehranipoor, C. Patel, J. Plusquellic, Securing designs against scan-based side-channel attacks, IEEE Trans. Dependable Sec. Comput. 4 (4) (2007) 325–336.

[20] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in: ACM CCS, 2006, pp. 89–98.

[21] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: IEEE Symposium on Security and Privacy, 2007, pp. 321–334.

[22] N. Attrapadung, J. Herranz, F. Laguillaumie, B. Libert, E. de Panafieu, C. Ràfols, Attribute-based encryption schemes with constant-size ciphertexts, Theor. Comput. Sci. 422 (2012) 15–38.

[23] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, B. Waters, Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption, in: EUROCRYPT, 2010, pp. 62–91.

[24] B. Waters, Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions, in: CRYPTO, 2009, pp. 619–636.

[25] Y. Rouselakis, B. Waters, Practical constructions and new proof methods for large universe attribute-based encryption, in: A. Sadeghi, V. D. Gligor, M. Yung (Eds.), ACM SIGSAC, ACM, 2013, pp. 463–474.

[26] A. Balu, K. Kuppusamy, An expressive and provably secure ciphertext-policy attribute-based encryption, Inf. Sci. 276 (2014) 354–362.

[27] Y. Shi, Q. Zheng, J. Liu, Z. Han, Directly revocable key-policy attribute-based encryption with verifiable ciphertext delegation, Inf. Sci. 295 (2015) 221–231.

[28] H. Cui, G. Wang, R. H. Deng, B. Qin, Escrow free attribute-based signature with self-revealability, Inf. Sci. 367-368 (2016) 660–672.

[29] J. Cui, H. Zhou, H. Zhong, Y. Xu, AKSER: attribute-based keyword search with efficient revocation in cloud computing, Inf. Sci. 423 (2018) 343–352.

[30] W. Susilo, G. Yang, F. Guo, Q. Huang, Constant-size ciphertexts in threshold attribute-based encryption without dummy attributes, Inf. Sci. 429 (2018) 349–360.

[31] Y. Watanabe, K. Emura, J. H. Seo, New revocable IBE in prime-order groups: Adaptively secure, decryption key exposure resistant, and with short public parameters, in: CT-RSA, 2017, pp. 432–449.

[32] J. Ning, Z. Cao, X. Dong, K. Liang, L. Wei, K. K. R. Choo, Cryptcloud+: Secure and expressive data access control for cloud storage, IEEE Transactions on Services Computing PP (99) (2018) 1–1.

[33] J. Li, W. Yao, Y. Zhang, H. Qian, J. Han, Flexible and fine-grained attribute-based data storage in cloud computing, IEEE Transactions on Services Computing PP (99) (2017) 1–1.

[34] J. Ning, Z. Cao, X. Dong, K. Liang, H. Ma, L. Wei, Auditable σ-time outsourced attribute-based encryption for access control in cloud computing, IEEE Trans. Information Forensics and Security 13 (1) (2018) 94–105.

[35] J. M. G. Nieto, M. Manulis, D. Sun, Fully private revocable predicate encryption, in: ACISP, 2012, pp. 350–363.

[36] K. Lee, S. G. Choi, D. H. Lee, J. H. Park, M. Yung, Self-updatable encryption: Time constrained access control with hidden attributes and better efficiency, in: ASIACRYPT, 2013, pp. 235–254.

[37] K. Lee, Self-updatable encryption with short public parameters and its extensions, Des. Codes Cryptography 79 (1) (2016) 121–161.