A Revocable Outsourcing Attribute-Based Encryption Scheme

Zoe L. Jiang[1], Ruoqing Zhang[2], Zechao Liu[1], S.M. Yiu[2], Lucas C.K. Hui[2(✉)], Xuan Wang[1], and Junbin Fang[3]

[1] Harbin Institute of Technology Shenzhen Graduate School, Harbin, China
[2] The University of Hong Kong, Pokfulam, Hong Kong
hui@cs.hku.hk
[3] Jinan University, Guangzhou, China

Abstract. Attribute-Based Encryption (ABE) is a generalized cryptographic primitive from normal public key encryption. It provides an access control mechanism over encrypted message using access policies and ascribed attributes. This scheme can solve the privacy issue when data is outsourced to cloud for storage well. However, there are some practical issues which must be fixed before ABE becomes applicable. One is that both the ciphertext size and the decryption time grows with the complexity of the access policy, which brings pressure to mobile devies. The other is that, from practical point of view, some users might be disabled for some attributes or be removed from the system. It demands on flexible revocation mechanism supporting both user and attribute granularities. In this research, we propose a solution adopting techniques on secure outsourcing of pairings to support outsourcing computation and adopting some techniques based on the tree-based scheme to solve user revocation and attribute revocation. We also give its security model and proof.

Keywords: Attribute-Based Encryption · Outsourced decryption · Revocation · Bilinear pairing

1 Introduction

Cloud computing offers the advantages of highly scalable and reliable storage on third-party servers. Its economical and efficient model typically results in an almost revolution of data storage ways. While going for cloud computing storage, the data owner and cloud servers are in two different domains. On one hand, cloud servers are not entitled to access the outsourced data content for data confidentiality; on the other hand, the data resources are not physically under the full control of data owner. Therefore, adopting expressive encryption and flexible authentication methods can balance the conflict between cloud users and servers.

R. Zhang—Co-first author.

J. Wan et al. (Eds.): CloudComp 2016, SPNCE 2016, LNICST 197, pp. 145–161, 2018.
https://doi.org/10.1007/978-3-319-69605-8_14

To address these issues, many techniques have been developed to ameliorate the situation, an important category was put forth called Attribute-Based Encryption (ABE), it could effectively bind the access-control policy to the data and the clients instead of having a server mediating access to files. The access control policy would be a policy that defines the kind of users who would have permissions to read the documents. The users or authenticated client are identified by attributes.

The first rudiment of this scheme is introduced by Sahai and waters in 2005, and then be divided into two formulations: Key-policy based ABE (KP-ABE) [1] and ciphertext-policy based (CP-ABE) [2]. There are several Attribute-Based Encryption proposed in literature. However, most of the existing ABE schemes are based on pairing based operations, the number of pairing computation to decrypt a ciphertext grow with the size and complexity of the access-control policy. In PC platform this issue should be able to handle normally, while it would be a significant challenge for using mobile phones or resource-constrained devices, limited battery life and users' appeal for fast process require high-efficiency and lightweight computation.

Outsourcing ABE scheme are developed to remedy this problem, we can give an overview of this concept [3]. In this concept, the general private key in previous ABE scheme is divided into two parts: a security key SK, held by the user and a transformation key TK, held by a proxy server. When a user want to obtain an encrypted message from a cloud center server, this file, namely defined as CT, is saved in proxy server firstly before sending to user. With the help of transformation key TK, proxy server translates this ABE ciphertext CT satisfied by that user's attributes or access policy into a simple ciphertext CT', thus the user can download this CT' from proxy server and it only incurs a small overhead for him to recover the message from the transformed ciphertext CT' by SK.

Outsourcing ABE has many attractive points compared with traditional ABE scheme, firstly, this scheme provide a efficient solution for File encrypting and decrypting on nowadays mobile devices platform, not only decreasing the operation time obviously, but also economize the limited memory and battery capacity of our phone and tablet. What's more, outsourcing ABE is based on secure bilinear pairing outsourcing algorithm, and it can guarantee an adversary has no power to access the plaintext both in cloud server and proxy server.

However, there still exists a critical issue before this scheme being able to deploy in practice: Revocation. For any cryptosystem that involve many users, if any of key compromising or user leaving situation happens, the corresponding user authority should be revoked from system. For ABE systems, there are two features about revocation issue, the first is about user revocation and the second is about attribute revocation. User revocation means revoking all the attributes of this user and let it remove from the system, attribute revocation stands that a user lost some authority but still exists in real situation. For example, if a employer is appointed to overseas for dealing with foreign affairs, his attribute related to working place will be changed to "overseas" from "local". Exactly in

practical the situation of revoking some attributes is more common than pure user level revocation.

Unfortunately, those authors of prime outsourcing ABE [3] scheme didn't consider revocation into their designing. There are also several verifiable Attribute-Based Encryption proposed in literature [4,5]. We revisit these schemes but find they didn't combine into revocation problem. In addition, user revocation has been well studied in previous work [6,7], while there still lacks breakthrough on attribute revocation. Therefore, the following questions arise naturally:

(1) Whether there exists a generic construction to introduce two revocation schemes to the outsourcing ABE?
(2) How to construct an outsourcing ABE with verifiable outsourced decryption and flexible revocation strategy?

We notice that in many previous research work about IBE revocation, most of them consider adopting tree structure to realize user identity distinguishing and revoking, The tree-based revocation approach is probably the most efficient one and it has been well studied in IBE scheme [8]. So for outsourcing ABE, adopting tree-based revocation is a prospective approach to meet those features above. But how to combine this structure into outsourcing ABE perfectly is still a no-easy problem to study.

1.1 Mainly Contribution

– *Supporting user revocation and attribute revocation simultaneously.* In our designing, we adopt tree based revocation scheme, one of most efficient methods, to realize a flexible revocation methods which support user revocation and attribute revocation.

1.2 Organization

The rest of this paper are organized as follows. The Sect. 2 includes some standard notations and definitions in cryptography. In Sect. 3, we review the relative architecture and security definition of outsourcing ABE. We present a new outsourcing ABE scheme supporting both user and attribute revocation in Sect. 4. Section 5 indicates the proof of relative security model. The last section concludes the paper.

2 Background

2.1 Access Structures

Definition 1 (Access Structure). Let $\{P_1, P_2, \cdots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \cdots, P_n\}}$ is monotone if $\forall B,C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then

$C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (resp. monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \cdots, P_n\}$, i.e. $\mathbb{A} \subseteq 2^{\{P_1, P_2, \cdots, P_n\}} \backslash \{\emptyset\}$. The sets in are called the authorized sets, and the sets not in $\mathbb{A}$ are called unauthorized sets.

For better understanding, we adopt the definition of [9]. Attributes plays the role of parties. Thus, the access structure $\mathbb{A}$ will contain the authorized sets of attributes, and we restrict our attention to monotone access structures. In this paper unless stated otherwise, by an access structure we mean a monotone access structure.

2.2 Bilinear Pairing

In the setting of bilinear pairings, we use the following symbols. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic additive groups. The order of $\mathbb{G}_1$ and $\mathbb{G}_2$ is a large prime and also denoted by the symbol q, Define $\mathbb{G}_T$ to be a cyclic multiplicative group of the same order q. A bilinear pairing is defined as a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ with the following properties:

1. Bilinear: $e(aR, bQ) = e(R, Q)^{ab}$ for any $R \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$, and $\alpha, \beta \in \mathbb{Z}_p$.
2. Non-degenerate: There are $R \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$ such that $e(aR, bQ) = e(R, Q)^{ab}$.
3. Computable: There is an efficient algorithm to compute $e(aR, bQ) = e(R, Q)^{ab}$ for any $R \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$.

The scheme we present in this paper are provably secure under the Decisional Parallel BDHE Assumption [10] in bilinear groups.

2.3 Ciphertext-Policy ABE

The classical CP-ABE encryption scheme can be described into 4 steps [10]:

- *Setup* $\rightarrow (pk, mk)$*:* The setup algorithm takes in as input a security parameter and provides a set of public parameters pk and the master key values mk.
- *KeyGen*$(w, mk) \rightarrow sk_w$*:* The KeyGen algorithm takes as input the master key values mk and the attribute set of the use w, to generate a secret key sk_w which confirms the users possession of all the attributes in w and no other external attribute.
  The above two algorithms being performed by the Trusted Authority. and the other two by the users:
- *Encryption*$(pk, m, \tau) \rightarrow C_\tau$*:* The Encryption algorithm is a randomized algorithm that takes as input the message m to be encrypted, the access structure τ which needs to be satisfied and the public parameters pk to output the ciphertext C_τ. It means that the access structure is embedded in the ciphertext such that only those users with attributes satisfying τ will be able to decrypt and retrieve the message.
- *Decryption*$(C_\tau, sk_w) \rightarrow m$*:* The decryption algorithm means that taking as input the ciphertext C_τ, the user secret keys sk_w can decrypt it.

A key module of attribute based encryption is access structure with special policy. An access control policy would be a policy that defines the kind of users who would have permissions to read the documents. e.g. In an academic setting, grade-sheets of a class may be accessible only to a professor handling the course and some teaching assistants (TAs) of that course. We can express such a policy in terms of a predicate:

$$(Prof \wedge CSdept.) \vee (student \wedge courseTA \wedge CSdept.)$$

The various credentials (or variables) of the predicate can be seen as attributes and the predicate itself which represents the access policy as the access-structure. In the above example here the access structure is quite simple. But in reality, access policies may be quite complex and may involve a large number of attributes [3].

3 Outsourcing ABE and Its Security

Outsourcing ABE is proposed in [3] and it aims to solve the computation problem in mobile devices.

3.1 Syntax of Outsourcing ABE

Let S represent a set of attributes, and $\mathbb{A}$ represent an access structure. In generally, we will define (I_{enc}, I_{key}), namely the inputs to the encryption and key generation function respectively. In a CP-ABE scheme we will have $(I_{enc}, I_{key}) = (\mathbb{A}, S)$, while in a KP-ABE scheme it means $(I_{enc}, I_{key}) = (S, \mathbb{A})$ Owing to CP-ABE is more generally than KP-ABE, in this paper we will consider CP-ABE as the main construction of outsourcing ABE scheme. A CP-ABE (resp. KP-ABE) scheme with outsourcing component consists of five algorithms:

- $Setup \rightarrow (\lambda, U)$*:* The setup algorithm takes security parameters and attribute universe description as input. It outputs the public parameters PK and the master key MK.
- $Encrypt \rightarrow (PK, M, I_{enc})$*:* The encryption algorithm takes as input the public parameters PK, a message M, and an access structure I_{enc}. It outputs the ciphertext CT.
- $KeyGen(MK, I_{key}) \rightarrow (SK, TK)$*:* The key generation algorithm takes as input the master key MK and an attribute set I_{key} and outputs a private key SK and a transformation key TK.
- $Transform(TK, CT) \rightarrow C'$*:* The ciphertext transformation algorithm takes as input a transformation key TK for I_{key} and ciphertext CT that was encrypted under I_{enc}. It outputs the partially decrypted ciphertext CT' if $S \in \mathbb{A}$ and the error symbol $\perp$ otherwise.
- $Decrypt_{out}(SK, CT')$*:* The decryption algorithm takes as input a private key SK for I_{key} and a partially decrypted ciphertext CT' that was originally encrypted under I_{enc}. It outputs the message M. If $S \in \mathbb{A}$ and the error symbol $\perp$ otherwise (Fig. 1).

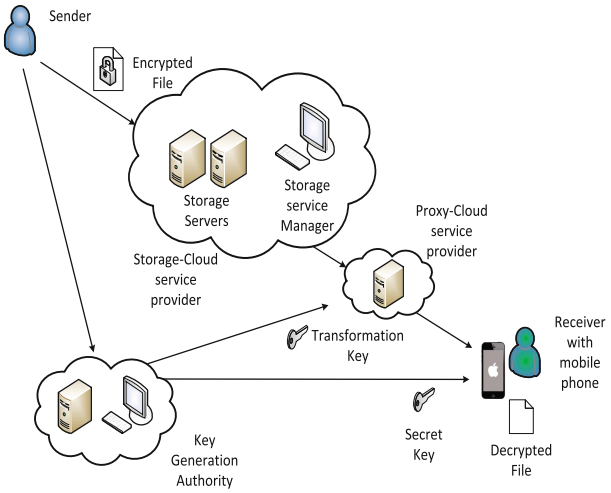

Fig. 1. Architecture of outsourcing ABE structure

3.2 Security of Outsourcing ABE

The conventional view of security against adaptive CCA (chosen-ciphertext attacks) is too rigorous due to it does not allow any bit of the cipher to be altered, Therefore we adopt a relaxation due to [11] called Replayable-CCA (RCCA) security. Which allows modification to the cipher provided they cannot change the underlying message in a meaningful approach. We can describe the RCCA security of Outsourcing ABE as a game between a challenger and an adversary. This game can be proceeds as follows:

- *Setup:* The challenger runs *Setup* algorithm to get the public parameters PK and a master secret key MSK, then gives the PK to the adversary. MSK is kept by himself.
- *QueryPhase*1*:* The challenger initializes an empty table T and an empty set D. The adversary adaptively issues queries:
  1. *Private key* query, on input a set of attributes S: The challenger runs $SK_S \leftarrow KeyGen(PK, MSK, S)$ and sets $D = D \bigcup S$. It then returns to the adversary the private key SK_S.
  2. *Transformation Key* query, on input a set of attributes S: The challenger searches the entry (S, SK_S, TK_S, RK_S) in table T. If such entry exists, it returns the transformation key TK_S. Otherwise, it runs $SK_S \leftarrow KeyGen(PK, MSK, S), (TK_S, RK_S) \leftarrow GenTK_{out}(PK, SK_S)$ and stores in table T the entry (S, SK_S, TK_S, PK_S). It then returns to the adversary the transformation key TK_S.
  3. *Decryption* query, on input a set of attributes S and a ciphertext CT: the challenger runs $SK_S \leftarrow KeyGen(PK, MSK, S)$ and $M \leftarrow Decrypt(PK, SK_S, CT)$. It then returns M to the adversary.

4. *Decrytpion$_{out}$* query, on input a set of attributes S and a pair of ciphertext (CT, CT') : The challenger searches the entry (S, SK_S, TK_S, PK_S) in table T. If such entry exists, it $M \leftarrow Decrypt_{out}(PK, CT, CT'RK_S)$ and returns to the adversary M; otherwise, it returns $\perp$.

– *Challenge:* The adversary submits two messages M_0,M_1 and an access structure $\mathbb{A}$, subject to the restriction that, for all $S \in D$, $\mathbb{A}$ cannot be satisfied by S. The challenger selects a random bit $\beta \in 0, 1$, sets $CT^* = Encrypt(PK, M_{/beta}, \mathbb{A})$ and sends CT^* to the adversary as its challenge ciphertext.
– *QueryPhase2:* The adversary continues to adaptively issue *Private key, Transformation key, Decryption* and *Decryption$_{out}$* queries, as in Query phase 1, but with the restrictions that the adversary cannot
  1. issue a *Private key* query that would result in a set of attributes S which satisfies the access structure $\mathbb{A}$ being added to D.
  2. issue a trivial decryption query. That is, *Decryption* and *Decryption$_{out}$* queries will be answered as in Query phase 1, except that if the response would be either M_0 or M_1, then the challenger responds with the error symbol $\perp$.
– *Guess.* The adversary $\mathbb{A}$ outputs its guess $\beta^{'} \in 0, 1$ for β and wins the game if $\beta = \beta^{'}$.

The advantage of the adversary in this game is defined as $|Pr[\beta = \beta^{'}] - \frac{1}{2}|$ where the probability is taken over the random bits used by the challenger and the adversary.

3.3 Revocation of Outsourcing ABE

– *User revocation and Attribute revocation.* User revocation have been taken notice in many research work [12,13] and it has many outcomes in some research work relative to IBE scheme [8,14,15]. In ABE, user revocation is also very important; user revocation means revoking all the attributes of this user and let it remove from the system, attribute revocation means a user lose some authority but still exists in real situation. Exactly, in practical the situation of revoking some attribute of a user is more common than pure user level revocation. Several ABE scheme have been support attribute granularity revocation [2,16,17], while they just adopt primary time-rekeying mechanism and cannot be compatible with user revocation.
– *Direct revocation and Indirect revocation.* From another perspective, The revocation issue can been seen two subsets: direct and indirect revocation mechanism, Imai proposed a direct revocation mechanism [18] and defined a revocation list in their scheme, to announce who can obtain the message directly during encryption. Sahai proposed an indirect revocation mechanism [19], in their designing an authority is needed to broadcast key-update notification periodically so that those revoked user cannot continue to update their user keys. Thus it achieves the user revoking goals. Direct revocation

enforces revocation directly by the sender who directly specifies a list of revocation when encrypting. Indirect revocation implements revocation by the key authority who releases a key update material periodically in such a way that only those non-revoked users are able to update their private keys. Obviously, An advantage o the indirect method over the direct one is that it does not require senders to know the revocation list in advance and hence reducing the workloads of senders. In contrast, an convenience of the direct method over the other is that it does not involve key update phase for all non-revoked users interacting with the key authority. So in our designing, we adopt indirection because it can reduce the workload of senders and communication between trust authority and users, but not based on the approach of time period updating.

– *Backward security and Forward security.* In traditional ABE schemes, backward security means that any user who comes to hold an attribute(that satisfies the access policy) should be prevented from accessing the plaintext of the previous data exchanged before he holds the attribute. In addition, forward security means that any user who drops an attribute should be prevented from accessing the plaintext of the subsequent data exchanged after he drops the attribute, unless the other valid attributes that he is holding satisfy the access policy.

4 Efficient Verifiable Outsourcing ABE Revocation Scheme

In this section, we provide the construction of verifiable outsourcing ABE. Our scheme are based on the tree-based revocation, due to Boldyreva, Goyal, and Kumar [8] which is the most efficient one. In our outsourcing ABE scheme, we split the decryption key in two components corresponding to transformation and final-decryption, that we call transformation key and retrieving key respectively.

Let $U = \{u_1, u_2, \cdots, u_n\}$ represent the universe of users and define $L = \{\lambda_1, \lambda_2, \cdots, \lambda_l\}$ as the attributes universe in the system, Let $G_i \subset U$ be a set of users that hold the attribute λ_i, we define G_i as an attribute group, it will be used as a user revocation list to λ_i, Let $\mathcal{G} = \{G_1, G_2, \cdots, G_l\}$ be the universe of such attribute groups. Let K_{λ_i} be the attribute group key that is shared among those nonrevoked users in $G_i \in \mathcal{G}$.

A outsourcing CP-ABE scheme with efficient revocation scheme consists of the following eight algorithms:

– $Setup(\lambda, L) \rightarrow (PK, MK)$*:* The setup function runs in the key generation authority. This algorithm takes as input a security parameter λ and attribute universe descryption L, then outputs the public parameters PK and a master key MK.
– $AttributeKeyGen(MK, w, \tau) \rightarrow (SK, TK)$*:* The attribute key generation function runs in the key generation authority. This algorithm takes as input the master key MK, a set of attributes w, and a set of user τ, then outputs

a set of private attribute keys SK and transformation key TK for each user in w that identifies with the attribute set.

- $KEKGen(\tau) \rightarrow (KEKs)$*:* This key encryption key (KEK) function runs in the storage service manager. This algorithm takes as input a set of user indices $\tau \subseteq U$, and outputs KEKs for each user in τ, which will be used to encrypt attribute group keys K_{λ_i} for each $G_i \in \mathcal{G}$.
- $Encrypt(PK, \mathcal{M}, \mathbb{A}) \rightarrow (CT)$*:* It takes as input the public parameters PK, a message $\mathcal{M}$ and an access structure $\mathbb{A}$. It outputs a ciphertext CT.
- $ReEncrypt(CT, G) \rightarrow (CT')$*:* This re-encryption algorithm runs in the storage service manager. This algorithm takes as input the ciphertext CT including an access structure $\mathbb{A}$, and a set of attribute groups G. If the attribute groups emerge in $\mathbb{A}$, it re-encrypts CT for the attributes; else, return $\perp$. It outputs a re-encrypted ciphertext CT' such that only a user who possesses a set of attributes that satisfies the access structure and has a valid membership for each of them simultaneously can decrypt this message.
- $Transform_{out}(TK, CT') \rightarrow (CT'_{pro})$*:* This transformation algorithm runs in the proxy cloud. It takes as input the ciphertext CT' and a transformation key TK. It outputs a partially decrypted ciphertext CT'_{pro}.
- $Decrypt(SK, CT'_{pro}) \rightarrow (M)$*:* When the receiver download the CT'_{pro} from proxy server. It takes as input a private key SK, a partially decrypted ciphertext CT'_{pro} and outputs a message M.

4.1 Scheme Construction

Our outsourcing scheme is based on [20]. To enable outsourcing we modify the KeyGen algorithm to output a transformation key. We define a new algorithm and modify the decryption algorithm to handle outputs of Encrypt as well as Transform.

Let $\mathbb{G}$ be a bilinear group of prime order p, and let g be a generator of $\mathbb{G}$, Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ denote the bilinear map. A security parameter λ will determine the size of the groups.

- *Setup.* The setup algorithm chooses a group $\mathbb{G}$ of prime order p and a generator g. In addition, it chooses random exponents $\alpha, \beta \in \mathbb{Z}_p^*$. In addition, we will adopt hash functions: $H : \{0,1\}^* \rightarrow \mathbb{G}$, $H_1 : \{0,1\}^* \rightarrow Z_p^*$ and H_2: $\{0,1\}^* \rightarrow \{0,1\}^k$. The public parameters is published as

$$PK = (g, e(g,g)^{\alpha}, h = g^{\beta}, H, H_1, H_2)$$

  Then the authority sets $MK = (\beta, g^{\alpha}, PK)$ as the master secret key.
  Next we will divide the *Key Generation* phase in traditional outsourcing ABE into two parts. In our scheme, it consists of *Attribute Key Generation* by the trusted authority and *KEK Generation* by the storage service manager.
- $AttributeKeyGen(MK, w, \tau)$. After setting up the system public and secret parameters, the trusted authority generates attribute keys for a set of users U by running $AttributeKeyGen(MK, w, \tau)$ algorithm, namely that taking a

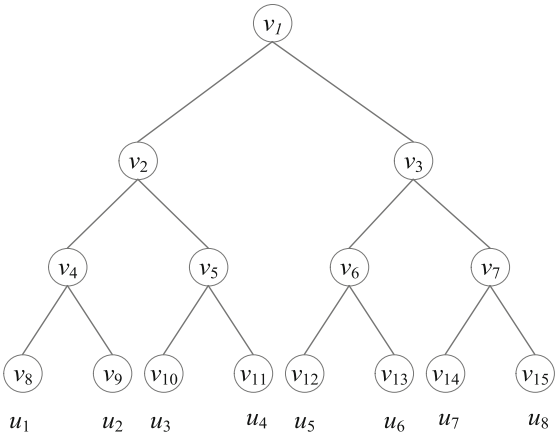

Fig. 2. KEK tree for attribute group key distribution

set of attributes $w \subseteq L$ and a set of user $\tau \subseteq U$ as inputs and outputs an attribute key for each user that identifies with that set τ.

For better understanding, We give an example based on business corporation scenes, assume that there are three staff members u_1, u_2, u_3 in the R&D Department, expressed as a_3, then we defines a_1 represents the vice heads title of this department and a_2 stands for those members who focus on daily administrative obligation. So u_1, u_2, u_3 are associated with $\{\lambda_1, \lambda_2, \lambda_3\}, \{\lambda_2, \lambda_3\}, \{\lambda_1, \lambda_3\}$ respectively, so the trust authority will have the attribute group list $G_1 = \{u_1, u_3\}, G_2 = \{u_1, u_2\}, G_3 = \{u_1, u_2, u_3\}$.

The algorithm first chooses a random $r \in \mathbb{Z}_p^*$(which is unique to each user), and random $r_j \in \mathbb{Z}_p^*$ for each attribute $\lambda_j \in w$. It creates a $SK' = (\bar{D} = g^{(\alpha+r)/\beta}, \{\bar{D}_j = g^r \cdot H(\lambda_j)^{r_j}, \bar{D}_j' = g^{r_j}\}_{\lambda_j \in w})$.

Then it chooses a random value $z \in \mathbb{Z}_p^*$ and sets the transformation key $TK = (PK, D = \bar{D}^{1/z}, \{D_j = \bar{D_j}^{1/z}, D_j' = \bar{D_j'}^{1/z}\}_{\lambda_j \in w})$. The authority finally sets the private key SK as (z, TK).

– *KEK Generation.* The storage service manager runs the *KEKGen(U)* and generates *KEKs* for users in U. The storage services manager sets a binary KEK tree for the universe of users U as in Fig. 2. In the tree, each node v_j of the tree holds a *KEK*, denoted by KEK_j. We use *path keys* to define a set of *KEKs* on the path nodes from a leaf to the root.

The storage service manager constructs the KEK tree as follows:

1. Every member in U is assigned to the leaf nodes of the tree. The storage service manager generates random keys and assigns them to each leaf node and internal node.
2. Each member $u_i \in \tau$ receives that path keys PK_i from its leaf node to the root node of the tree securely. For instance, u_2 stores $PK_2 = \{KEK_9, KEK_4, KEK_2, KEK_1\}$ as its path keys in Fig. 2.

Then the path keys will be used as *KEKs* to encrypt the attribute group keys by the storage service manager in the data re-encryption phase.

– *Encryption*$(PK, \mathcal{M}, \mathcal{T})$. The encryption algorithm takes as input the public parameters PK, a message $\mathcal{M}$, and the tree access structure $\mathcal{T}$. This algorithm chooses a polynomial q_x for each node x in the tree $\mathcal{T}$. These polynomials are chosen in a top-down manner, starting from the root node R. For each node x in the tree, the algorithm sets the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node. For root node R, it chooses random $s \in \mathbb{Z}_p^*$ and sets $q_R(0) = s$, then chooses d_R others points of polynomial q_R randomly. For any other node x, it sets $q_x(0) = q_{p(x)}(index(x))$, while $p(x)$ represents the parent of node x in the tree. Let Y be the set of leaf nodes of access tree $\mathcal{T}$.
This algorithm selects a random $R \in \mathbb{G}_T$ and then computes $s = H_1(R, \mathcal{M})$ and $r = H_2(R)$. The ciphertext is published as $CT =$

$$(\mathcal{T}, C = \mathcal{R} \cdot e(g,g)^{\alpha s}, C' = h^s, C'' = \mathcal{M} \oplus r,$$
$$\forall y \in Y : C_y = g^{q_y(0)}, C_y^{'} = H(\lambda_y)^{q_y(0)})$$

– *Re–Encryption*(CT, G). Before receiver getting the encrypted data CT, the storage service manager re-encrypts the ciphertext using a set of the members information for each attribute group G that appears in the access tree embedded in CT, to enforce user-level access control per each attribute group on top of the ciphertext. This algorithm runs as follows:

1. For all $G_y \in G$, chooses a random $K_{\lambda_y} \in \mathbb{Z}_p^*$, then re-encrypts CT and generates $CT' =$

$$(\mathcal{T}, C = \mathcal{R} \cdot e(g,g)^{\alpha s}, C' = h^s, C'' = \mathcal{M} \oplus r,$$
$$\forall y \in Y : C_y = g^{q_y(0)}, C_y^{'} = (H(\lambda_y)^{q_y(0)})^{K_{\lambda_y}})$$

2. Selects root nodes of the minimum cover sets in the KEK tree that can cover all of the leaf nodes associated with users in G_i, for all $G_i \in G$. We denote a set of KEKs that this collection covers all users in G_i, e.g., if $G_i = u_1, u_2, u_3, u_4, u_7, u_8$ in Fig. 2, then $KEK(G_i) = \{KEK_2, KEK_7\}$, owing that v_2 and v_7 are the root nodes of the minimum cover sets that can cover all the members in G_i. Notes that this collection covers all users in G_i and only them, and any user $u \notin G_i$ can by no means know any KEK in $KEK(Gi)$.
3. Generate a header message $Hdr = (\forall i \in [1, l] : \{E_K(K_{\lambda_i})\}_{K \in KEK(G_i)})$

Once receiving a query from a user, the storage service manager responds with (Hdr, CT') to the user. It is necessary to declare that the attribute group key distribution protocol throuth Hdr is a stateless approach. Thus, even if users cannot update their key state frequently in practical applications, they will be able to decrypt the attribute group key from Hdr at any time they receive it, as long as they are not revoked from any of the attribute groups and authorized to decrypt it.

– $Transformation(TK, CT')$. The transformation algorithm takes as input a transformation key TK and the ciphertext CT'. The whole phase can be

divided into two parts: *attribute group key decrypt* and *message decrypt*.
Attribute group key decrypt. When proxy cloud server receives the ciphertext (Hdr, CT') from the data service manager, he first obtains the attribute group keys for all attributes in w that the user holds from Hdr. If a user u_t has a valid attribute λ_i(means $u_t \in G_i$), he can decrypt the attribute group key K_{λ_i} from Hdr using a KEK that is common in $KEK(G_i)$ and PK_t (that is, KEK$\in KEK(G_i) \cap PK_t$). Note that there can be only one such KEK, so the user may belong to at most one subset rooted by one KEK in $KEK(G_i)$. For example, if $G_i = \{u_1, u_3\}$ in the example above. u_1 can decrypt the K_{λ_i} using the path key $KEK_2 \in PK_3$. Then proxy cloud server updates the transform key with attribute group keys as follows: $TK = (PK, D, \{D_j, D_j^{''} = (D_j^{'})^{1/K_{\lambda_j}}\}_{\lambda_j \in w})$. Note that any user $u \notin G_j$ can by no means decrypt K_{λ_j}.
Message decrypt. We first define a recursive algorithm $DecryptNode$ (CT, SK, x) which takes as input ciphertext CT, a private key SK and a node x from the tree $\mathcal{T}$. If x is a leaf node, then $DecryptNode(CT, SK, x)$:

$$:= \frac{e(D_x, C_x)}{e(D_x^{''}, C_x^{'})} = \frac{e(g^{r/z} H(\lambda_x)^{r_x/z}, g^{q_x(0)})}{e(g^{r_x/(z \cdot K_{\lambda_x})}, H(\lambda_x)^{q_x(0) \cdot K_{\lambda_x}})} = e(g,g)^{r \cdot q_x(0)/z}$$

We now consider the recursive case when x is a nonleaf node. For all nodes z that are children of x, it calls $DecryptNode(CT, SK, z)$ and stores the output as F_z. Let S_x be an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$. If no such set exists, then the node was not satisfied and the function returns $\perp$. Otherwise, we compute F_x:

$$\begin{aligned}
&= \prod_{z \in S_x} F_x^{\triangle_{i,S_x'}(0)} \\
&= \prod_{z \in S_x} (e(g,g)^{r \cdot q_z(0)/z})^{\triangle_{i,S_x'}(0)} \\
&= \prod_{z \in S_x} (e(g,g)^{r \cdot q_{p(z)}(0)/z})^{\triangle_{i,S_x'}(0)} \\
&= \prod_{z \in S_x} (e(g,g)^{r \cdot q_x(i)/z) \cdot \triangle_{i,S_x'}(0)} \\
&= e(g,g)^{r \cdot q_x(0)/z}
\end{aligned}$$

For the root node R of the access tree, we observe that $DecryptNode(CT, SK, R) = e(g,g)^{rs/z}$ if the tree $\mathcal{T}$ is satisfied. Then we compute $e(C^{'}, D)/DecryptNode(CT, SK, R) = e(h^s, g^{(\alpha+r)/(\beta z)})/e(g,g)^{rs/z} = e(g,g)^{\alpha s/z}$. It finally outputs the partially decrypted ciphertext CT'_{pro} as $(C, C'', e(g,g)^{\alpha s/z})$,

– $Decryption(SK, CT'_{pro})$. The decryption algorithm takes as input a private key $SK = (z, TK)$ and a ciphertext CT'_{pro}. If the ciphertext is not partially decrypted, then the algorithm first executes $Transformation(TK, CT')$. If the outputs is $\perp$, then this algorithm outputs $\perp$ as well. Otherwise, it takes the ciphertext (T_0, T_1, T_2) and computes $R = T_0/T_2^z$, $\mathcal{M} = T_1 \oplus H_2(R)$, and $s = H_1(R, \mathcal{M})$. If $T_0 = R \cdot e(g,g)^{\alpha s}$ and $T_2 = e(g,g)^{\alpha s/z}$, it outputs $\mathcal{M}$; otherwise, it outputs $\perp$.

4.2 Key Update

When trusted authority revokes an attribute from a user, it can be seen as sending a leave request for some attribute. On receipt of the membership change request for some attributes groups, the trusted authority notifies the data service manager of the event and sends the updated membership list of the attribute group to it. When the data service manager receives the membership change notification from the trusted authority, it changes the attribute group key for the attribute which is affected by the membership change. Without loss of generality, suppose a user drop attribute a_1, Then the key update procedure progresses as follows:

1. The data service manager selects a random $s' \in \mathbb{Z}_p$, and a K'_{λ_i} which is different from the previous attribute group key K_{λ_i}. Then it re-encrypts the ciphertext using the public parameters PK as $CT^{'} =$

$$(\mathcal{T}, C = R \cdot e(g,g)^{\alpha(s+s')}, C' = h^{s+s'} = g^{\beta(s+s')}, C'' = \mathcal{M} \oplus r,$$
$$C_i = g^{q_i(0)+s'}, C_i^{'} = (H(\lambda_i)^{q_i(0)+s'})^{K_{\lambda_i}^{'}},$$
$$\forall y \in Y \setminus \{i\} : C_y = g^{q_y(0)+s'}, C_y^{'} = (H(\lambda_y)^{q_y(0)+s'})^{K_{\lambda_y}}).$$

   For the other attribute groups that are not affected by the membership changes, the attribute group keys do not necessarily need to be updated.
2. The data service manager selects new minimum cover sets for G_i excluding a leaving user who comes to drop an attribute. Then it generates a new header message with the updated $KEK(G_i)$ as follows.
   $Hdr = (\{E_K(K'_{\lambda_i})\}_{K \in KEK(G_i)}, \forall y \in Y \setminus \{i\} : \{E_K(K_{\lambda_y})\}_{K \in KEK(G_y)})$
   When a user sends a request query for the outsourced data afterward, the data service manager responds with the above *Hdr* and ciphertext *CT'* encrypted under the updated keys.

5 Security Proof of Revocation Scheme

Theorem 1. *Suppose the scheme of Waters [2] is a CPA-secure CP-ABE scheme. Then our revocable outsourcing ciphertext policy ABE scheme is selectively RCCA-secure.*

Proof. Suppose there exists a polynomial-time adversary $\mathcal{A}$ who can attack our scheme in the selective RCCA-security model for outsourcing with advantage ϵ. Therefore, we build a simulator $\mathcal{B}$ that can attack the Waters scheme [2] in the CPA-secure model with advantage ϵ minus a negligible amount.

Init. The simulator $\mathcal{B}$ runs $\mathcal{A}$. $\mathcal{A}$ chooses the challenge access structure $\mathcal{T}$ to $\mathcal{B}$. Then $\mathcal{B}$ sends it to the Waters challenger.

Setup. The simulator $\mathcal{B}$ obtains the Waters public parameters $PK = (g, e(g,g)^{\alpha}, g^{\beta}, H)$, in which H is a description of the Hash Function. Next $\mathcal{B}$

chooses other two hash functions: H_1: $\{0,1\}^* \rightarrow \mathbb{Z}_p^*$ and H_2: $\{0,1\}^* \rightarrow \{0,1\}^k$. Finally $\mathcal{B}$ sends $PK^{'} = (g, e(g,g)^{\alpha}, g^{\beta}, H, H_1, H_2)$ to the adversary $\mathcal{A}$ as the public parameters.

Phase 1. The simulator $\mathcal{B}$ initialized three empty tables T, T_1, T_2,an empty set D and an integer $j = 0$. Then it answers the adversary's queries as follows:

- Random Oracle Hash $H_1(R, M)$: If there is an entry (R, M, s) in T_1, return s. Otherwise, choose a random $s \in \mathbb{Z}_p^*$, then record $(R, \mathcal{M}, s)$ in T_1 and return s.
- Random Oracle Hash $H_2(R)$: If there is an entry (R, r) in T_2, return r. Otherwise, choose a random $r \in \{0,1\}^k$, then record (R, r) in T_2 and return r.
- Create(S): $\mathcal{B}$ sets $j := j + 1$. Then it excutes one of the two ways:
  1. If attributes set S satisfies $\mathcal{T}$, then simulator $\mathcal{B}$ chooses a fake transformation key pair as follows: choose random $a, r^{'} \in {\mathbb{Z}_p}^*$ and run *Attribute KeyGen*(a, r', PK, S) to Obtain $SK' = (PK, \bar{D} = g^{(a+r')/\beta}, \{\bar{D}_i = g^{r^{'}} H(\lambda_i)^{r_i}, \bar{D_i^{'}} = g^{r_i}\}_{\lambda_i \in S})$. Let $a = \alpha/z$ and $r^{'} = r/z$, then we replace a and r. We have $TK = SK' =$

$$(PK, \bar{D} = g^{(\alpha+r)/(\beta z)}, \{\bar{D}_i = g^{r/z} H(\lambda_i)^{r_i}, \bar{D_i^{'}} = g^{r_i}\}_{\lambda_i \in S})$$
$$= (PK, D = (g^{\frac{\alpha+r}{\beta}})^{1/z}, \{D_i = (g^{r} H(\lambda_i)^{r_i^{'}})^{1/z}, D_i^{'} = (g^{r_i^{'}})^{1/z}\}_{\lambda_i \in S})$$

     Note that we implicitly set $r_i = r_i^{'}/z$. Then the TK is properly distributed.
  2. Otherwise, simulator $\mathcal{B}$ calls the Waters key generation oracle on S to obtain the key $SK' = (PK, D, \{D_j, D_j^{'}\}_{\forall j \in S})$. Next $\mathcal{B}$ chooses random $z \in \mathbb{Z}_p^*$, then it sets $SK = z$ and $TK = (PK, D^{1/z}, \{{D_j}^{1/z}, (D_j^{'})^{1/z})$. Finally, simulator $\mathcal{B}$ gets (j, S, SK, TK) in table T and return TK to $\mathcal{A}$.
- *Corrupt*(i): If there exists an ith entry in table T, then the simulator $\mathcal{B}$ obtains the entry (j, S, SK, TK) and sets $D := D \bigcup S$. It then return SK to $\mathcal{A}$, or returns $\perp$ if there is no such entry existing.
- *Decrypt*(i, CT): Without loss of generality, let $CT = (C_0, C_1, C_2)$ be associated with an access structure $\mathcal{T}$. Obtain the record (i, S, SK, TK) from table T. If it is not there or $S \notin \mathcal{T}$, return $\perp$ to $\mathbb{A}$.

1. If the i^{th} entry (j, S, SK, TK) does not satisfy the challenge structure $\mathcal{T}$, it proceeds as follows:
   (a) Compute $R = C_0/C_2$, parse $SK = (z, TK)$.
   (b) Obtain the records $(R, \mathcal{M}_i, s_i)$ from table T_1. If none exist, return $\perp$ to $\mathcal{A}$.
   (c) If there exists indices $i_1 \neq i_2$ such that $(R, \mathcal{M}_{i_1}, s_{i_1})$ and $(R, \mathcal{M}_{i_2}, s_{i_2})$ are in table T_1, $\mathcal{M}_{i_1} \neq \mathcal{M}_{i_2}$ and $s_{i_1} = s_{i_2}$, then $\mathcal{B}$ aborts the simulation.
   (d) Contrarily, obtain the records (R, r) from table T_2. If there no such record exists, the simulator $\mathcal{B}$ outputs $\perp$.
   (e) Test if $C_0 = R \cdot e(g,g)^{s_i, \alpha}, C_1 = \mathcal{M}_{i_1} \oplus r$ and $C_2 = e(g,g)^{s_i \alpha/z}$ for each i in the records.
   (f) If there is an i that passes the above test, output $\mathcal{M}_i$; otherwise return $\perp$.

2. If key i does satisfy the challenge structure $\mathcal{T}$, proceed as follows:
   (a) Compute $\gamma = C_2^{1/d}$, parse $SK = (d, TK)$.
   (b) For each record $(R, \mathcal{M}_i, s_i)$ in table T_1, test if $\gamma = e(g,g)^{s_i}$.
   (c) If there is no match, $\mathcal{B}$ return $\perp$.
   (d) If there is more than one matches, $\mathcal{B}$ aborts the simulation.
   (e) Contrarily, let $(R, \mathcal{M}, s)$ be the sole match. Obtain the record (R, r) from the T_2. If the record does not exist, the simulator $\mathcal{B}$ outputs $\perp$.
   (f) Test if $C_0 = R \cdot e(g,g)^{s\alpha}$, $C_1 = \mathcal{M} \oplus r$ and $C_2 = e(g,g)^{ds}$ for each i in the records, If all tests pass, return $\mathcal{M}$; else return $\perp$.

Challenge. Ultimately, the adversary $\mathcal{A}$ submits a message pair $(\mathcal{M}_0^*, \mathcal{M}_1^*) \in \{0,1\}^{2k}$, the simulator $\mathcal{B}$ operates as follows.

- The simulator $\mathcal{B}$ chooses random "messages" $(\mathcal{R}_\prime, \mathcal{R}_\infty) \in \mathbb{G}_T^2$ and passes them on to the Waters challenger to obtain a ciphertext $CT = (C, C', \{C_i, C_i^{'}\}_{i \in S})$ under $\mathcal{T}$.
- $\mathcal{B}$ chooses a random value $C^{''} \in \{0,1\}^k$.
- Then $\mathcal{B}$ sends the challenge ciphertext $CT^* = (C, C', C^{''}, \{C_i, C_i^{'}\}_{i \in S})$ to the adversary $\mathcal{A}$.

Phase 2. The simulator $\mathcal{B}$ continues to answer queries as in Phase 1, except that if the response to a Decrypt query would be either $\mathcal{M}_0^*$ or $\mathcal{M}_1^*$, then $\mathcal{B}$ responds with the message **test**.

Guess. Ultimately, the adversary $\mathcal{A}$ must either output a bit or abort, either way $\mathcal{B}$ ignores it. Next, $\mathcal{B}$ searches through tables T_1 and T_2 to see if the values $\mathcal{R}_0$ or $\mathcal{R}_1$ appear as the first element of any entry (i.e., that $\mathcal{A}$ issued a query of the form $H_1(\mathcal{R}_i)$ or $H_2(\mathcal{R}_i)$). If neither of both $\mathcal{R}_0$ and $\mathcal{R}_1$ are revealed, $\mathcal{B}$ outputs a random bit as its guess.

The simulator $\mathcal{B}$ in Game is obviously negligible, and Theorem 1 is RCCA secure within this negligible advantage. So the proof of Theorem 1 is complete.

6 Conclusion

In this paper, we considered a new requirement of ABE with outsourced decryption: revocation. We modified the original model of ABE with outsourced decryption to include revocation. We prove our revocation support RCCA security level. As expected, the scheme substantially reduced the computation time required for mobile phones to recover plaintexts.

Acknowledgement. This work is supported in part by National High Technology Research and Development Program of China (No. 2015AA016008).

References

1. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 89–98. ACM (2006)

2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, SP 2007, pp. 321–334. IEEE (2007)
3. Green, M., Hohenberger, S., Waters, B.: Outsourcing the decryption of ABE ciphertexts. In: USENIX Security Symposium, vol. 2011 (2011)
4. Qin, B., Deng, R.H., Liu, S., Ma, S.: Attribute-based encryption with efficient verifiable outsourced decryption. IEEE Trans. Inf. Forensics Secur. **10**(7), 1384–1393 (2015)
5. Li, J., Huang, X., Li, J., Chen, X., Xiang, Y.: Securely outsourcing attribute-based encryption with checkability. IEEE Trans. Parallel Distrib. Syst. **25**(8), 2201–2210 (2014)
6. Chen, Y., Jiang, Z.L., Yiu, S.M., Liu, J.K., Au, M.H., Wang, X.: Fully secure ciphertext-policy attribute based encryption with security mediator. In: Hui, L.C.K., Qing, S.H., Shi, E., Yiu, S.M. (eds.) ICICS 2014. LNCS, vol. 8958, pp. 274–289. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21966-0_20
7. Lueks, W., Alpár, G., Hoepman, J.-H., Vullers, P.: Fast revocation of attribute-based credentials for both users and verifiers. In: Federrath, H., Gollmann, D. (eds.) SEC 2015. IAICT, vol. 455, pp. 463–478. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18467-8_31
8. Boldyreva, A., Goyal, V., Kumar, V.: Identity-based encryption with efficient revocation. In: Proceedings of the 15th ACM Conference on Computer and Communications Security, pp. 417–426. ACM (2008)
9. Beimel, A.: Secure schemes for secret sharing and key distribution. Technion Israel Institute of Technology, Faculty of Computer Science (1996)
10. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19379-8_4
11. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 565–582. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_33
12. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 195–203. ACM (2007)
13. Staddon, J., Golle, P., Gagné, M., Rasmussen, P.: A content-driven access control system. In: Proceedings of the 7th Symposium on Identity and Trust on the Internet, pp. 26–35. ACM (2008)
14. Li, J., Li, J., Chen, X., Jia, C., Lou, W.: Identity-based encryption with outsourced revocation in cloud computing. IEEE Trans. Comput. **64**(2), 425–437 (2015)
15. Qin, B., Deng, R.H., Li, Y., Liu, S.: Server-aided revocable identity-based encryption. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9326, pp. 286–304. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24174-6_15
16. Yu, S., Wang, C., Ren, K., Lou, W.: Attribute based data sharing with attribute revocation. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, pp. 261–270. ACM (2010)
17. Pirretti, M., Traynor, P., McDaniel, P., Waters, B.: Secure attribute-based systems. J. Comput. Secur. **18**(5), 799–837 (2010)
18. Attrapadung, N., Imai, H.: Conjunctive broadcast and attribute-based encryption. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 248–265. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03298-1_16

19. Sahai, A., Seyalioglu, H., Waters, B.: Dynamic credentials and ciphertext delegation for attribute-based encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 199–217. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_13
20. Hur, J., Noh, D.K.: Attribute-based access control with efficient revocation in data outsourcing systems. IEEE Trans. Parallel Distrib. Syst. **22**(7), 1214–1221 (2011)