1-2020

Server-aided revocable attribute-based encryption for cloud computing services

Hui CUI
Murdoch University

Tsz Hon YUEN
University of Hong Kong

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Guilin WANG
Huawei International Pte Ltd

Published in Concurrency and Computation: Practice and Experience, 2020, advance online. https://doi.org/10.1002/cpe.5680

Server-Aided Revocable Attribute-Based Encryption for Cloud Computing Services*

Hui Cui[a,*], Tsz Hon Yuen[b], Robert H. Deng[c], Guilin Wang[d]

[a] *Discipline of Information Technology, Mathematics and Statistics, Murdoch University, Perth, Australia*
[b] *Department of Computer Science, University of Hong Kong, Hong Kong, China*
[c] *Singapore Management University, Singapore, Singapore*
[d] *Huawei International Pte Ltd, Singapore, Singapore*

Abstract

Attribute-based encryption (ABE) has been regarded as a promising solution in cloud computing services to enable scalable access control without compromising the security. Despite of the advantages, efficient user revocation has been a challenge in ABE. One suggestion for user revocation is using the binary tree in the key generation phase of an ABE scheme, which enables a trusted key generation center (KGC) to periodically distributes the key update information to all non-revoked users over a public channel. This revocation approach reduces the size of key updates from linear to logarithmic in the number of users. But it requires each user to keep a private key of the logarithmic size, and asks each non-revoked user to periodically update his/her decryption key for each new time period. To further optimize user revocation in ABE, a server-aided revocable ABE (SR-ABE) scheme has been proposed, in which almost all workloads of users incurred by the user revocation are outsourced to an untrusted server, and each user only needs to store a private key of the constant size. In addition, SR-ABE does not require any secure channel for the key transmission, and a user only needs to perform a small amount of calculations to decrypt a ciphertext. In this paper, we revisit the notion of SR-ABE, and present a generic construction of SR-ABE, which can transform a revocable ABE (RABE) scheme to an SR-ABE scheme. In addition, we give an instantiation of SR-ABE by applying the generic construction on a concrete RABE scheme, and implement an instantiation of SR-ABE and an RABE scheme to evaluate the performance of SR-ABE.

Keywords: Cloud computing, access control, revocation, outsourced computation, attribute-based encryption.

*Note that this paper is an extended version of its original publication in ESORICS 2016.

*Corresponding author
Email address: hui.cui@murdoch.edu.au (Hui Cui)

1. Introduction

On behalf of the protection of data security, attribute-based encryption (ABE) [1] is envisioned as one of the promising solutions for scalable and fine-grained access control over encrypted data in cloud computing services. It has two forms [2], of which one is called ciphertext-policy ABE (CP-ABE) and the other one is called key-policy ABE (KP-ABE). In a CP-ABE scheme, the data is encrypted under an access structure over a set of attributes by a the data owner, each user is issued a private attribute-key reflecting his/her attributes from a trusted key generation center (KGC), and a user is able to decrypt a ciphertext if the attributes ascribed to his/her private attribute-key satisfy the access structure associated with this ciphertext. In a KP-ABE scheme, the situation is reversed that each user's private attribute-key is associated with an access structure and each ciphertext is ascribed to a set of attributes, and a user is able to decrypt a ciphertext if the access structure of his/her private attribute-key is satisfied by the attributes associated with this ciphertext.

Efficient user revocation has been a very important and challenging problem in ABE, since an ABE based system may have a large number of users, and a user's private key might be compromised. Boldyreva, Goyal and Kumar [3] put forth an efficient revocation method by combining the fuzzy identity-based encryption (IBE) scheme [1] (which is essentially a KP-ABE scheme) with the binary tree [4], where the KGC issues a long-term private key to each user and publicly broadcasts key updates at the beginning of each time period, and only non-revoked users are able to generate decryption keys from their long-term private keys and the key updates to decrypt the newly created ciphertexts (i.e., ciphertexts generated after the revocation happens). The revocable ABE (RABE) schemes (e.g., [3, 5, 6, 7, 8, 9, 10]) following the Boldyreva-Goyal-Kumar approach mitigate the KGC's communication overhead incurred in the key update process, but they fail to reduce the workloads of users since every user needs to keep a private key of the logarithmic size (i.e., $O(R\log(\frac{N}{R}))$ with N being the number of all users and R is the number of revoked users) and all non-revoked users need to periodically update their decryption keys to decrypt newly encrypted messages. With this observation in mind, Cui et al. [11] put forward a concrete server-aided revocable attribute-based encryption (SR-ABE) scheme to achieve the efficient and secure user revocation in ABE, where almost all workloads of users are outsourced to an untrusted server who manages users' public keys and key updates sent by the KGC periodically, and each user keeps just one private key of the constant size (i.e., O(1)). Later, Qin et al. [8, 9] extended the results in [11] by adding the security to the private key stored by the user, and built a concrete construction to satisfy such an additional security. In this paper, based on the results in [11], we consider SR-ABE one step further by giving a generic construction of SR-ABE which can

convert a revocable ABE (RABE, including revocable CP-ABE and revocable KP-ABE) scheme to an SR-ABE (server-aided revocable CP-ABE or server-aided revocable CP-ABE) scheme.

1.1. Our Contributions

In brevity, SR-ABE is a protocol running among a KGC, data owners, users and an untrusted server[1]. Take a KP-ABE scheme as an example, in which the KGC publishes the public parameter, and keeps the master private key. When a user, say Alice, wants to join an SR-ABE based cloud computing service, she generates a public and private user-key pair. Alice keeps the private user-key to herself, and sends the public user-key to the KGC who will create a public attribute-key for Alice based on her public user-key and an access policy over attributes. The KGC sends the public attribute-key for Alice to the server. Also, the KGC periodically generates key updates for all non-revoked users and publicly transmits them to the server. When outsourcing a message to the cloud, a data owner encrypts the message over an attribute set under the current time period, and uploads the resulting ciphertext to the cloud. To decrypt a ciphertext, Alice forwards the ciphertext to the server. If Alice is not revoked and her access structure can be satisfied by the attribute set ascribed to the ciphertext, the server is able to generate a transformation key (from this user's public attribute-key and the key update information) to partially decrypt the ciphertext. This partially decrypted ciphertext can be fully decrypted by Alice using her private user-key.

The key challenge in constructing an SR-ABE scheme is how to prevent collusion attacks among users and the server such that any revoked user, given a partially decrypted ciphertext by the untrusted server, cannot obtain the plaintext from the partially decrypted ciphertext, even though his/her set of attributes satisfies the access structure of the ciphertext in CP-ABE (or access structure can be satisfied by the attribute set of the ciphertext in KP-ABE). Our solution is to subtly embed the public user-key into the public attribute-key generated by the KGC such that the untrusted server can still partially decrypt ciphertexts for non-revoked users, but every partially decrypted ciphertext is bound with a public user-key, which can only be decrypted by the user possessing the corresponding private user-key. We found that there exists a generic way to build SR-ABE schemes, assuming that there exists a secure revocable ABE (RABE) scheme [3, 5, 10] (please refer to Section 4.1 for the formal definition about RABE) and a secure public-key encryption (PKE) scheme. Specifically, each user

[1]The server is untrusted in the sense that it honestly follows the protocol, but it does not hold any secret information (i.e., it may collude with users), and all operations done by the server can be performed by anyone, including users (i.e., any dishonest behavior from the server can be easily detected).

has a public and private user-key pair created via the key generation algorithm of a PKE scheme. To generate a public attribute-key for a user, the KGC first generates a private attribute-key by running the key extraction algorithm of the RABE scheme, and then runs the encryption algorithm of the PKE scheme to encrypt the private attribute-key and the resulting ciphertext of the private attribute-key is the corresponding pubic attribute-key. The ciphertext is created as that in the RABE scheme, but since the public attribute-key is not purely attribute-based, the decryption result of a ciphertext is a partially decrypted ciphertext rather than the real encrypted data, which can be recovered by running the decryption algorithm of the PKE scheme using the private user-key.

It is worth noticing that the generic SR-ABE construction has the following merits.

- Almost all workloads of users caused by the revocation procedure are outsourced to an untrusted server.
- Each user only needs to store a private key of constant size, which is independent to the size of the attribute set or the access structure.
- There is no need of secure channels for the distribution of private keys, since they are generated by each user himself/herself.
- Regardless of the complexity of an access structure, each user only needs to perform a small amount of computations to decrypt a ciphertext.

1.2. Related Work

Revocation Mechanisms. Since Boneh and Franklin [12] suggested to periodically renew users' private keys to achieve user revocation, user revocation has been intensively investigated in identity-based encryption (IBE) [12]. As the Boneh-Franklin revocation approach requires all users to regularly contact the KGC over secure channels, regardless of whether their keys have expired or not, the size of key updates is linear to the number of all non-revoked users (i.e., $O(N-R)$ with N being the number of all users and R being the number of revoked users). Hanaoka et al. [13] gave a solution such that users can regularly renew their private keys without interacting with the KGC (the KGC publicly posts the key update information), but the approach requires each user to possess a tamper-resistant hardware device, which is inconvenient in practice. Boldyreva, Goyal and Kumar [3] proposed an efficient revocation mechanism to remove the secure channels required during the key update phase and reduce the size of key updates from linear to logarithmic (i.e., $O(R\log(\frac{N}{R}))$), but it still needs all non-revoked users to periodically update their private keys for decryption. There are also revocable schemes using

a third party to address the user revocation, in which a semi-trusted[2] (e.g., [14, 15, 16, 17, 18, 19]) or an untrusted (e.g., [20, 11]) third party holds the shares of all users' private keys and help them with the ciphertext decryption. Once a user is revoked, the third party stops decrypting (or is disallowed to decrypt) for this user.

Revocable ABE. There have been two kinds of mechanisms to achieve user revocation in ABE [5, 7]: direct revocation and indirect revocation. In the direct revocation, a data owner directly specifies a revocation list when generating the ciphertext (e.g., [5, 21, 22, 23]). A variant of it is to give the direct revocation capability to a semi-trusted server who shares the decryption ability with users, and will terminate decryption operations for revoked users, e.g., the revocable ABE scheme in [24]. In the indirect revocation, the KGC indirectly prevents revoked users from accessing the ciphertexts through a key update process (e.g., [3, 5, 6, 7, 8, 9, 10]).

Direct revocation can be immediately accomplished without key updates. But it is impurely attribute-based, because it requires the data owner to keep a current revocation list to store each user's status, while data owners in the attribute-based setting should create ciphertexts based solely on attributes. In this paper, we focus on the indirect revocation in ABE.

1.3. Organization

The remainder of this paper is organized as follows. In Section 2, we briefly review the notions and definitions relevant to this paper. In Section 3, we give a generic construction of SR-ABE as well as its security, put forth an instantiation by applying the generic construction into the concrete scheme, and compare SR-ABE with previous solutions on RABE. In Section 4, we discuss several applications of SR-ABE in the real world. We conclude the paper in Section 5.

2. Preliminaries

In this section, we review some basic cryptographic definitions that are to be used in this paper.

2.1. Bilinear Pairings

Prime-Order Bilinear Pairing. Let G be a group of a prime order p generated from g. We define $\hat{e} : G \times G \rightarrow G_1$ to be a bilinear pairing if it is bilinear such that $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$ for all $g \in G$, and a, $b \in Z_p$, and non-degenerate such that $\hat{e}(g, g) \neq 1$. [12].

[2] In this paper, unless otherwise specified, "semi-trusted" means that the third party is disallowed to collude with users.

If the group operation in G is efficiently computable and there exists a group G_1 and an efficiently computable bilinear map $\hat{e} : G \times G \to G_1$ as above, then G is a prime-order bilinear group.

Composite-Order Bilinear Pairing. Let G be a group of an order $\mathcal{N} = p_1p_2p_3$, where p_1, p_2, p_3 are three distinctive primes. We define $\hat{e} : G \times G \to G_1$ to be a bilinear pairing if it is bilinear such that $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$ for all g, $h \in G$, and a, $b \in Z_{\mathcal{N}}$, and non-degenerate such that there exists $g \in G$ to make $\hat{e}(g, g)$ have an order $\mathcal{N}$ in G_1 [25].

If the group operation in G is efficiently computable and there exists a group G_1 and an efficiently computable bilinear map $\hat{e} : G \times G \to G_1$ as above. Let G_{p_1}, G_{p_2} and G_{p_3} denote the subgroups of orders p_1, p_2 and p_3 in G, respectively. Note that when $h_i \in G_{p_i}$ and $h_j \in G_{p_j}$ for $i \neq j$, $\hat{e}(h_i, h_j)$ is the identity element "1" in G_1, then G is a composite-order bilinear group.

Decisional Linear Assumption [26]. The Decisional Linear problem is that for any probabilistic polynomial-time (PPT) algorithm, given g_1, g_2, g_3, ${g_1}^a$, ${g_2}^b$, it is difficult to distinguish (g_1, g_2, g_3, ${g_1}^a$, ${g_2}^b$, ${g_3}^{a+b}$) from (g_1, g_2, g_3, ${g_1}^a$, ${g_2}^b$, Z), where g_1, g_2, g_3, $Z \in G$, a, $b \in Z_p$ are chosen independently and uniformly at random.

2.2. Access Structures and Linear Secret Sharing Schemes

We review notions about access structures and linear secret sharing schemes in [25, 27] as follows.

Definition 1. (Access Structures). Let $\{P_1, ..., P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,...,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \subseteq \mathbb{A}$. A monotone access structure is a monotone collection $\mathbb{A}$ of non-empty subsets of $\{P_1, ..., P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,...,P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called authorized sets, and the sets not in $\mathbb{A}$ are called unauthorized sets.

Definition 2. (Linear Secret Sharing Schemes). Let P be a set of parties. Let $\mathbb{M}$ be a matrix of size $l \times n$. Let $\rho : \{1, ..., l\} \to P$ be a function that maps a row to a party for labeling. A linear secret sharing scheme (LSSS) 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, and there exists a share-generating matrix $\mathbb{M}$ with l rows and n columns for Π. For $i = 1, ..., l$, the x-th row of the matrix $\mathbb{M}$ is labelled by a party $\rho(i)$, where $\rho : \{1, ..., l\} \to P$ is a function that maps a row to a party for labelling. Considering that the column vector $\overrightarrow{v} = (\mu, r_2, ..., r_n)$, where $\mu \in Z_p$ is the secret to be shared and $r_2, ..., r_n \in Z_p$ are randomly chosen, then $\mathbb{M}\overrightarrow{v}$ is the vector of l shares of the secret μ according to the LSSS Π. The share $(\mathbb{M}\overrightarrow{v})_i$ belongs to the party $\rho(i)$.

It has been noted in [25] that every LSSS also enjoys the linear reconstruction property. Suppose that Π is an LSSS for an access structure $\mathbb{A}$. Let $\mathbf{A}$ be an authorized set, and define $I \subseteq \{1, ..., l\}$ as

$I = \{i|\rho(i) \in \mathbf{A}\}$. Then the vector (1, 0, ..., 0) is in the span of rows of matrix $\mathbb{M}$ indexed by I, and there exist constants $\{w_i \in Z_p\}_{i\in I}$ such that, for any valid shares $\{v_i\}$ of a secret μ according to Π, we have $\sum_{i\in I} w_i v_i = \mu$. These constants $\{w_i\}$ can be found in polynomial time with respect to the size of the share-generating matrix $\mathbb{M}$ [28].

Notice that for the construction in composite-order groups, the LSSS matrix over $Z_{\mathcal{N}}$ rather than Z_p will be employed, where $\mathcal{N} = p_1p_2p_3$ is a product of three distinct primes. Different from the definition above over Z_p, a set of attributes $\mathbf{A}$ is authorized if the rows of the access matrix $\mathbb{M}$ labeled by attributes in $\mathbf{A}$ have the vector (1, 0, ..., 0) in their span modulo $\mathcal{N}$.

2.3. Terminologies for Binary Tree Based Revocation

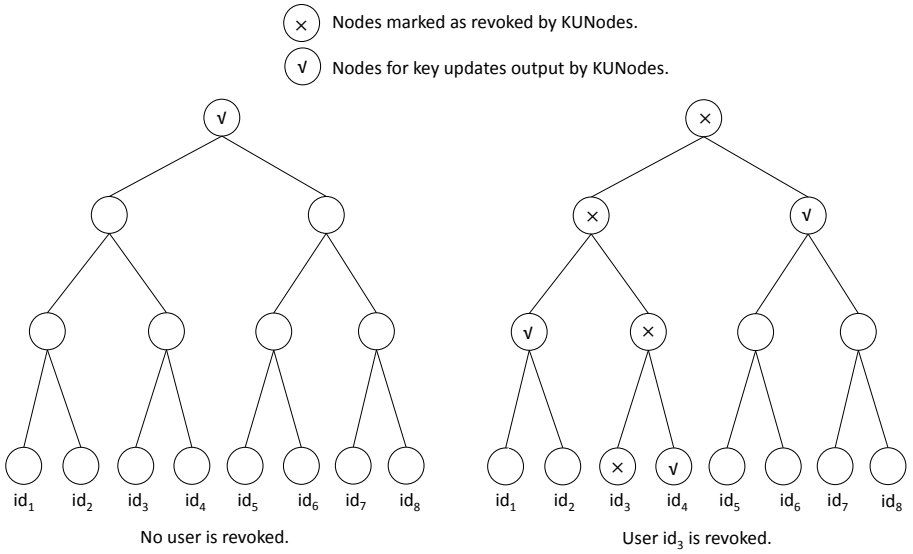


Figure 1: The KUNodes algorithm.

We review the definition of the binary tree described in [3, 20]. Denote BT as a binary tree with N leaves corresponding to N users. Let **root** be the root node of the tree BT. If θ is a leaf node, then Path(θ) denotes the set of nodes on the path from θ to **root**, which includes both θ and **root**. If θ is a non-leaf node, then θ_l, θ_r denote left and right child of θ. Assume that nodes in the tree are uniquely encoded as strings, and the tree is defined by all of its node descriptions. The algorithm KUNodes is used to compute the minimal set of nodes for which key update needs to be published so that only the non-revoked users at a time period t are able to decrypt the ciphertexts. This algorithm takes a binary tree BT, a revocation list rl and a time period t as the input, and outputs a set of nodes which is the minimal set of nodes in BT such that none of the nodes in rl with corresponding time period

before or at t (users revoked at or before t) have any ancestor (or, themselves) in the set, and all other leaf nodes (corresponding to non-revoked users) have exactly one ancestor (or, themselves) in the set. We give a pictorial depiction on how the KUNodes algorithm works in Fig. 1, where it firstly marks all the ancestors of the revoked nodes as revoked, and then it outputs all the non-revoked children of revoked nodes. Below is a formal definition of the KUNodes algorithm.

$\text{KUNodes}(\text{BT}, rl, t)$

$X, Y \leftarrow \emptyset.$

$\forall\ (\theta_i, t_i) \in rl,\ \text{if } t_i \leq t,\ \text{then add Path}(\theta_i) \text{ to } X.$

$\forall\ x \in X,\ \text{if } x_l \notin X,\ \text{then add } x_l \text{ to } Y;$

$\text{if } x_r \notin X, \text{then add } x_r \text{ to } Y.$

$\text{If } Y = \emptyset,\ \text{then add } \mathbf{root} \text{ to } Y.$

Return Y.

2.4. Public-Key Encryption

A public-key encryption (PKE) scheme is composed of the following algorithms: a setup algorithm PKE.Setup$(1^\lambda) \rightarrow par_{pke}$, which takes a security parameter λ as the input, and outputs the public parameter par_{pke}; a key generation algorithm PKE.KeyGen$(par_{pke}) \rightarrow (pk, sk)$ which takes the public parameter par_{pke} as the input, and outputs a public and private key pair (pk, sk); an encryption algorithm PKE.Encrypt$(par_{pke}, pk, M) \rightarrow$ CT which takes the public parameter par_{pke}, the public key pk and an a message $M \in \mathbf{M}$ (here $\mathbf{M}$ is the message space) as the input, and outputs a ciphertext CT; and a decryption algorithm PKE.Decrypt$(par_{pke}, \text{CT}, sk) \rightarrow M/\perp$ which takes the public parameter par_{pke}, a ciphertext CT and the private key sk as the input, and outputs a message M or a failure symbol $\perp$. The correctness of a PKE scheme requires that for any security parameter λ and any message $M \in \mathbf{M}$, if all parties follow the described algorithms as above, then we have Decrypt$(par_{pke}, \text{CT}, sk) = M$.

A PKE scheme is indistinguishable under chosen plaintext attacks (IND-CPA secure) if for any probabilistic polynomial time (PPT) adversary $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\}$, the advantage function $\mathbf{Adv}^{\text{ind-cpa}}_{\text{PKE},\mathcal{A}}(\lambda)$

=

$$\Pr\left[b'=b \;\middle|\; \begin{array}{l} par_{pke} \leftarrow \text{PKE.Setup}(1^{\lambda}),\ b \leftarrow \{0,1\} \\ (pk, sk) \leftarrow \text{PKE.KeyGen}(par_{pke}) \\ (M_0, M_1, state) \leftarrow \mathcal{A}_1(par_{pke}, pk) \\ \text{CT}^* \leftarrow \text{PKE.Encrypt}(par_{pke}, pk, M_b) \\ b' \leftarrow \mathcal{A}_2(par_{pke}, pk, M_0, M_1, state, \text{CT}^*) \end{array}\right] - 1/2$$

is negligible in the security parameter λ, where $|M_0| = |M_1|$.

Linear Encryption [26]. Let G be a group of a prime order p with a generator g_3. Assume that $((g_1, g_2, g_3), (\beta_1, \beta_2))$ is the public and private key pair (pk, sk), where $\beta_1, \beta_2 \in Z_p$, and $g_1 = {g_3}^{\frac{1}{\beta_1}}$, $g_2 = {g_3}^{\frac{1}{\beta_2}}$. The encryption of a message M is $\text{CT} = (C_1, C_2, C_3) = ({g_1}^{r_1}, {g_2}^{r_2}, {g_3}^{r_1+r_2}) \cdot M)$, where $r_1, r_2 \in Z_p$. The message M can be recovered with the private key (β_1, β_2) by computing $M = \frac{C_3}{{C_1}^{\beta_1}{C_2}^{\beta_2}}$. The Linear Encryption scheme is a natural variant of the ElGamal encryption scheme [29], which is CPA-secure under the Linear assumption and has desirable properties that it is hard if decisional Diffie-Hellman (DDH) is hard, but, at least in generic groups, remains hard even if DDH is easy [26].

2.5. System Overview

The system architecture of SR-ABE under a cloud computing scenario is shown in Fig. 2, which involves four types of entities: the KGC, data owners, users and an untrusted server which could be operated by anyone including the cloud. The KGC possesses the master private key, and publishes the corresponding public parameter. To join the cloud storage system, a user generates a public and private user-key pair by himself/herself, and keeps the private user-key. Then, this user sends the public user-key (along with a proof showing that he/she knows the corresponding private user-key) to the KGC. Assume that the SR-ABE scheme is a server-aided revocable KP-ABE scheme. Based on the public user-key and the access structure over attributes of this user, the KGC creates a public attribute-key for this user using the master private key and sends it to the untrusted server (here the KGC authenticates the users, and only eligible users will be issued a public attribute-key). In addition, the KGC periodically generates key updates for all non-revoked users and publicly transmits them to the untrusted server. Before storing a message to the cloud, a data owner encrypts the message over an attribute set under the current time period with the public parameter. Thereafter, the data owner uploads the resulting ciphertext to the cloud. To decrypt a ciphertext, a user forwards the ciphertext to the untrusted server. If the user is not revoked and his/her access structure can be satisfied by the

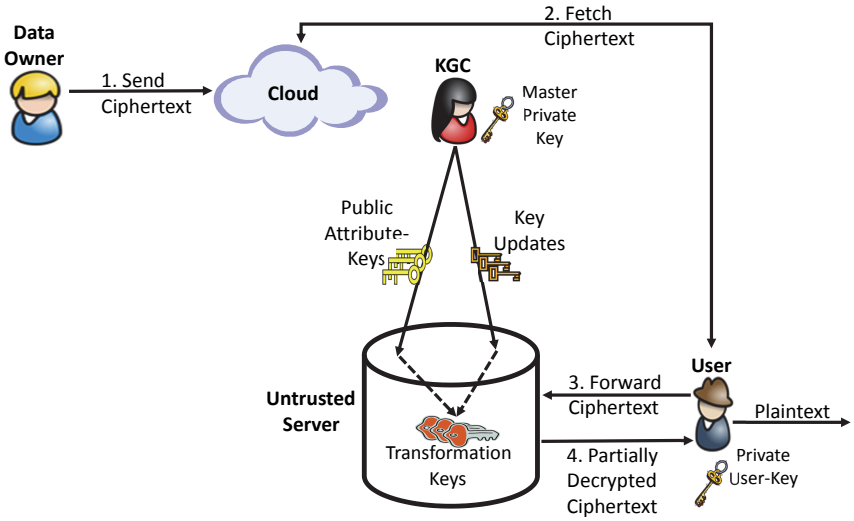


Figure 2: System architecture of SR-ABE.

attribute set ascribed to the ciphertext, the untrusted server can generate a transformation key from this user's public attribute-key and the key update information, with which the server can partially decrypt the ciphertext. A user can fully decrypt a partially decrypted ciphertext using his/her private user-key.

2.6. Framework

An SR-ABE scheme (with ABE being KP-ABE) consists of nine algorithms given below. Assume that the server in Fig. 2 keeps a list of tuples (identity, access structure, public attribute-key), i.e., $(id, (\mathbb{M}, \rho), pk_{id}^{(\mathbb{M},\rho)})$. Let $\mathbf{T}$ denote the space of the time periods, and $\mathbf{M}$ be the message space.

- Setup$(1^\lambda) \rightarrow (par, msk, rl, st)$. Taking a security parameter λ as the input, this algorithm outputs the public parameter par, the master private key msk, an initially empty revocation list rl and a state st. This algorithm is run by the KGC.

- UserKG$(par, id) \rightarrow (sk_{id}, pk_{id})$. Taking the public parameter par and an identity (in the identity space) as the input, this algorithm outputs a public and private user-key pair (sk_{id}, pk_{id}). This algorithm is run by each user.

- PubKG(par, msk, id, pk_{id}, $(\mathbb{M}, \rho)$, st) $\rightarrow$ ($pk_{id}^{(\mathbb{M},\rho)}$, st). Taking the public parameter par, the master private key msk, an identity id (in the identity space) with a public user-key pk_{id} and an access structure $(\mathbb{M}, \rho)$ (in the space of the access structures) and a state st as the input, this algorithm outputs a public attribute-key $pk_{id}^{(\mathbb{M},\rho)}$ for user id with an access structure $(\mathbb{M}, \rho)$ and an updated state st. This algorithm is run by the KGC, and ($pk_{id}^{(\mathbb{M},\rho)}$, st) is sent to the untrusted server.

- TKeyUp(par, msk, t, rl, st) $\rightarrow$ (tku_t, st). Taking the public parameter par, the master private key msk, a time period $t \in \mathbf{T}$, a revocation list rl and a state st as the input, this algorithm outputs the transformation key update information tku_t and an updated state st. This algorithm is run by the KGC, and (tku_t, st) is sent to the server.

- TranKG(par, id, $pk_{id}^{(\mathbb{M},\rho)}$, tku_t) $\rightarrow$ $tk_{id,t}^{(\mathbb{M},\rho)}$. Taking the public parameter par, an identity id (in the identity space) with the corresponding public attribute-key $pk_{id}^{(\mathbb{M},\rho)}$ and the transformation key update information tku_t as the input, this algorithm outputs a transformation key $tk_{id,t}^{(\mathbb{M},\rho)}$ for user id in time period t. This algorithm is run by the server.

- Encrypt(par, $\mathbf{A}$, t, M) $\rightarrow$ $\mathrm{CT}_{\mathbf{A},t}$. Taking the public parameter par, an attribute set $\mathbf{A}$ (in the attribute space), a time period $t \in \mathbf{T}$ and a message $M \in \mathbf{M}$ as the input, this algorithm outputs a ciphertext $\mathrm{CT}_{\mathbf{A},t}$. This algorithm is run by each data owner, and $\mathrm{CT}_{\mathbf{A},t}$ will be stored in the cloud.

- Transform(par, id, $(\mathbb{M}, \rho)$, $tk_{id,t}^{(\mathbb{M},\rho)}$, $\mathrm{CT}_{\mathbf{A},t}$) $\rightarrow$ $\mathrm{CT}_{id}/\perp$. Taking the public parameter par, an identity id (in the identity space) with an access structure $(\mathbb{M}, \rho)$ (in the space of access structures) and the corresponding transformation key $tk_{id,t}^{(\mathbb{M},\rho)}$ and a ciphertext $\mathrm{CT}_{\mathbf{A},t}$ as the input, this algorithm outputs either a partially decrypted ciphertext CT_{id} when the transformation key $tk_{id,t}^{(\mathbb{M},\rho)}$ matches the ciphertext $\mathrm{CT}_{\mathbf{A},t}$ or $\perp$ indicating the failure of the transformation. This algorithm is run by the server. After the partial decryption, CT_{id} is sent to the user id.

- Decrypt(par, id, sk_{id}, CT_{id}) $\rightarrow$ $M/\perp$. Taking the public parameter par, an identity id (in the identity space) with a private user-key sk_{id} and a transformed ciphertext CT_{id} as the input, this algorithm outputs a message M or a failure symbol $\perp$. This algorithm is run by a user id.

- Revoke(id, t, rl, st) $\rightarrow$ (rl, st). Taking an identity id to be revoked, a time period t, a revocation list rl and a state st, this algorithm outputs an updated revocation list rl and an updated state st. This algorithm is run by the KGC.

The correctness of the above SR-ABE scheme requires that for any security parameter λ, any id (in the identity space) with an authorized attribute set $\mathbf{A}$ (in the attribute space) of any access structure $(\mathbb{M}, \rho)$ (in the space of access structures), any time period $t \in \mathbf{T}$ and any message $m \in \mathbf{M}$, $(par, msk, rl, st) \leftarrow \text{Setup}(1^\lambda)$, $(sk_{id}, pk_{id}) \leftarrow \text{UserKG}(par, id)$, $(pk_{id}^{(\mathbb{M},\rho)}, st) \leftarrow \text{PubKG}(par, msk, id, pk_{id}, (\mathbb{M}, \rho), st)$, $(tku_t, st) \leftarrow \text{TKeyUp}(par, msk, t, rl, st)$, $tk_{id,t}^{(\mathbb{M},\rho)} \leftarrow \text{TranKG}(par, id, pk_{id}^{(\mathbb{M},\rho)}, tku_t)$, $\text{CT}_{\mathbf{A},t} \leftarrow \text{Encrypt}(par, \mathbf{A}, t, M)$, then we have $\text{CT}_{id} \leftarrow \text{Transform}(par, id, \mathbb{O}, tk_{id,t}^{(\mathbb{M},\rho)}, \text{CT}_{\mathbf{A},t})$ and $M \leftarrow \text{Decrypt}(par, id, sk_{id}, \text{CT}_{id})$ if $\text{Revoke}(id, t, rl, st)$ has never been run.

Notice that the above algorithms (as well as the corresponding correctness definition) also work for server-aided revocable CP-ABE except that the server will keep a list of tuples (identity, attribute set, public attribute-key), i.e., $(id, \mathbf{A}, pk_{id}^{\mathbf{A}})$, and the relevant access structure $(\mathbb{M}, \rho)$ in the PubKG, TranKG and Transform algorithms will be replaced by the attribute set $\mathbf{A}$, and the input attribute set $\mathbf{A}$ in the Encrypt algorithm will be replaced by the access structure $(\mathbb{M}, \rho)$.

2.7. Security Model

Considering that ABE is KP-ABE for SR-ABE, in the security model, all possible adversarial capabilities are taken into consideration. The adversary is able to learn the private user-keys and public attribute-keys of users with access structures of its choice. The adversary should not be able to learn any partial information about the message encrypted for the challenge attribute set. In addition, the adversary is given access to the periodic key updates, transformation keys for different time periods and can revoke users of its choice. The adversary should also not be able to learn any partial information about the messages encrypted for any revoked user whose access structure can be satisfied by the challenge attribute set when the encryption is done after the time of revocation.

We describe a security definition of indistinguishability under chosen plaintext attacks, i.e., IND-CPA security, for the SR-ABE framework in the previous section between an adversary algorithm $\mathcal{A}$ and a challenger algorithm $\mathcal{B}$ to meet the security requirements.

- Setup. Algorithm $\mathcal{B}$ runs the setup algorithm, and gives the public parameter par to algorithm $\mathcal{A}$, and keeps the master private key msk, an initially empty revocation list rl and a state st.
- Phase 1. Algorithm $\mathcal{A}$ adaptively issues a sequence of the following queries to algorithm $\mathcal{B}$.
  - Private-User-Key oracle. Algorithm $\mathcal{A}$ issues a private user-key query on an identity id. Algorithm $\mathcal{B}$ returns sk_{id} by running $\text{UserKG}(par, id)$. Note that once algorithm $\mathcal{B}$ runs $\text{UserKG}(par, id)$, it adds (id, pk_{id}, sk_{id}) to a list so that the same (sk_{id}, pk_{id}) is used for all queries on id.

- Public-Attribute-Key oracle. Algorithm $\mathcal{A}$ issues a public attribute-key query on an identity id and an access structure $(\mathbb{M}, \rho)$. Algorithm $\mathcal{B}$ returns $pk_{id}^{(\mathbb{M},\rho)}$ by running UserKG(par, id) (if id has not been issued to the Private-User-Key oracle), PubKG(par, msk, id, pk_{id}, $(\mathbb{M}, \rho)$, st).
- Transformation-Key-Update oracle. Algorithm $\mathcal{A}$ issues a key update query on a time period t. Algorithm $\mathcal{B}$ runs TKeyUp(par, msk, t, rl, st) and returns tku_t.
- Transformation-Key oracle. Algorithm $\mathcal{A}$ issues a transformation key query on a time period t and an identity id with an access structure $(\mathbb{M}, \rho)$. Algorithm $\mathcal{B}$ returns $tk_{id,t}^{(\mathbb{M},\rho)}$ by running UserKG(par, id) (if id has not been issued to the Private-User-Key oracle), PubKG(par, msk, id, pk_{id}, $(\mathbb{M}, \rho)$, st), TKeyUp(par, msk, t, rl, st), TranKG(par, id, $pk_{id}^{(\mathbb{M},\rho)}$, tku_t). Note that this oracle cannot be queried on a time period t before a transformation key update oracle has been queried on t.
- Revocation oracle. Algorithm $\mathcal{A}$ issues a revocation query on an identity id and a time period t. Algorithm $\mathcal{B}$ runs Revoke(id, t, rl, st) and outputs an updated revocation list rl and an updated status st.

- Challenge. Algorithm $\mathcal{A}$ outputs two messages M_0^*, M_1^* of the same size, an attribute set $\mathbf{A}^*$ and a time period t^* satisfying the following constraints.
  1. Case 1: if (1) an identity id^* has been queried to the Private-User-Key oracle, and (2) the attribute set $\mathbf{A}^*$ of this identity id^* satisfies a query on (id^*, $(\mathbb{M}^*, \rho^*)$) issued to the Public-Attribute-Key oracle, then (1) the revocation oracle must be queried on (id^*, t) on $t = t^*$ or any t occurs before t^*, and (2) the Transformation-Key oracle cannot be queried on (id^*, t^*).
  2. Case 2: if an identity id^* whose access structure $(\mathbb{M}^*, \rho^*)$ can be satisfied by the challenge attribute set $\mathbf{A}^*$ is not revoked at or before t^*, then id^* should not be previously queried to the Private-User-Key oracle.

  Algorithm $\mathcal{B}$ randomly chooses $\gamma \in \{0, 1\}$, and forwards the challenge ciphertext $\mathrm{CT}_{\mathbf{A}^*,t^*}$ to algorithm $\mathcal{A}$ by running Encrypt(par, $\mathbf{A}^*$, t^*, M_γ^*).

- Phase 2. Algorithm $\mathcal{A}$ continues issuing queries to algorithm $\mathcal{B}$ as in Phase 1, following the restrictions defined in the Challenge phase.

- Guess. Algorithm $\mathcal{A}$ makes a guess γ' for γ, and it wins the game if $\gamma' = \gamma$.

The advantage of algorithm $\mathcal{A}$ in this game is defined as $\Pr[\gamma = \gamma'] - 1/2$. An SR-ABE scheme as described in Section 2.6 is IND-CPA secure if any PPT adversary has at most a negligible advantage in the security parameter λ. In addition, an SR-ABE scheme as described in Section 2.6 is said to be selectively IND-CPA secure if an Init stage is added before the Setup phase where algorithm $\mathcal{A}$ commits to the challenge attribute set $\mathbf{A}^*$ (and the challenge time period t^*) which it attempts to attack.

Notice that the above security game can also be applied to SR-ABE with ABE being CP-ABE except that the queries on access structures ($\mathbb{M}$, ρ) in the Public-Attribute-Key and Transformation-Key oracles will be replaced by attribute sets $\mathbf{A}$, and the attribute set $\mathbf{A}^*$ in the Challenge phase will be replaced by the access structure ($\mathbb{M}^*$, ρ^*).

Remarks. To prevent decryption key exposure attacks such that the ciphertext leaks no information about the underlying plaintext even if all (short-term) decryption keys of "different time periods" are exposed, which the revocable ABE schemes in [3, 5, 6, 7] following the Boldyreva-Goyal-Kumar technique cannot resist[3], Seo and Emura [30] proposed a stronger security definition for revocation schemes. To cover such attacks in the IND-CPA security of SR-ABE, given that the decryption key generated by a data user in a normal ABE scheme is now created by the server and renamed as transformation key in an SR-ABE scheme, the adversary in the above security definition is given access to an additional Transformation-Key oracle.

3. Server-Aided Revocable Attribute-Based Encryption

In this section, we propose a generic construction of SR-ABE, and analyze its security and performance.

3.1. Revocable Attribute-Based Encryption

Take the revocable KP-ABE scheme as an instance, which consists of seven algorithms. Let $\mathbf{M}$ be the message space, and $\mathbf{T}$ be the space of time periods.

- RABE.Setup(1^λ) $\rightarrow$ (par_{abe}, msk, rl, st): On input a security parameter λ, this algorithm outputs the public parameter par_{abe}, the master private key msk, an initially empty revocation list rl and a state st. This algorithm is run by the KGC.

[3]This does not contradict with the security proofs of these schemes, because such attacks are excluded from their security models.

- RABE.KeyGen(par_{abe}, msk, id, ($\mathbb{M}$, ρ), st) $\rightarrow$ ($sk_{id}^{(\mathbb{M},\rho)}$, st): On input the public parameter par_{abe}, the maser private key msk, an identity id (in the identity space) with an access structure ($\mathbb{M}$, ρ) (in the space of access structures) and a state st, this algorithm outputs a private attribute-key $sk_{id}^{(\mathbb{M},\rho)}$ and an updated state st. This algorithm is run by the KGC.

- RABE.KeyUp(par_{abe}, msk, t, rl, st) $\rightarrow$ (uk_t, st): On input the public parameter par_{abe}, the master private key msk, a time period $t \in \mathbf{T}$, a revocation list rl and a state st, this algorithm outputs the key update information uk_t and an updated state st. This algorithm is run by the KGC.

- RABE.DecKey(par_{abe}, $sk_{id}^{(\mathbb{M},\rho)}$, uk_t) $\rightarrow$ $dk_{id,t}^{(\mathbb{M},\rho)}$: On input the public parameter par_{abe}, the private attribute-key $sk_{id}^{(\mathbb{M},\rho)}$ and the key update information uk_t, this algorithm outputs a decryption key $dk_{id,t}^{(\mathbb{M},\rho)}$. This algorithm is run by each user.

- RABE.Encrypt(par_{abe}, $\mathbf{A}$, M, t) $\rightarrow$ $\mathrm{CT}_{\mathbf{A},t}$: On input the public parameter par_{abe}, an attribute set $\mathbf{A}$ (in the attribute space), a message $M \in \mathbf{M}$ and a time period $t \in \mathbf{T}$ as the input, this algorithm outputs a ciphertext $\mathrm{CT}_{\mathbf{A},t}$. This algorithm is run by each data owner, and $\mathrm{CT}_{\mathbf{A},t}$ will be stored in the cloud.

- RABE.Decrypt(par_{abe}, $dk_{id,t}^{(\mathbb{M},\rho)}$, $\mathrm{CT}_{\mathbf{A},t}$) $\rightarrow$ $M/\perp$: On input the public parameter par_{abe}, a decryption key $dk_{id,t}^{(\mathbb{M},\rho)}$ and a ciphertext $\mathrm{CT}_{\mathbf{A},t}$, this algorithm outputs a message M or a failure symbol $\perp$. This algorithm is run by a user.

- RABE.Revoke(id, t, rl, st) $\rightarrow$ (rl, st): On input an identity id to be revoked, a time period t, a revocation list rl and a state st, this algorithm outputs an updated revocation list rl and an updated status st. This algorithm is run by the KGC.

The correctness of the above RABE scheme requires that for any security parameter λ, any message $M \in \mathbf{M}$, and any identity id (in the identity space) with any authorized attribute set $\mathbf{A}$ (in the attribute space) of any access structure ($\mathbb{M}$, ρ) (in the space of access structures), if the user id is not revoked at the time period t, and if (par_{abe}, msk, rl, st) $\leftarrow$ RABE.Setup(1^λ), ($sk_{id}^{(\mathbb{M},\rho)}$, st) $\leftarrow$ RABE.KeyGen(par_{abe}, msk, id, ($\mathbb{M}$, ρ), st), (uk_t, st) $\leftarrow$ RABE.KeyUp(par_{abe}, msk, t, rl, st), $dk_{id,t}^{(\mathbb{M},\rho)}$ $\leftarrow$ RABE.DecKey(par_{abe}, $sk_{id}^{(\mathbb{M},\rho)}$, uk_t), $\mathrm{CT}_{\mathbf{A},t} \leftarrow$ RABE.Encrypt(par_{abe}, $\mathbf{A}$, M, t), we have that RABE.Decrypt(par_{abe}, $\mathrm{CT}_{\mathbf{A},t}$, $dk_{id,t}^{(\mathbb{M},\rho)}$) $= M$.

The security game, IND-CPA security for the above RABE scheme is defined between an adversary algorithm $\mathcal{A}$ and a challenger algorithm $\mathcal{B}$ as follows.

- Setup. Algorithm $\mathcal{B}$ runs the setup algorithm, and gives the public parameter par_{abe} to algorithm $\mathcal{A}$, and keeps the master private key msk, an initially empty revocation list rl and a state st.

- Phase 1. Similar to that in the IND-CPA security game of SR-RABE but without issuing queries to the Private-User-Key oracle. Algorithm $\mathcal{A}$ adaptively issues queries to the Private-Attribute-Key oracle, Key-Update oracle, Decryption-Key oracle and Revocation oracle. Algorithm $\mathcal{B}$ returns the results to algorithm $\mathcal{A}$ by running the algorithms including RABE.KeyGen(par_{abe}, msk, id, ($\mathbb{M}$, ρ), st), RABE.KeyUp(par_{abe}, msk, t, rl, st), RABE.DecKey(par_{abe}, id, $sk_{id}^{(\mathbb{M},\rho)}$, tku_t).

- Challenge. Algorithm $\mathcal{A}$ outputs two messages M_0^*, M_1^* of the same size, an attribute set $\mathbf{A}^*$ and a time period t^* following the constraint that if an identity id^* with an attribute set $\mathbf{A}^*$ that satisfies a query on (id^*, ($\mathbb{M}^*$, ρ^*)) issued to the Private-Attribute-Key oracle, then the Revocation oracle must be queried on (id^*, t) on $t = t^*$ or any t occurs before t^*, and the Decryption-Key oracle cannot be queried on (id^*, t^*). Algorithm $\mathcal{B}$ randomly chooses $\gamma \in \{0, 1\}$, and forwards the challenge ciphertext $\mathrm{CT}_{\mathbf{A}^*,t^*}$ to algorithm $\mathcal{A}$ by running Encrypt(par_{abe}, $\mathbf{A}^*$, t^*, M_γ^*).

- Phase 2. Algorithm $\mathcal{A}$ continues issuing queries to algorithm $\mathcal{B}$ as in Phase 1, following the restrictions defined in the Challenge phase.

- Guess. Algorithm $\mathcal{A}$ makes a guess γ' for γ, and it wins the game if $\gamma' = \gamma$.

The advantage of algorithm $\mathcal{A}$ in this game is defined as $\Pr[\gamma = \gamma'] - 1/2$. An RABE scheme as above is IND-CPA secure if any PPT adversary has at most a negligible advantage in the security parameter λ. In addition, an RABE scheme as above is said to be selectively IND-CPA secure if an Init stage is added before the Setup phase where algorithm $\mathcal{A}$ commits to the challenge attribute set $\mathbf{A}^*$ (and the challenge time period t^*) which it attempts to attack.

Likewise, the above framework, correctness and security definition can be applied to a revocable CP-ABE scheme by replacing the attribute set in the RABE.Encrypt algorithm and the Challenge phase with the access structure, and the relevant access structure in the RABE.KeyGen, RABE.DecKey and RABE.Decrypt algorithms and the queries in Phase 1 with the attribute set.

3.2. Generic Construction

Suppose that the PKE scheme (the PKE scheme can use the Linear encryption scheme introduced by Boneh, Boyen and Shacham [26][4], a natural variant of the ElGamal encryption [29] which is secure in the generic groups under the Linear assumption [26]) consists of four algorithms as (PKE.Setup, PKE.KeyGen, PKE.Enc, PKE.Dec), and the RABE (revocable KP-ABE) scheme (e.g., [3, 5, 6]) is composed of sever algorithms as (RABE.Setup, RABE.KeyGen, RABE.KeyUp, RABE.DecKey, RABE.Encrypt, RABE.Decrypt, RABE.Revoke). We describe the proposed generic construction on SR-ABE in the setting of KP-ABE (note that this generic construction can be applied to SR-ABE in the setting of CP-ABE by replacing the access structure in the PubKG algorithm with the attribute set and the attribute set in the Encrypt algorithm with the access structure).

- Setup. This algorithm runs the PKE.Setup and RABE.Setup algorithms, and outputs the public parameter $par = (par_{pke}, par_{abe})$, the master private key msk, a revocation list rl and a state st.

- UserKG. This algorithm runs the PKE.KeyGen algorithm to obtain (pk, sk), and outputs (pk, sk) as the public and private user-key pair (pk_{id}, sk_{id}) for a user id.

- PubKG. This algorithm runs the RABE.KeyGen algorithm to obtain $(sk_{id}^{(\mathbb{M},\rho)}, st)$, and then it runs the PKE.Enc algorithm to encrypt $sk_{id}^{(\mathbb{M},\rho)}$ to obtain a ciphertext $pk_{id}^{(\mathbb{M},\rho)}$, and outputs $(pk_{id}^{(\mathbb{M},\rho)}, st)$ as the public attribute-key and an update state.

- TKeyUp. This algorithm runs the RABE.KeyUp algorithm to obtain (uk_t, st), and outputs (uk_t, st) as the transformation key update information tku_t and an updated state st.

- TranKG. This algorithm runs the RABE.DecKey algorithm on $pk_{id}^{(\mathbb{M},\rho)}$ and tku_t to obtain $dk_{id,t}^{(\mathbb{M},\rho)}$, and outputs $dk_{id,t}^{(\mathbb{M},\rho)}$ as a transformation key $tk_{id,t}^{(\mathbb{M},\rho)}$ for user id during a time period t.

- Encrypt. This algorithm runs the RABE.Encrypt algorithm, and outputs a ciphertext $\mathrm{CT}_{\mathbf{A},t}$.

- Transform. This algorithm runs the RABE.Decrypt algorithm on $tk_{id,t}^{(\mathbb{M},\rho)}$ and $\mathrm{CT}_{\mathbf{A},t}$, and outputs a transformed ciphertext CT_{id}.

[4]Notice that as almost all existing ABE schemes are built from the bilinear maps, it is crucial to require the PKE scheme to be secure in the bilinear groups. Therefore, the ElGamal encryption scheme in [29] secure under the DDH assumption can not be applied here, because the DDH assumption is easy in the bilinear groups.

- Decrypt. This algorithm runs the PKE.Decrypt algorithm on sk_{id} and CT_{id}, and outputs a message M.

- Revoke. This algorithm runs the RABE.Revoke algorithm, and outputs an updated revocation list rl and an updated status st.

In order to achieve the correctness (defined in Section 2.6) of the above SR-ABE scheme, we require that $M' = \mathrm{PKE.Decrypt}(par, \mathrm{CT}_{id}, sk_{id}) = \mathrm{PKE.Decrypt}(par, \mathrm{RABE.Decrypt}(par, tk_{id,t}^{(\mathbb{M},\rho)}, \mathrm{CT}_{\mathbf{A},t}), sk_{id}) = \mathrm{RABE.Decrypt}(par, \mathrm{PKE.Decrypt}(par, tk_{id,t}^{(\mathbb{M},\rho)}, sk_{id}), \mathrm{CT}_{\mathbf{A},t}) = \mathrm{RABE.Decrypt}(par, dk_{id,t}^{(\mathbb{M},\rho)}, \mathrm{CT}_{\mathbf{A},t}) = M$. In other words, the underlying RABE scheme in the proposed generic construction of SR-ABE should satisfy the property that $\mathrm{PKE.Decrypt}(par, \mathrm{RABE.Decrypt}(par, tk_{id}^{(\mathbb{M},\rho)}, \mathrm{CT}_{\mathbf{A},t}), sk_{id}) = \mathrm{RABE.Decrypt}(par, \mathrm{PKE.Decrypt}(par, tk_{id,t}^{(\mathbb{M},\rho)}, sk_{id}), \mathrm{CT}_{\mathbf{A},t})$. Briefly, the sequence of the decryption in PKE and the decryption in RABE can be interchanged, without affecting the correctness of the scheme.

Notes. A drawback of SR-ABE is that it requires the server to be on-line all the time to meet the the decryption requests from users. One solution to this problem is to make the server partially decrypt the incoming ciphertexts for all eligible users beforehand and store the corresponding partially decrypted ciphertexts for them, but this wastes both the computation and storage resources.

Theorem 1. Assuming that the underlying RABE scheme is (selectively) IND-CPA secure and PKE scheme is IND-CPA secure, then the above generic construction on SR-ABE is (selectively) IND-CPA secure.

Proof. Assume that there exists an adversary algorithm $\mathcal{A}$ that breaks the IND-CPA security of the proposed generic construction on SR-ABE. Then we can build an adversary algorithm $\mathcal{A}'$ that breaks the IND-CPA security of the underlying RABE scheme or the underlying PKE scheme. Denote $\mathcal{B}_0$ by the challenger algorithm in the IND-CPA security game of the underlying RAKE scheme, and $\mathcal{B}_1$ by the challenger algorithm in the IND-CPA security game of the underlying PKE scheme. Note that for the selective IND-CPA security, there will be an Init stage before the Setup phase where algorithm $\mathcal{A}$ outputs a challenge attribute set $\mathbf{A}^*$ (and a challenge time period t^*), which algorithm $\mathcal{A}'$ sets as its own output in the Init stage for the IND-CPA security game of the underlying RABE scheme.

- Setup. Algorithm $\mathcal{A}'$ is given par_{abe} from algorithm $\mathcal{B}_0$ of the RABE scheme, and par_{pke}, pk from the algorithm $\mathcal{B}_1$ of the PKE scheme. Algorithm $\mathcal{A}'$ sends $par = (par_{pke}, par_{abe})$ to algorithm $\mathcal{A}$.

- Phase 1. Algorithm $\mathcal{A}$ adaptively issues the following sequence of queries to algorithm $\mathcal{A}'$.

  - Private-User-Key oracle on an identity id. Algorithm $\mathcal{A}'$ returns sk_{id} by running $(sk_{id}, pk_{id}) \leftarrow \text{UserKG}(par, id)$. It adds (id, sk_{id}, pk_{id}) to a list L so that the same (sk_{id}, pk_{id}) is used for all future queries on id.

  - Public-Attribute-Key oracle on an identity id and an access structure $(\mathbb{M}, \rho)$. Algorithm $\mathcal{A}'$ issues the private attribute-key generation query on $(id, (\mathbb{M}, \rho))$ to algorithm $\mathcal{B}_0$, and algorithm $\mathcal{B}_0$ returns $sk_{id}^{(\mathbb{M},\rho)}$ to algorithm $\mathcal{A}$.

    1. If (id, sk_{id}, pk_{id}) already exists in the list L, algorithm $\mathcal{A}'$ returns $pk_{id}^{(\mathbb{M},\rho)}$ by running $pk_{id}^{(\mathbb{M},\rho)} \leftarrow \text{PKE.Enc}(par, pk_{id}, sk_{id}^{(\mathbb{M},\rho)})$.
    2. Otherwise, algorithm $\mathcal{A}'$ runs $(sk_{id}, pk_{id}) \leftarrow \text{UserKG}(par, id)$ to obtains pk_{id}, and then returns $pk_{id}^{(\mathbb{M},\rho)}$ by running $pk_{id}^{(\mathbb{M},\rho)} \leftarrow \text{PKE.Enc}(par, pk_{id}, sk_{id}^{(\mathbb{M},\rho)})$. Note that at some point, algorithm $\mathcal{A}'$ randomly chooses $sk_{id^*}^{(\mathbb{M},\rho)}$, and returns $pk_{id^*}^{(\mathbb{M},\rho)}$ by running $pk_{id^*}^{(\mathbb{M},\rho)} \leftarrow \text{PKE.Encrypt}(par, pk, sk_{id^*}^{(\mathbb{M},\rho)})$. In other words, it implicitly sets the public key pk as the public key for id^*, and puts $(id^*, \perp, pk)$ to the list L. Because of the IND-CPA security of the PKE scheme, algorithm $\mathcal{A}$ cannot distinguish whether $sk_{id^*}^{(\mathbb{M},\rho)}$ is randomly chosen or not. Otherwise, algorithm $\mathcal{A}'$ can make use of algorithm $\mathcal{A}$ to break the IND-CPA security of the underlying PKE scheme.

  - Transformation-Key-Update oracle on a time period t. Algorithm $\mathcal{A}'$ issues the key update query on t to algorithm $\mathcal{B}_0$, and returns the result from algorithm $\mathcal{B}_0$ to algorithm $\mathcal{A}$ as tku_t.

  - Transformation-Key oracle on a time period t and an identity id with an access structure $(\mathbb{M}, \rho)$. Algorithm $\mathcal{A}'$ issues the decryption key query on $(id, (\mathbb{M}, \rho), t)$ to algorithm $\mathcal{B}_0$, and sends the result from algorithm $\mathcal{B}_0$ as the transformation key $tk_{id,t}^{(\mathbb{M},\rho)}$ to algorithm $\mathcal{A}$.

  - Revoke oracle on an identity id and a time period t. Algorithm $\mathcal{A}'$ issues the revoke query on (id, t) to algorithm $\mathcal{B}_0$, and forwards the updated revocation list rl form algorithm $\mathcal{B}_0$ to algorithm $\mathcal{A}$.

- Challenge. Algorithm $\mathcal{A}$ outputs two messages M_0, M_1, an attribute set $\mathbf{A}^*$ and a time period t^*. Algorithm $\mathcal{A}'$ forwards them to algorithm $\mathcal{B}_0$ to obtain the challenge ciphertext CT^*. Algorithm $\mathcal{A}'$ returns CT^* to algorithm $\mathcal{A}$.

- Phase 2. Algorithm $\mathcal{A}$ continues issuing queries to algorithm $\mathcal{A}'$ as in Phase 1, following the restrictions defined in the security model.

- Guess. Algorithm $\mathcal{A}$ makes a guess γ' for γ, algorithm $\mathcal{A}'$ forwards γ' to algorithm $\mathcal{B}_0$ as the guess to the IND-CPA security game for the underlying RABE scheme.

Denote the event that algorithm $\mathcal{A}'$ sets pk as the public key for an identity id^* as Event. It is not difficult to see that the simulation is the same as that in the real game except for the occurrence of the event Event. Denote q_{PAK} by the number of queries issued to the Public-Attribute-Key oracle. With the probability $1/q_{\mathrm{PAK}}$, the event Event happens for the identity id^*. In this case, the private attribute-key $sk_{id^*}^{(\mathbb{M},\rho)}$ query for some ($\mathbb{M}$, ρ) satisfied by $\mathbf{A}^*$ would never be previously asked to algorithm $\mathcal{B}_0$. Therefore, the simulation is correct.

To conclude, if algorithm $\mathcal{A}$ can win the IND-CPA game of SR-ABE with the non-negligible probability ϵ, then algorithm $\mathcal{A}'$ can win the IND-CPA game of the underlying ABE scheme with the probability $\epsilon/q_{\mathrm{PAK}}$. □

3.3. Instantiation

Below we give a concrete server-aided revocable KP-ABE scheme, based on the Linear encryption scheme [26] and the indirectly revocable KP-ABE scheme in the prime-order groups in [5][5]. Let the attribute space be Z_p, the message space be G_1, and the space of the time periods be Z_p.

- Setup. This algorithm takes the security parameter λ as the input. It randomly chooses a group G of a prime order p with a generator $g \in G$, and defines a prime-order bilinear pairing $\hat{e} : G \times G \to G_1$. Also, it randomly chooses u_0, ..., u_d, h_0, ..., $h_m \in G$, $\alpha \in Z_p$. Let rl be an empty list storing revoked users and BT be a binary tree with at least N leaf nodes. Define two functions mapping elements in Z_p to elements in G as $P(x) = \prod_{j=0}^{d} {u_j}^{x^j}$, $F(x) = \prod_{j=0}^{m} {h_j}^{x^j}$. The public parameter is $par = (g, \hat{e}(g,g)^{\alpha}, u_0, ..., u_d, h_0, ..., h_m)$ along with rl and st where st is a state set to be BT, and the master private key is $msk = \alpha$.

- UserKG. This algorithm takes the public parameter par and an identity id as the input. It randomly chooses β_1, $\beta_2 \in Z_p$, $g_3 \in G$, and computes $g_1 = {g_3}^{\frac{1}{\beta_1}}$, $g_2 = {g_3}^{\frac{1}{\beta_2}}$. It outputs the public and private user-key pair as $(pk_{id}, sk_{id}) = ((g_1, g_2, g_3), (\beta_1, \beta_2))$.

[5]For the construction of a server-aided revocable CP-ABE scheme, please refer to the conference version [11], or apply the generic construction on SR-ABE with ABE being CP-ABE to a revocable CP-ABE scheme. Due to the space limit, we omit the details here.

- PubKG. This algorithm takes the public parameter par, the master private key msk, an identity id with a public user-key pk_{id} and an access structure $(\mathbb{M}, \rho)$, and a state st as the input. Let $\mathbb{M}$ be an $l \times n$ matrix, and ρ be a function mapping each row $\mathbb{M}_i$ of $\mathbb{M}$ to an attribute $\rho(i)$. It randomly chooses an undefined leaf node θ from the binary tree BT, and stores id in this node. Then, for each node $x \in \text{Path}(\theta)$, it runs as follows.

  1. To share α with $(\mathbb{M}, \rho)$, it randomly chooses $y_{x,2}, ..., y_{x,n} \in Z_p$, and sets $\overrightarrow{v_x} = (\alpha, y_{x,2}, ..., y_{x,n})$. For $i = 1, ..., l$, it computes $v_{x,i} = \mathbb{M}_i \cdot \overrightarrow{v_x}$, where $\mathbb{M}_i$ is the vector corresponding to the i-th row of $\mathbb{M}$.
  2. It fetches g_x from the node x. If x has not been defined, it randomly chooses $g_x \in G$. For $i = 1, ..., l$, it computes $g'_{x,i} = g^{v_{x,i}}/g_x$, and stores g_x in the node x.
  3. It randomly chooses $r_1, r_2, r_{x,1}, ..., r_{x,l} \in Z_p$, and computes

$$D_1 = {g_1}^{r_1}, \quad D_2 = {g_2}^{r_2}, \quad D^{(2)}_{x,i} = g^{r_{x,i}}$$
$$D^{(1)}_{x,i} = {g_3}^{r_1+r_2} \cdot g'_{x,i} \cdot F(\rho(i))^{r_{x,i}}.$$

  It outputs the public attribute-key $pk^{\mathbb{M},\rho}_{id} = (D_1, D_2, \{x, \{D^{(1)}_{x,i}, D^{(2)}_{x,i}\}_{i\in[1,l]}\}_{x\in\text{Path}(\theta)})$.

- TKeyUp. This algorithm takes the public parameter par, the master private key msk, a time period t, a revocation list rl and a state st as the input. For all $x \in$ KUNodes(BT, rl, t), it fetches g_x (note that g_x is always predefined in the PubKG algorithm) from the node x. Also, it randomly chooses $s_x \in Z_p$, and computes $U^{(1)}_x = g_x \cdot P(t)^{s_x}$, $U^{(2)}_x = g^{s_x}$. It outputs the transformation key update information $tku_t = (\{x, U^{(1)}_x, U^{(2)}_x\}_{x\in\text{KUNodes(BT},rl,t)})$.

- TranKG. This algorithm takes the public parameter par, an identity id with a public attribute-key $pk^{(\mathbb{M},\rho)}_{id}$ and the transformation key update information tku_t as the input. Denote I as Path(θ), J as KUNodes(BT, rl, t). It parses $pk^{\mathbb{M},\rho}_{id}$ as $(D_1, D_2, \{x, \{D^{(1)}_{x,i}, D^{(2)}_{x,i}\}_{i\in[1,l]}\}_{x\in I})$, tku_t as $(\{x, U^{(1)}_x, U^{(2)}_x\}_{x\in J})$ for some set of nodes I, J. If $I \cap J = \emptyset$, it returns $\perp$. Otherwise, for any node $x \in I \cap J$, it randomly chooses $r'_{x,1}, ..., r'_{x,k}, s'_x \in Z_p$, and computes

$$\begin{aligned} tk^{(1)}_i &= D^{(1)}_{x,i} \cdot U^{(1)}_x \cdot F(\rho(i))^{r'_{x,i}} \cdot P(t)^{s'_x} \\ &= {g_3}^{r_1+r_2} \cdot g^{v_{x,i}} \cdot F(\rho(i))^{r_{x,i}+r'_{x,i}} \cdot P(t)^{s_x+s'_x}, \\ tk^{(2)}_i &= D^{(2)}_{x,i} \cdot g^{r'_{x,i}} = g^{r_{x,i}+r'_{x,i}}, \\ tk^{(3)} &= U^{(2)}_x \cdot g^{s'_x} = g^{s_x+s'_x}. \end{aligned}$$

  It outputs the transformation key $tk^{\mathbb{M},\rho}_{id,t} = (D_1, D_2, \{tk^{(1)}_i, tk^{(2)}_i\}_{i\in[1,l]}, tk^{(3)})$.

- Encrypt. This algorithm takes the public parameter par, an attribute set $\mathbf{A}$, a time period t and a message M as the input. Let $\mathbf{A}$ be $\{A_1, ...A_k\}$. It randomly chooses $\mu \in Z_p$, and computes

$$C = \hat{e}(g, g)^{\alpha\mu} \cdot M, \quad C^{(1)} = g^{\mu},$$
$$C_i^{(2)} = F_1(A_i)^{\mu}, \quad C^{(3)} = P(t)^{\mu}.$$

  It outputs the ciphertext $\mathrm{CT}_{\mathbf{A},t} = (\mathbf{A}, t, C, C^{(1)}, \{C_i^{(2)}\}_{i\in[1,k]}, C^{(3)})$.

- Transform. This algorithm takes the public parameter par, an identity id with a transformation key $tk_{id,t}^{\mathbb{M},\rho}$ over an access structure $(\mathbb{M}, \rho)$ and a time period t and a ciphertext $\mathrm{CT}_{\mathbf{A},t}$ over an attribute set $\mathbf{A}$ and the same time period t as the input. Suppose that the attribute set $\mathbf{A}$ satisfies the access structure $(\mathbb{M}, \rho)$. Let I be defined as $I = \{i : \rho(i) \in \mathbf{A}\}$. Denote by $\{w_i \in Z_p\}_{i\in I}$ a set of constants such that if $\{v_i\}$ are valid shares of any secret μ according to $\mathbb{M}$, then $\sum_{i\in I} w_i v_i = \mu$. It parses $\mathrm{CT}_{\mathbf{A},t}$, and computes

$$C' = \prod_{i\in I} \left(\frac{\hat{e}(C_i^{(2)}, tk_i^{(2)})\hat{e}(C^{(3)}, tk^{(3)})}{\hat{e}(tk_i^{(1)}, C^{(1)})} \right)^{w_i}$$
$$= \frac{1}{\hat{e}({g_3}^{r_1+r_2}, C^{(1)})\hat{e}(g, C^{(1)})^{\alpha}}.$$

  It outputs the transformed ciphertext $\mathrm{CT}_{id} = (C', C^{(1)}, D_1, D_2, C)$.

- Decrypt. This algorithm takes the public parameter par, an identity id with a private user-key sk_{id} and a transformed ciphertext CT_{id} as the input. It outputs the message M as $M = C' \cdot \hat{e}({D_1}^{\beta_1} {D_2}^{\beta_2}, C^{(1)}) \cdot C$.

- Revoke. This algorithm takes an identity id, a time period t, a revocation list rl and a state st as the input. For all the nodes x associated with identity id, it adds (x, t) to rl, and outputs the updated rl and st.

Theorem 2. Assuming that the underlying revocable KP-ABE scheme is selectively IND-CPA secure, and the underlying PKE scheme is IND-CPA secure, then the above server-aided revocable KP-ABE scheme is selectively IND-CPA secure.

Proof. It has been proved in [5] that the underlying revocable KP-ABE scheme is selectively IND-CPA secure. Also, the Linear encryption scheme [26] is known to be IND-CPA secure. According to Theorem 1, it is clear that the proposed server-aided revocable KP-ABE scheme is selectively IND-CPA secure. □

Table 1: Comparison between SR-ABE and the existing solutions on RABE.

	RABE in [3]	RABE in [5]	RABE in [6]	RABE in [10]	RABE in [8, 9]	SR-ABE
Type	KP-ABE	KP-ABE	Generic	Generic	CP-ABE	Generic
Server	–	–	Semi-trust	Semi-trust	Untrust	Untrust
Secure Channel	Yes	Yes	Yes	Yes	No	No
Key Updates	$O(R\log(\frac{N}{R}))$	$O(R\log(\frac{N}{R}))$	$O(R\log(\frac{N}{R}))$	$O(R\log(\frac{N}{R}))$	$O(R\log(\frac{N}{R}))$	$O(R\log(\frac{N}{R}))$
User Key	$O(l\log N)$	$O(l\log N)$	$O(l\log N)$ & $O(k\log N)$	$O(l\log N)$ & $O(k\log N)$	$O(1)$	$O(1)$
Decryption Cost	$\geq 2(\mathrm{E}+\mathrm{P})$	$\geq 3\mathrm{E}+4\mathrm{P}$	$\geq \mathrm{E}+\mathrm{P}$	$\geq \mathrm{E}+\mathrm{P}$	2P	$2\mathrm{E}+\mathrm{P}$

3.4. Comparison

To our knowledge, in addition to the work in this paper, constructions in [3], [5], [6], [10], and [8, 9] are also about how to achieve indirect revocation in ABE. Recall that the goal in this paper is to achieve indirect user revocation in an ABE scheme by outsourcing users' workloads to an untrusted server such that the KGC indirectly accomplishes user revocation by stopping updating the keys for revoked users. In [3], a KP-ABE scheme with indirect revocation is proposed where the KGC enables user revocation by stopping posting key update information for revoked users, thereby forcing revoked users to be unable to update their decryption keys. A hybrid revocable KP-ABE system is given in [5], which allows a data owner to select either direct or indirect revocation mode when encrypting a message. In [8, 9], in addition to prevent transformation key exposure attacks, the proposed revocable CP-ABE scheme is also secure against private user-key exposure attacks. A generic way to realize ABE supporting dynamic credentials is provided in [6], where the KGC indirectly accomplishes revocation by stopping updating the keys for revoked users. The generic revocable CP-ABE scheme introduced in [10] which is secure against key exposure attacks has an addition ciphertext delegation function.

Table 1 compares our generic SR-ABE construction with several existing solutions on RABE proposed in [3], [5], [6], [10] and [8, 9]. Assume that the underlying PKE scheme used in SR-ABE is the Linear encryption scheme introduced by Boneh, Boyen and Shacham [26]. Let N be the number of all users, 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, "E" denote exponentiation operation, and "P" denote pairing operation, respectively. It is straightforward to see from Table 1 that the constructions in [3], [5], [6] and [10] require secure channels between the KGC and every user for key transmission, and every user to keep a private key of which the size is determined by their attributes and the associated nodes in the predefined binary tree. While the constructions in [8, 9] achieve favourable security and performance, they are concrete schemes and their decryption costs are more expensive due to the security enhancement. Clearly, SR-ABE has an edge over the previous solutions in that it does not require any secure channels between the participants, and is secure against collusion attacks between the untrusted server and revoked users. Also, SR-ABE achieves the desirable efficiency in the decryption run by users, which always requires two exponentiation and one pairing operations (regardless of the complexity of the access structure).

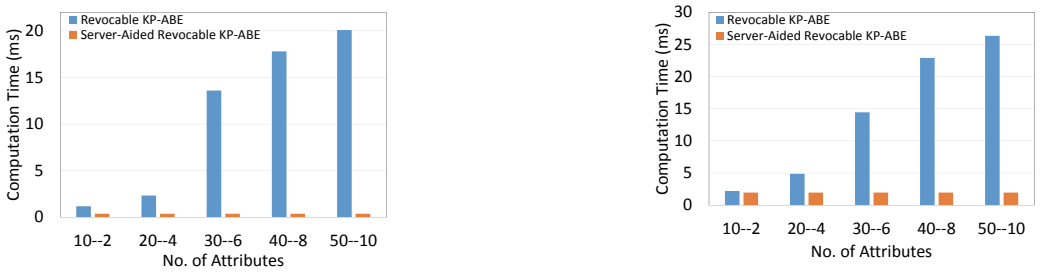
3.5. Experimental Results

In order to clearly show that SR-ABE has an edge over RABE in the computational overhead of the user, we implement the revocable KP-ABE scheme in [5] and the proposed server-aided revocable KP-ABE scheme in Charm [31][6]. We use the Charm-0.43 and Python 3.4 in our implementation. Along with the Charm-0.43, we install the PBC library for the underlying cryptographic operations. Our experiments are run on a laptop with Intel Core i5-4210U CPU @ 1.70GHz and 8.00 GB RAM running 64-bit Ubuntu 16.04.

We conduct the experiments over the elliptic curves: SS512 and MNT159 to provide security level of 80-bit, where SS512 is a supersingular elliptic curve with the symmetric Type 1 pairing on it, and MNT159 is an asymmetric Type 3 pairing. In our experiments, all attribute-keys and ciphertexts are randomly generated over randomly chosen access structures and attribute sets. We test the average computation time spent by a user in decrypting ciphertexts over 10 to 50 attributes using attribute-keys over access policies of 2 to 10 attributes (See Fig. 3, note that the computational cost for decrypting a ciphertext of the server in the given server-aided revocable KP-ABE scheme is similar to that of the data user in the underlying revocable KP-ABE scheme [5]). In the revocable KP-ABE scheme [5], the average computation time of decrypting ciphertexts of 10 to 50 attributes using attribute-keys for access structures with 2 to 10 attributes ranges from 1.2ms to 21.0ms for the SS512 curve and 2.5ms to 27.0ms for the MNT159 curve, respectively, while in our server-aided revocable KP-ABE scheme,

[6]For the explicit information on Charm, please refer to [31].

the average computation time of decrypting ciphertexts of 10 to 50 attributes using attribute-keys for access structures with 2 to 10 attributes is about 0.4ms in terms of the SS512 curve and 2.0ms in terms of the MNT159 curve, respectively. From Fig. 3, it is easy to see that the framework of SR-ABE can be applied to greatly reduce the computational overheads of users in decrypting ciphertexts, where the computation time for a user to decrypt a ciphertext is independent to the number of attributes associated with the ciphertexts and the attribute-keys.



(a) SS512 (b) MNT159

Figure 3: Computation time of the user to obtain the encrypted message in a ciphertext.

4. Applications

The market of cloud computing services has experienced exponential growth during the last decade. However, since software systems are not guaranteed to be bug-free and hardware platforms are not under the direct control of data owners in the cloud, there exist various kinds of security threats. A common solution to protect the data privacy is to encrypt the data before the data reaches the cloud. This keeps the data private even if service providers are compromised or untrusted. However, it has been very challenging to efficiently share large amounts of data items that are encrypted using traditional cryptographic techniques because of the difficulty in distributing decryption keys and managing decryption key revocations. SR-ABE provides a practical and scalable solution for access control over encrypted data in the cloud. Note that the ciphertext transformation performed by the untrusted server in Fig. 2 is functionally equivalent to outsourced ABE decryption [32]. As a matter of fact, most of the operational cost resulted by ABE decryption in SR-ABE is performed by the untrusted server while a user only needs to perform two exponentiation and one pairing operations to decrypt a ciphertext. Therefore, in addition to support server-aided user revocation, SR-ABE also greatly reduces the computational costs for users in decrypting ciphertexts and is highly suitable for users using mobile devices with limited power and computational capabilities.

Email is an indispensable means of communication for both corporations and individuals. In order to provide security for email systems, existing solutions such as Pretty Good Privacy (PGP) [33] and Secure/Multipurpose Internet Mail Extensions (S/MIME) [34] employ traditional cryptographic techniques and hence are cumbersome and undesirable to use for group-based Email messaging systems. Clearly, securing email is another area in which SR-ABE finds a promising application. In a secure email system built from SR-ABE, the email server can also act as the "untrusted server" of SR-ABE (referring to Fig. 2) to undertake the responsibility of user revocation and ciphertext transformation.

A blockchain is a decentralized, distributed and public digital ledger that is used to record digital transactions so that any involved record cannot be altered retroactively, without the alteration of all subsequent blocks [35]. The blockchain allows the participants to verify transactions independently, and is widely applied in many cryptocurrencies for all kinds of monetary transactions. Since the server in SR-ABE does not hold any secret information, and all its operations can be performed by anybody, SR-ABE can be simply moved to the blockchain to confirm the correctness of the computation performed by the server. This is useful in practice, as the server may charge for the provided service.

5. Conclusions

To achieve efficient user revocation in attribute-based encryption (ABE), Cui et al. [11] proposed a server-aided revocable attribute-based encryption (SR-ABE) scheme. In this paper, we revisited the notion of SR-ABE, and considered it one step further by proposing a generic construction of SR-ABE which can transform any revocable ABE (RABE) scheme into an SR-ABE scheme. After proving that the proposed generic SR-ABE construction is secure under the defined security model, we presented an instantiation to transform a secure RABE scheme into a secure SR-ABE scheme using the proposed generic construction. Compared with the previous solutions on RABE, SR-ABE has three salient advantages. Firstly, SR-ABE outsources almost all operational overheads of users resulted in the key update phase to an untrusted server. Secondly, instead of storing a private key, of which the size is logarithmic to the number of users, by each user as in most of the existing revocable ABE schemes, each user in SR-ABE only needs to keep a private key of one group element. Thirdly, in SR-ABE, most of the computational overheads in decryption is outsourced to the untrusted server, and a user is only required to perform a small amount of computations to decrypt a ciphertext.

Acknowledgments

This research work is supported by the National Natural Science Foundation of China under the Grant Number 61872292, the Singapore National Research Foundation under the NCR Award Number NRF2014NCR-NCR001-012 and the AXA Research Fund.

References

[1] A. Sahai, B. Waters, Fuzzy identity-based encryption, in: Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings, Vol. 3494 of Lecture Notes in Computer Science, Springer, 2005, pp. 457–473.

[2] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006, Vol. 5126 of Lecture Notes in Computer Science, Springer, 2006, pp. 89–98.

[3] A. Boldyreva, V. Goyal, V. Kumar, Identity-based encryption with efficient revocation, in: Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008, ACM, 2008, pp. 417–426.

[4] D. Naor, M. Naor, J. Lotspiech, Revocation and tracing schemes for stateless receivers, in: Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings, Vol. 2139 of Lecture Notes in Computer Science, Springer, 2001, pp. 41–62.

[5] N. Attrapadung, H. Imai, Attribute-based encryption supporting direct/indirect revocation modes, in: Cryptography and Coding, 12th IMA International Conference, Cryptography and Coding 2009, Cirencester, UK, December 15-17, 2009. Proceedings, Vol. 5921 of Lecture Notes in Computer Science, Springer, 2009, pp. 278–300.

[6] A. Sahai, H. Seyalioglu, B. Waters, Dynamic credentials and ciphertext delegation for attribute-based encryption, in: Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings, Vol. 7417 of Lecture Notes in Computer Science, Springer, 2012, pp. 199–217.

[7] H. Cui, R. H. Deng, Revocable and decentralized attribute-based encryption, Comput. J. 59 (8) (2016) 1220–1235.

[8] B. Qin, Q. Zhao, D. Zheng, H. Cui, Server-aided revocable attribute-based encryption resilient to decryption key exposure, in: Cryptology and Network Security - 16th International Conference, CANS 2017, Hong Kong, China, November 30 - December 2, 2017, Revised Selected Papers, Vol. 11261 of Lecture Notes in Computer Science, Springer, 2017, pp. 504–514.

[9] B. Qin, Q. Zhao, D. Zheng, H. Cui, (dual) server-aided revocable attribute-based encryption with decryption key exposure resistance, Inf. Sci. 490 (2019) 74–92.

[10] S. Xu, G. Yang, Y. Mu, Revocable attribute-based encryption with decryption key exposure resistance and ciphertext delegation, Inf. Sci. 479 (2019) 116–134.

[11] H. Cui, R. H. Deng, Y. Li, B. Qin, Server-aided revocable attribute-based encryption, in: Computer Security - ESORICS 2016 - 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part II, Vol. 9879 of Lecture Notes in Computer Science, Springer, 2016, pp. 570–587.

[12] D. Boneh, M. Franklin, Identity-based encryption from the weil pairing, in: CRYPTO, Vol. 2139 of Lecture Notes in Computer Science, Springer-Verlag, 2001, pp. 213–219.

[13] Y. Hanaoka, G. Hanaoka, J. Shikata, H. Imai, Identity-based hierarchical strongly key-insulated encryption and its application, in: Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings, Vol. 3788 of Lecture Notes in Computer Science, Springer, 2005, pp. 495–514.

[14] D. Boneh, X. Ding, G. Tsudik, C. Wong, A method for fast revocation of public key certificates and security capabilities, in: 10th USENIX Security Symposium, August 13-17, 2001, Washington, D.C., USA, USENIX, 2001.

[15] X. Ding, G. Tsudik, Simple identity-based cryptography with mediated RSA, in: Topics in Cryptology - CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003, San Francisco, CA, USA, April 13-17, 2003, Proceedings, Vol. 2612 of Lecture Notes in Computer Science, Springer, 2003, pp. 193–210.

[16] B. Libert, J. Quisquater, Efficient revocation and threshold pairing based cryptosystems, in: Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing, PODC 2003, Boston, Massachusetts, USA, July 13-16, 2003, ACM, 2003, pp. 163–171.

[17] J. Baek, Y. Zheng, Identity-based threshold decryption, in: Public Key Cryptography - PKC 2004, 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004, Vol. 2947 of Lecture Notes in Computer Science, Springer, 2004, pp. 262–276.

[18] K. Liang, J. K. Liu, D. S. Wong, W. Susilo, An efficient cloud-based revocable identity-based proxy re-encryption scheme for public clouds data sharing, in: Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part I, Vol. 8712 of Lecture Notes in Computer Science, Springer, 2014, pp. 257–272.

[19] J. Li, J. Li, X. Chen, C. Jia, W. Lou, Identity-based encryption with outsourced revocation in cloud computing, IEEE Trans. Computers 64 (2) (2015) 425–437.

[20] B. Qin, R. H. Deng, Y. Li, S. Liu, Server-aided revocable identity-based encryption, in: Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I, Vol. 9326 of Lecture Notes in Computer Science, Springer, 2015, pp. 286–304.

[21] N. Attrapadung, H. Imai, Conjunctive broadcast and attribute-based encryption, in: Pairing-Based Cryptography - Pairing 2009, Third International Conference, Palo Alto, CA, USA, August 12-14, 2009, Proceedings, Vol. 5671 of Lecture Notes in Computer Science, Springer, 2009, pp. 248–265.

[22] Q. Li, H. Xiong, F. Zhang, Broadcast revocation scheme in composite-order bilinear group and its application to attribute-based encryption, IJSN 8 (1) (2013) 1–12.

[23] M. Horváth, Attribute-based encryption optimized for cloud computing, in: SOFSEM 2015: Theory and Practice of Computer Science - 41st International Conference on Current Trends in Theory and Practice of Computer Science, Pec pod Sněžkou, Czech Republic, January 24-29, 2015. Proceedings, Vol. 8939 of Lecture Notes in Computer Science, Springer, 2015, pp. 566–577.

[24] Y. Yang, X. Ding, H. Lu, Z. Wan, J. Zhou, Achieving revocable fine-grained cryptographic access control over cloud data, in: Information Security, 16th International Conference, ISC 2013, Dallas,

Texas, USA, November 13-15, 2013, Proceedings, Vol. 7807 of Lecture Notes in Computer Science, Springer, 2013, pp. 293–308.

[25] A. B. Lewko, B. Waters, Decentralizing attribute-based encryption, in: Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings, Vol. 6632 of Lecture Notes in Computer Science, Springer, 2011, pp. 568–588.

[26] D. Boneh, X. Boyen, H. Shacham, Short group signatures, in: Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings, Vol. 3152 of Lecture Notes in Computer Science, Springer, 2004, pp. 41–55.

[27] B. Waters, Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization, in: Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings, Vol. 6571 of Lecture Notes in Computer Science, Springer, 2011, pp. 53–70.

[28] A. Beimel, Secure schemes for secret sharing and key distribution, Ph.D. thesis, Israel Institute of Technology, Israel Institute of Technology (June 1996).

[29] T. E. Gamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Trans. Information Theory 31 (4) (1985) 469–472.

[30] J. H. Seo, K. Emura, Revocable identity-based encryption revisited: Security model and construction, in: Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings, Vol. 7778 of Lecture Notes in Computer Science, Springer, 2013, pp. 216–234.

[31] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, A. D. Rubin, Charm: a framework for rapidly prototyping cryptosystems, J. Cryptographic Engineering 3 (2) (2013) 111–128.

[32] M. Green, S. Hohenberger, B. Waters, Outsourcing the decryption of ABE ciphertexts, in: 20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings, USENIX Association, 2011.

[33] B. Schneier, Applied Cryptography (2Nd Ed.): Protocols, Algorithms, and Source Code in C, John Wiley & Sons, Inc., New York, NY, USA, 1995.

[34] M. Toorani, Smemail - A new protocol for the secure e-mail in mobile environments, CoRR abs/1002.3176.

[35] V. Buterin, Ethereum white paper: a next generation smart contract & decentralized application platformhttps://github.com/ethereum/wiki/wiki/White-Paper.