2012 32nd IEEE International Conference on Distributed Computing Systems

Attributed-based Access Control for Multi-Authority Systems in Cloud Storage

Kan Yang
Department of Computer Science
City University of Hong Kong
Hong Kong SAR
Email: kanyang3@student.cityu.edu.hk

Xiaohua Jia
Department of Computer Science
City University of Hong Kong
Hong Kong SAR
Email: csjia@cityu.edu.hk

***Abstract*—Ciphertext-Policy Attribute-base Encryption (CP-ABE) is regarded as one of the most suitable technologies for data access control in cloud storage. In almost all existing CP-ABE schemes, it is assumed that there is only one authority in the system responsible for issuing attributes to the users. However, in many applications, there are multiple authorities co-exist in a system and each authority is able to issue attributes independently. In this paper, we design an access control framework for multi-authority systems and propose an efficient and secure multi-authority access control scheme for cloud storage. We first design an efficient multi-authority CP-ABE scheme that does not require a global authority and can support any LSSS access structure. Then, we prove its security in the random oracle model. We also propose a new technique to solve the attribute revocation problem in multi-authority CP-ABE systems. The analysis and simulation results show that our multi-authority access control scheme is scalable and efficient.**

***Index Terms*—Access Control, Multi-Authority, CP-ABE, Attribute Revocation, Cloud Storage**

I. Introduction

Cloud storage is an important service of cloud computing [1], which offers services for data owners to host their data in the cloud. This new paradigm of data hosting and data access services introduces a great challenge to the data access control. Because the cloud server may give data access to the users who do not have the access permission for profit gain, the data owners can no longer trust the cloud servers and rely on them to do data access control. Ciphertext-Policy Attribute-based Encryption (CP-ABE) [2], [3] is regarded as one of the most suitable technologies for data access control in cloud storage systems, because it gives the data owner more direct control on access policies. In CP-ABE schemes, the access policy checking is implicitly conducted inside the cryptography. That is, there is no one to explicitly evaluate the policies and make decisions on whether allows the user to access the data. In CP-ABE scheme, there is an authority that is responsible for attribute management and key distribution. (The authority can be the registration office in a university, the human resource department in a company or the government education organization, etc.) The data owner defines the access policies and encrypts data according to the policies. Each user will be issued a secret key that reflects its attributes. A user can decrypt the data only when its attributes satisfy the access policies.

In most existing CP-ABE schemes [2]–[6], there is only one authority in the system and all the public keys and secret keys are issued by this authority. However, in many applications, there are multiple authorities, each of which manages the attributes under its own domain independently. A user can hold the attributes issued by different authorities. For instance, a data owner may want to share medical data only with a user who has the attribute of ”Doctor” issued by a medical organization and the attribute ”Medical Researcher” issued by the administrator of a clinical trial. Another scenario is that two companies such as IBM or Google may have a joint project and both of them issue attributes to users who participate this joint project. The existing CP-ABE schemes with single authority cannot be applied to data access control for multi-authority systems, because no authority is able to verify attributes across different organizations and to issue secret keys to all the users in the system. The aim of this paper is to study the multi-authority access control issue in cloud storage.

There are two challenging issues in the design of multi-authority access control schemes for cloud storage systems. The first issue is the problem of *Collusion*. Multiple users holding attributes from different authorities may collude together to obtain illegal access to the data. Existing multi-authority CP-ABE schemes [7], [8] usually rely on a global authority to collect and verify users' attributes and generate the secret keys for them. With the global authority, the collusion problem can be solved by using the key randomization mechanism as in the single authority schemes. However, the global authority is too powerful and it becomes a vulnerable point for security attacks and the performance bottleneck for large scale systems. Some multi-authority CP-ABE schemes [9], [10] are proposed to remove the global authority, but they still lack of scalability or efficiency. The other issue is the difficulty of *Attribute Revocation*. Existing attribute revocation methods designed for single authority CP-ABE [11]–[13] cannot be applied to multi-authority scenario. That is because, in multi-authority systems, there is no party to deal with the attribute revocation while still keeping the system secure against the collusion attack.

In this paper, we design an efficient multi-authority CP-ABE method without using a global authority and propose

1063-6927/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDCS.2012.42

a multi-authority access control scheme for cloud storage systems. With no global authority, existing techniques for key randomization in multi-authority schemes are no longer applicable, because there is no such a global authority to tie all the pieces together. In our method, we introduce a certificate authority (CA) to assign a global user identifier (UID) to each user as in [7] and an authority identifier (AID) to each authority. The UID can uniquely identify a user in the system and it is used together with the secret keys issued by different authorities for data decryption, such that it is impossible for two users to collude together to gain illegal access of data. We also propose a new technique to solve the attribute revocation problem in multi-authority CP-ABE systems. To improve the efficiency of attribute revocation, we move the work of re-encrypting the ciphertext to the server by using proxy encryption method, such that there is no need for the server to decrypt the ciphertext before re-encryption (i.e., the server cannot get the content key).

The main contributions of this work can be summarized as follows.

1) We design an access control framework for multi-authority systems and propose an efficient and secure multi-authority access control scheme for cloud storage.
2) We design an efficient multi-authority CP-ABE method that does not require a global authority and can support any LSSS access structure. Our multi-authority CP-ABE scheme is provably secure in the random oracle model.
3) We propose an efficient attribute revocation method for multi-authority CP-ABE, while still keeping the CP-ABE system secure against the collusion attack.

The remaining of this paper is organized as follows. In Section II, we give the related work on data access control and the attribute revocation in ABE systems. After the definition of system model and security model in Section III, some assumptions and definitions are given in Section IV. Section V proposes our multi-authority access control scheme in cloud storage and Section VI gives the analysis of our proposed scheme in terms of security, scalability and efficiency. Finally, the conclusion is given in Section VII.

II. Related Work

Cryptographic techniques are well applied to access control for remote storage systems [14]–[16]. The data owners encrypt files by using the symmetric encryption approach with content keys and then use every user's public key to encrypt the content keys. However, the key management is very complicated when there are a large number of data owners and users in the system. Also, the key distribution is not convenient in the situation of user dynamically joining or leaving the system, since it requires each data owner to be online all the time. Some methods [17]–[20] deliver the key management and distribution from the data owners to the remote server under the assumption that the server is trusted or semi-trusted. However, the server is cannot be trusted by the data owners in cloud storage systems and thus these methods cannot be applied to access control for cloud storage systems.

Attribute-based Encryption (ABE) is a promising technique that is very suitable for access control of encrypted data. After Sahai and Waters introduced the first ABE scheme [21], Goyal *et al.* [22] formulated the ABE into two complimentary forms: Key-Policy ABE (KP-ABE) and Ciphertext-Policy (CP-ABE). In KP-ABE systems, keys are associated with access policies and ciphertext is associated with a set of attributes; while in CP-ABE systems, keys are associated with a set of attributes and ciphertext are associated with access policies. In [23], the authors proposed a fine-grained data access control scheme based on the KP-ABE [22]. In their scheme, the data owner encrypts the data with a content key and then encrypt the content key by using the KP-ABE technique. The data owner assigns the access structure and the corresponding secret key to users by encrypting them by the user's public key and stores it on the server. However, their scheme requires the data owner to always be online for user joining, which is not appropriate in cloud storage systems. Some access control schemes are proposed based on CP-ABE [2], [11], [12], since CP-ABE is more suitable for the access control in cloud storage systems than KP-ABE. It allows the data owners to define an access structure on attributes and encrypt the data under this access structure, such that the data owners can define the attributes that the user needs to possess in order to decrypt the ciphertext. However, their schemes only serve for the situation that there is only one authority in the system and cannot be applied for multi-authority cloud storage systems.

Some new cryptographic methods are proposed to the multi-authority ABE problem [7]–[10], [24], [25]. Chase [7] proposed a solution that introduced a global identifier to tie users' keys together. The proposed scheme also relies on a central authority to provide a final secret key to integrate the secret keys from different attribute authorities. However, the central authority would be able to decrypt all the ciphertext in Chase's scheme, since it holds the master key of the system. Thus, the central authority would be a vulnerable point for security attacks and a performance bottleneck for large scale systems. Another limitation of Chase's scheme is that it can only express a strict "AND" policy over a pre-determined set of authorities. To improve Chase's scheme, Muller *et al.* [8] proposed a multi-authority ABE scheme that can handle any expressions in LSSS access policy, but it also requires a central authority. Chase *et al.* [9] also proposed a method to remove the central authority by using a distributed PRF (pseudo-random function). But it has the same limitation to strict "AND" policy of pre-determined authorities. Lin *et al.* [24] proposed a decentralized scheme based on threshold mechanism. In this scheme, the set of authorities is pre-determined and it requires the interaction among the authorities during the system setup. This scheme can tolerate collusion attacks for up to m colluding users, where m is a system parameter chosen at setup time. In [10], Lewko *et al.* proposed a new comprehensive scheme, which does not require any central authority. It is secure against any collusion attacks and it can process the access policy expressed in any Boolean formula over attributes. However, their method

is constructed in composite order bilinear groups that incurs heavy computation cost. They also proposed a multi-authority CP-ABE scheme constructed in prime order group, but they did not consider attribute revocation, which is one of the major challenges in multi-authority access control for cloud storage.

For attribute revocation in ABE systems, Pirretti *et al.* [26] proposed a timed rekeying mechanism, where an expiration time is set for each attribute. This approach requires the user to periodically go to the authority for key update, which incurs high overhead. In [2], the authors improved Pirretti's scheme by assigning an expiration time to the user's secret key (instead of attribute), so that the keys can be updated less frequently. In both schemes, user's secret keys can only be disabled at a designated time and thus the attribute revocation cannot take immediate effect. Some other revocation schemes [5], [27] only allow the user level revocation. That is, when one attribute of a user is revoked, the user loses all the access permissions to the data. Another user-revocable ABE scheme was proposed in [28], where the data owner uses broadcast encryption to update the new keys. This method requires the data owner to maintain full membership lists for all attribute groups, which is hard to achieve in cloud storage systems. An alternative method is to rely on the server to do attribute revocation [11], [12]. By the method proposed in [11], when a user is revoked, the authority redefines the master key and generates the new public keys for re-encrypting the data and new secret keys for users. Then, it asks the server to re-encrypt the data and update the secret keys for users. The revocation method proposed by Hur *et al.* in [12] lets the server re-encrypt the ciphertext with a set of attribute group keys. During an attribute revocation, the server only needs to change the attribute group key for the revoked attribute and re-encrypt the ciphertext with the new set of attribute group keys. However, both methods assume the server is trustable and thus cannot be applied to solve the attribute revocation problem in cloud storage systems.

III. Multi-authority Access Control Model

This section describes the system model and security model of our multi-authority access control scheme for cloud storage.

A. System Model

We consider a multi-authority access control system for cloud storage, as described in Fig.1. There are five types of entities in the system: the data owners (owners), the cloud server (server), the data consumers (users), the attribute authorities (AAs) and a certificate authority (CA). The owners define the access policies and encrypt their data under the policies before hosting them in the cloud. The server stores the owners' data and provides data access service to users. Each attribute authority is a trusted entity that is responsible for setting, revoking and updating user's attributes within its administration domain. The CA is a fully trusted entity which is responsible for issuing a global UID for each user and an AID for each AA in the system.

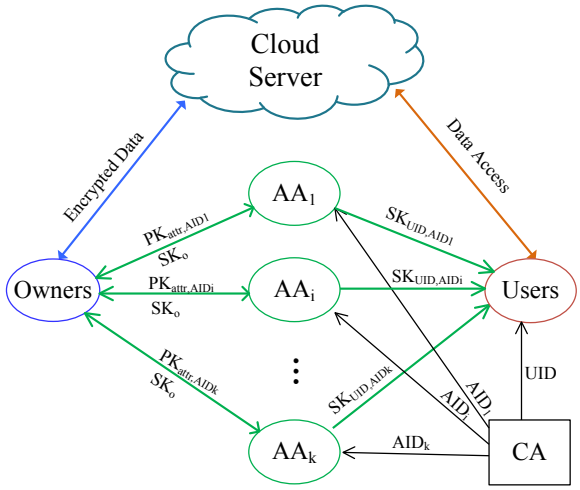


Fig. 1. System Model of Multi-authority Access Control in Cloud Storage

B. Security Model

In cloud storage systems, we consider that the server may send the owners' data to the unauthorized users who do not have access permission in order to gain more profit. The server is also curious about the content of the encrypted data or the received message. But we assume that the server will execute correctly the task assigned by the owner. The users, however, are dishonest and may collude to get unauthorized access to data.

We now describe the security model for multi-authority CP-ABE systems by the following game between a challenger and an adversary. Similar to the identity-based encryption schemes [29]–[31], the security model allows the adversary to query for any secret keys that cannot be used to decrypt the challenge ciphertext. We assume that the adversaries can corrupt authorities only statically similar to [7], [9], [10], but key queries are made adaptively. Let S_A denote the set of authorities. The security game is defined as follows.

Setup. The global Setup algorithm is run by the CA. The adversary specifies a set $S'_A \subseteq S_A$ of corrupted authorities. The challenger generates the pairs of public key and the secret key by running the key generation algorithm. For uncorrupted authorities in $S_A - S'_A$, the challenger sends only the public keys to the adversary. For corrupted authorities in S'_A, the challenger sends both the public keys and secret keys to the adversary. The adversary can also get all the public parameters such as the user's public key and all the public attribute keys.

Secret Key Query Phase 1. The adversary makes secret key queries by submitting pairs (S_{AID}, UID) to the challenger, where S_{AID} is a set of attribute belonging to an uncorrupted AA with AID and UID is a user identifier. The challenger gives the corresponding secret key $SK_{AID,UID}$ to the adversary.

Challenge. The adversary submits two equal length messages M_0 and M_1. In addition, the adversary gives a challenge access structure $(\mathbb{A}^*, \rho)$ which must satisfy the following constraints. We let V denote the subset of rows of $\mathbb{A}^*$ labeled

by attributes controlled by corrupted AAs. For each UID, we let V_{UID} denote the subset of rows of $\mathbb{A}^*$ labeled by attributes x belongs to the attribute sets that the adversary has queried. For each UID, we require that the subspace spanned by $V \cup V_{UID}$ must not include $(1,0,\ldots,0)$. In other words, the adversary cannot ask for a set of keys that allow decryption, in combination with any keys that can obtained from corrupted AAs. The challenger then flips a random coin b, and encrypts M_b under the access structure $(\mathbb{A}^*,\rho)$. Then, the ciphertext CT^* is given to the adversary.

Secret Key Query Phase 2. The adversary may query more secret keys, as long as they do not violate the constraints on the challenge access structure $(\mathbb{A}^*,\rho)$.

Guess. The adversary outputs a guess b' of b.

The advantage of an adversary $\mathcal{A}$ in this game is defined as $Pr[b'=b]-\frac{1}{2}$.

Definition 1. *A multi-authority CP-ABE scheme is secure against static corruption of authorities if all polynomial time adversaries have at most a negligible advantage in the above security game.*

IV. Assumptions and Definitions

In this section, we first give some background information on Bilinear Pairings and its security assumptions. Then, we describe the framework definition of our multi-authority access control scheme.

A. Bilinear Pairings

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three multiplicative groups with the same prime order p. A bilinear map is a map $e:\mathbb{G}_1\times\mathbb{G}_2\rightarrow\mathbb{G}_T$ with the following properties:

1) Bilinearity: $e(u^a,v^b)=e(u,v)^{ab}$ for all $u\in\mathbb{G}_1$, $v\in\mathbb{G}_2$ and $a,b\in\mathbb{Z}_p$.
2) Non-degeneracy: There exist $u\in\mathbb{G}_1$, $v\in\mathbb{G}_2$ such that $e(u,v)\neq I$, where I is the identity element of $\mathbb{G}_T$.
3) Computability: e can be computed in an efficient way.

Such a bilinear map is called a bilinear pairing.

If g_1 and g_2 are the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, $e(g_1,g_2)$ is the generator of $\mathbb{G}_T$. The bilinear pairing applied in our proposed scheme is symmetric, where $\mathbb{G}_1=\mathbb{G}_2=\mathbb{G}$.

B. Decisional Bilinear Diffie-Hellman Exponent Assumption

We define the decisional q-Bilinear Diffie-Hellman Exponent (q-BDHE) problem as follows. A challenger chooses a group $\mathbb{G}$ of prime order p according to the security parameter. Let $a,s\in\mathbb{Z}_p$ be chosen at random and g be a generator of $\mathbb{G}$. Let g_i denote g^{a^i}. When given $param=(g,g_1,\cdots,g_q,,q_{q+2},\cdots,g_{2q},g^s)$, the adversary must distinguish a valid tuple $e(g,g)^{a^{q+1}s}\in\mathbb{G}_T$ from a random element R in $\mathbb{G}$.

An algorithm $\mathcal{B}$ that outputs $z\in\{0,1\}$ has advantage ε in solving q-BDHE in $\mathbb{G}_T$ if $|Pr[\mathcal{B}(param,T=e(g,g)^{a^{q+1}s})=0]-Pr[\mathcal{B}(param,T=R)=0]|\geq\varepsilon$.

Definition 2. *The decisional q-BDHE assumption holds if no polynomial time algorithm has a non-negligible advantage in solving the q-BDHE problem.*

C. Access Control Framework

Definition 3. ***(Multi-Authority Access Control Scheme)*** *A multi-authority access control scheme is a collection of the following algorithms:* Setup, OwnerGen, AAGen, KeyGen, Encrypt, Decrypt, ReKey *and* ReEncrypt.

In our multi-authority access control scheme, the ReKey and ReEncrypt is used for attribute revocation. Each algorithm is defined as follows.

Setup. The setup algorithm takes no input other than the implicit security parameter. It outputs an AID for each AA, a global UID for each user and the user's public key PK_{UID}.

OwnerGen. The owner generation algorithm takes no input other than the public parameters. It outputs the owner's master key MK_o and the owner's secret key SK_o.

AAGen(AID). The authority generation algorithm takes the AID as input. It outputs the version key VK_{AID} and public attribute keys $\{PK_{x,AID}\}$ for all attributes x issued by the AA with AID.

KeyGen$(S,SK_o,VK_{AID},PK_{UID})$. The key generation algorithm takes as inputs a set of attributes S that describes the secret key, the owner's secret key SK_o, the current version key VK_{AID} and the user's public key PK_{UID}. It outputs the public key for the owner $PK_{o,AID}$ used for encryption and a secret key $SK_{UID,AID}$ for the user with UID.

Encrypt$(\{PK_{o,AID_k}\}_{k\in I_A},\{PK_{x,AID_k}\}_{x\in S_{AID_k},k\in I_A},MK_o,m,\mathbb{A})$. The encryption algorithm takes as inputs a set of owner's public keys $\{PK_{o,AID_k}\}_{k\in I_A}$ from the involved authority set I_A, a set of public attribute keys $\{PK_{x,AID_k}\}_{x\in S_{AID_k},k\in I_A}$ (S_{AID_k} is the selected attribute set from the AA with AID_k), the owner's master key MK_o, a message m and an access structure $\mathbb{A}$ over all the selected attributes from the involved AAs. The algorithm encrypts m according to the access structure and outputs a ciphertext CT. We will assume that the ciphertext implicitly contains the access structure $\mathbb{A}$.

Decrypt$(CT,PK_{UID},\{SK_{UID,AID_k}\}_{k\in I_A})$. The decryption algorithm takes as input the ciphertext CT which contains an access structure $\mathbb{A}$, the user's public key PK_{UID} and a set of user's secret keys $\{SK_{UID,AID_k}\}_{k\in I_A}$ from different AAs. If the set of attributes S satisfies the access structure $\mathbb{A}$, the algorithm will decrypt the ciphertext and return a message m.

ReKey$(SK_{UID,AID},\tilde{S}_{UID,AID})$. The key regeneration algorithm takes as inputs a secret key $SK_{UID,AID}$ for some set of attributes S and a new attribute set $\tilde{S}_{UID,AID}\subset S_{UID,AID}$. It outputs a new secret key $\widetilde{SK}_{UID,AID}$, a new version key $\widetilde{VK}_{AID}$ and the update key UK_{AID}.

ReEncrypt(CT,UI_{AID},UK_{AID}). The re-encryption algorithm takes as inputs the ciphertext CT, the update information UI_{AID} and the update key UK_{AID}. It outputs a new ciphertext $\widetilde{CT}$.

V. Multi-Authority Access Control Scheme with Efficient Revocation

In this section, we first give an overview of the challenges and techniques of designing multi-authority access control schemes for cloud storage systems. Then, we propose the

detailed construction of multi-authority CP-ABE method and use it to design an access control scheme by considering the attribute revocation problem.

A. Overview of Our Method

One challenging issue of the multi-authority CP-ABE scheme is the problem of collusion. Because we cannot rely on a central authority to tie together different components of a user's secret key, the traditional key randomization method cannot be applied to solve the collusion problem in multi-authority CP-ABE systems. Also, without a central authority, we need to design a new technique to tie the user's different key components together. In our method, we introduce a certificate authority (CA) to assign a global UID to each user and an AID to each attribute authority. We observe that each user's secret key is associated with a random number t in the traditional single-authority CP-ABE scheme. If a user holds multiple secret keys which are issued by different authorities, the random number in each secret key is different with each other. This makes each secret key independent with each other and thus cannot be colluded together for decryption. We use the global UID to replace this random number in each secret key for the user with UID. Because the UID is unique in the system, secret keys with the same UID can be tied together for decryption and it can prevent the collusion attack. With the AID, all the attributes are distinguishable even though some attributes present the same meaning.

The other challenging issue is the attribute revocation. Traditional attribute revocation methods usually require the authority to generate the new pair of public key and secret key by changing both the authority's master key and the random number in each user's secret key. In our scheme, the random number in each user's secret key is replaced by the user's UID. However, the UID of each user cannot be changed, thus traditional attribute revocation methods cannot be applied to our multi-authority CP-ABE systems. We propose a new technique by letting each owner hold its own master key, while each authority only holds its version key. The owner defines an access structure and encrypts the content key under this access structure by using its master key and the current public key, which comes from and can be updated by the authority. When an attribute revocation happens, the corresponding authority only needs to change its version key and generate an update key. It then sends the update key to the owner and all the users but the revoked one. By using the update key, the owner can generate the new public key and use it to encrypt the new data. All the users except the revoked one can efficiently update their secret keys via the update key, such that the revoked user cannot decrypt the data which is encrypted by the new public key. Then, the owner generates the update information and sends both the update information and the update key to the server for data re-encryption. Upon receiving the update key, the server can re-encrypt the ciphertexts affected by the revoked attributes by using the proxy encryption method, such that any new user who has sufficient attributes can still decrypt the ciphertexts which are published before it joins the system. Here, the ciphertexts is the encrypted form of content keys not the data components. Although the data component may be very large in cloud storage systems, the size of content keys is relatively small.

In order to realize the fine-grained access control, the owner first divides the data into several components according to the logic granularities (e.g., "name, address, security number, employer, salary") and encrypts each data component with different content keys by using symmetric encryption techniques. Then, the owner applies our multi-authority CP-ABE method to encrypt each content key, such that only the user whose attributes satisfy the access structure in the ciphertext can decrypt the content keys. Users with different attributes can decrypt different number of content keys and thus obtain different granularities of information from the data.

B. Our Proposed Multi-authority Access Control Scheme

Let $\mathbb{G}$ and $\mathbb{G}_T$ be the multiplicative groups with the same prime order p and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be the bilinear map. Let g be the generator of $\mathbb{G}$. Let $H : \{0,1\}^* \to \mathbb{Z}_p$ be a hash function such that the security will be modeled in the random oracle.

The construction of our access control scheme consists of four phases: System Initialization, Key Generation, Encryption and Decryption.

Phase 1: System Initialization

1) *CA Setup*. The CA authenticates all the authorities and assigns an AID to each of them. It also authenticates the users in the system and assign a UID to each user. Then, it generates the user's public key as $PK_{UID} = g^u$ by randomly choosing a secret $u \in \mathbb{Z}_p$.

2) *AA Setup*. Each AA runs the AAGen algorithm and generates a version key $VK_{AID} = \alpha_{AID}$. It then computes a public attribute key $PK_{x,AID}$ for each attributes x managed by the AA with AID as

$$PK_{x,AID} = g^{\alpha_{AID} \cdot H(x)}.$$

The public attribute keys can be accessed by all the owners.

3) *Owner Setup*. Each owner runs the OwnerGen algorithm. It randomly chooses $\beta, r \in \mathbb{Z}_p$ as the master key $MK_o = \{\beta, r\}$. Then, it computes the owner's secret key as $SK_o = \{g^{\frac{1}{\beta}}, \frac{r}{\beta}\}$ and sends it to each AA through a secure channel.

Phase 2: Key Generation

Each AA runs the KeyGen algorithm to generate the owner's public key and the user's secret key. The owner's public key is used for data encryption and the user's secret key is used for data decryption.

1) *Public Key Generation*. The AA with AID computes the owner's public key as

$$PK_{o,AID} = e(g,g)^{\alpha_{AID}}$$

from its version key VK_{AID}. Then, the AA sends the $PK_{o,AID}$ to the owner. Note that the owner's public key is associated with the current version key VK_{AID}.

2) *User's Secret Key Generation*. For each user with UID, each AA first authenticates whether the user has any

attributes managed by this authority. If the user has some attributes, the authority then assigns a set of attributes $S_{UID,AID}$ to this user according to its role or identity in its administration domain. Then, the AA generates the secret key $SK_{UID,AID}$ according to its attribute set $S_{UID,AID}$ as

$$SK_{UID,AID} = (\ K_{UID,AID} = (PK_{UID})^{\frac{r}{\beta}} \cdot g^{\frac{\alpha_{AID}}{\beta}},$$
$$\forall x \in S_{UID,AID} : K_{x,UID,AID} = (PK_{UID})^{\alpha_{AID} \cdot H(x)}\ ).$$

Phase 3: Encryption

Before hosting data M to the cloud servers, the owner processes the data as follows:

1) It divides the data into several data components as $M = \{m_1, \cdots, m_n\}$ according to the logic granularities. For example, the personal data may be divided into {name, address, security number, employer, salary}.
2) It encrypts data components with different content keys $\{k_1, \cdots, k_n\}$ by using symmetric encryption methods.
3) It then defines an access structure $\mathcal{M}$ for each content key $k_i (i = 1, \cdots, n)$ and encrypts each content key under its corresponding access structure by running the Encrypt algorithm.

When user's attributes satisfy the access structure embedded in the ciphertext, it can decrypt the ciphertext to obtain the content key and use it to further decrypt the data component. Because different users may have different attributes, they can decrypt different number of data components, such that they can get different granularities of information for the data.

The encryption algorithm can be constructed as follows.

$\mathsf{Encrypt}(\{PK_{o,AID_k}\}_{k\in I_A}, \{PK_{x,AID_k}\}_{x\in S_{AID_k}, k\in I_A}, MK_o, m, \mathbb{A})$. The encryption algorithm takes as inputs a set of owner's public keys $\{PK_{o,AID_k}\}_{k\in I_A}$ from the involved authority set I_A, a set of public attribute keys $\{PK_{x,AID_k}\}_{x\in S_{AID_k}, k\in I_A}$, the owner's master key MK_o, a message m and an access structure $\mathbb{A}$ over all the selected attributes from the involved AAs. Let $\mathcal{M}$ be a $l \times n$ matrix, where l denotes the total number of all the attributes. The function ρ associates rows of $\mathcal{M}$ to attributes. In this construction, we limit ρ to be an injective function, that is an attribute is associates with at most one row of $\mathcal{M}$.

For encryption, it first chooses a random encryption exponent $s \in \mathbb{Z}_p$ and chooses a random vector $\vec{v} = (s, y_2, \cdots, y_n) \in \mathbb{Z}_p^n$, where $y_2, \cdots, y_n$ are used to share the encryption exponent s. For $i = 1$ to l, it computes $\lambda_i = \vec{v} \cdot \mathcal{M}_i$, where $\mathcal{M}_i$ is the vector corresponding to the i-th row of $\mathcal{M}$. It then computes the ciphertext as

$$CT = (\ C = m \cdot (\prod_{k\in I_A} PK_{o,AID_k})^s, C' = g^{\beta s},$$
$$C_i = g^{r\lambda_i} (PK_{\rho(i),AID_i})^{-\beta s} (i = 1, \cdots, l)\ ).$$

In our scheme, the message m is the content keys $k_i (i = 1, \cdots, n)$. After the generation of all the ciphertexts, the owner sends the data to the server in the format as described in Fig. 2.

Phase 4: Decryption

CT_1	$En(m_1)_{k_1}$	...	.....	CT_n	$En(m_n)_{k_n}$

Fig. 2. Data Format on Cloud Server

Upon receiving the data from the server, the user runs the Decrypt algorithm to decrypt the ciphertext by using its secret keys from different AAs. Only the attributes the user possesses satisfy the access structure defined in the ciphertext CT, the user can get the content key from the ciphertext.

The decryption algorithm can be constructed as follows.

$\mathsf{Decrypt}(CT, PK_{UID}, \{SK_{UID,AID_k}\}_{k\in I_A})$. The decryption algorithm takes as inputs a ciphertext CT with access structure $(\mathcal{M}, \rho)$, the user's public key PK_{UID} and a set of user's secret keys $\{SK_{UID,AID_k}\}_{k\in I_A}$. If the user's attributes can satisfy the access structure, then the user proceeds as follows.

Let I be $\{I_{AID_k}\}_{k\in I_A}$, where $I_{AID_k} \subset \{1, 2, \cdots, l\}$ is defined as $I_{AID_k} = \{i : \rho(i) \in S_{AID_k}\}$. Let $n_A = |I_A|$ be the number of AAs involved in the ciphertext. Then, it chooses a set of constants $\{w_i \in \mathbb{Z}_p\}_{i\in I}$ and reconstructs the encryption exponent as $s = \sum_{i\in I} w_i \lambda_i$ if $\{\lambda_i\}$ are valid shares of the secret s according to $\mathcal{M}$.

The decryption algorithm first computes

$$\begin{aligned}
&\frac{\prod_{k\in I_A} e(C', K_{UID,AID_k})}{\prod_{k\in I_A}\prod_{i\in S_{AID_k}} (e(C_i, PK_{UID}) \cdot e(C', K_{\rho(i),UID,AID_k}))^{w_i n_A}} \\
=&\frac{\prod_{k\in I_A} e(g^{\beta s}, g^{\frac{ur}{\beta}} \cdot g^{\frac{\alpha_{AID_k}}{\beta}})}{\prod_{k\in I_A}\prod_{i\in S_{AID_k}} (e(g^{r\lambda_i} g^{-\alpha_{AID_k} H(\rho(i))\beta s}, g^u) \cdot e(g^{\beta s}, (g^u)^{\alpha_{AID_k} \cdot H(\rho(i))}))^{w_i n_A}} \\
=&\frac{e(g,g)^{usrn_A} \cdot \prod_{k\in I_A} e(g,g)^{\alpha_{AID_k} s}}{\prod_{k\in I_A}\prod_{i\in S_{AID_k}} e(g,g)^{ur\lambda_i w_i n_A}} \\
=&\frac{e(g,g)^{usrn_A} \cdot \prod_{k\in I_A} e(g,g)^{\alpha_{AID_k} s}}{e(g,g)^{usrn_A}} \\
=&\prod_{k\in I_A} e(g,g)^{\alpha_{AID_k} s}.
\end{aligned} \tag{1}$$

It can then decrypt the message $m = C / \prod_{k\in I_A} e(g,g)^{\alpha_{AID_k} s}$. Then, the user can use the decrypted content key to further decrypt the data component.

C. Attribute Revocation

Suppose an attribute of the user with UID' is revoked from the AA with AID'. The attribute revocation includes two phases: *Key Update* and *Data Re-encryption*. Note that the data re-encryption means to re-encrypt the ciphertexts of the content keys not the data components. The key update can prevent the revoked user from decrypting the new data which is encrypted by the new public keys. The data re-encryption can make sure that the newly joined user can still access the

previous data which is published before it joins the system, when its attributes satisfy the access policy associated with the ciphertext.

Phase 1: Key Update

The AA with AID' runs the ReKey algorithm to generate the new secret key $\widetilde{SK}_{UID',AID'}$ for the user with UID', the new version key $\widetilde{VK}_{AID'}$ and the update key $\widetilde{UK}_{AID'}$.

1) *Secret key update for the user with UID'*. The key regeneration algorithm takes as inputs a secret key $SK_{UID',AID'}$ for some set of attributes $S_{UID',AID'}$ and a new attribute set $\tilde{S}_{UID',AID'} \subset S_{UID',AID'}$.

The AA with AID' first randomly chooses $\tilde{\alpha}_{AID'} \in \mathbb{Z}_p$ as the new version key, where $\tilde{\alpha}_{AID'}$ is different to the previous version keys. Then, it generates a new secret key $\widetilde{SK}_{UID',AID'}$ for the user with UID' as

$$\widetilde{SK}_{UID',AID'} = (\ \tilde{K}_{UID',AID'} = (PK_{UID'})^{\frac{r}{\beta}} \cdot g^{\frac{\tilde{\alpha}_{AID'}}{\beta}},$$
$$\forall x \in \tilde{S}_{UID',AID'} : \tilde{K}_{x,UID',AID'} = (PK_{UID'})^{\tilde{\alpha}_{AID'} \cdot H(x)}\ ).$$

It then uses the new version key $\widetilde{VK}_{AID'}$ to generate the update key as

$$UK_{AID'} = (UK1_{AID'} = g^{\frac{\tilde{\alpha}_{AID'} - \alpha_{AID'}}{\beta}}, UK2_{AID'} = \frac{\tilde{\alpha}_{AID'}}{\alpha_{AID'}}).$$

The AA with AID' then sends the new secret key $\widetilde{SK}_{UID',AID'}$ to the user with UID'.

2) *Secret key update for other users who hold any attribute from the AA with AID'*. After the key regeneration, the AA with AID' sends out the update key $UK_{AID'} = (UK1_{AID'}, UK2_{AID'})$ to all the other users in its administration domain but the one with UID'. Upon receiving the update key $UK_{AID'}$, each user with UID_i who holds any attribute from the AA with AID' updates their own secret keys as

$$\widetilde{SK}_{UID_i,AID'} = (\ \tilde{K}_{UID_i,AID'} = K_{UID_i,AID'} \cdot UK1_{AID'},$$
$$\forall x \in S_{UID_i,AID'} : \tilde{K}_{x,UID_i,AID'} = (K_{x,UID_i,AID'})^{UK2_{AID'}}\ ).$$

3) *Public key update for the owner*. The AA with AID' also sends the update key $UK_{AID'} = (UK1_{AID'}, UK2_{AID'})$ to the owner for updating the public key and all the public attribute keys for all the attributes issued by this AA. Upon receiving the $UK_{AID'}$, the owner updates the previous public key $PK_{o,AID'}$ to the current version as $\widetilde{PK}_{o,AID'} = (PK_{o,AID'})^{UK2_{AID'}}$ and update each public attribute keys as $\widetilde{PK}_{x,AID'} = (PK_{x,AID'})^{UK2_{AID'}}$.

Phase 2: Data Re-encryption

For the data re-encryption, the owner first generate the update information $UI_{AID'}$. For each attributes in AA with AID' and involved in the ciphertext CT, the owner first calculates the update information $UI_{i,AID'}$ as

$$UI_{AID'} = (\forall x \in S_{AID'} :\ UI_{x,AID'} = (PK_{x,AID'} / \widetilde{PK}_{x,AID'})^{\beta s})$$

Then, it sends the update information $UI_{AID'} = \{UI_{i,AID'}\}_{i \in I_{AID'}}$ and the update key $UK_{AID'} = (UK1_{AID'}, UK2_{AID'})$ to the server for the data re-encryption. Upon receiving the update message, the server runs the ReEncrypt algorithm to re-encrypt the ciphertext. The new ciphertext $\widetilde{CT}$ is published as

$$\widetilde{CT} = (\ \tilde{C} = C \cdot e(UK1_{AID'}, C'),\ C' = g^{\beta s},$$
$$\forall i = 1, \cdots, l : \tilde{C}_i = C_i,\ \text{if } \rho(i) \notin S_{AID'},$$
$$\tilde{C}_i = C_i \cdot UI_{\rho(i),AID'},\ \text{if } \rho(i) \in S_{AID'}\ ). \quad (2)$$

From the Eq. 2, it is easy to find that our method only need to re-encrypt part of the ciphertext. This can greatly improve the computation efficiency of the attribute revocation. It is especially important in multi-authority systems because the ciphertext is usually encrypted by the attributes from multiple authorities and thus any attribute revocation from any involved AA may require the server to re-encrypt the ciphertext.

VI. Analysis of our Multi-Authority Access Control Scheme

This section first give the security analysis and scalability analysis of our multi-authority access control scheme. Then, it gives the performance analysis by comparing with the Lewko's scheme [10] in terms of storage overhead, communication cost and computation efficiency.

A. Security Analysis

We prove that our multi-authority access control is secure under the security model we defined in Section III-B as described in the following theorems.

Theorem 1. *Our multi-authority access control scheme is secure against the collusion attack.*

Proof: In our scheme, each user is assigned with a UID, which is a unique identifier in the system. For each user, all the secret keys it received from different authorities are all generated based on the user's UID. Thus, it is impossible for multiple users with different UIDs collude together to decrypt the ciphertext. Moreover, each attribute authority is assigned with an AID. With the AID, all the attributes are distinguishable even though some attributes present the same meaning. This can prevent the user from replacing the components of secret key from authority A with those components from another secret key issued by authority B. ■

Theorem 2. *When the decisional q-BDHE assumption holds, no polynomial time adversary can selectively break our system with a challenge matrix of size $l^* \times n^*$, where $n^* \leq q$.*

Proof: Suppose we have an adversary $\mathcal{A}$ with non-negligible advantage $\varepsilon = Adv_{\mathcal{A}}$ in the selective security game against our construction and suppose it chooses a challenge matrix M^* with the dimension at most q columns. In the security game we defined in Section III-B, the adversary can query any secret keys that cannot be used for decryption in combination with any keys it can obtain from the corrupted AAs. With these constraints, the security game in multi-authority systems can be treated equally to the one in single authority systems. Therefore, we can build a simulator $\mathcal{B}$ that plays the decisional q-BDHE problem with non-negligible advantage as the construction in [3]. ■

Besides, our access control scheme can prevent the server from getting the content of the data stored in the cloud. That is because the server cannot decrypt the data stored on it without any content keys. Also, by using the proxy-encryption method, the server does not need to decrypt the ciphertexts before doing the re-encryption.

B. Scalability Analysis

We give the scalability analysis by comparing our multi-authority CP-ABE method with the existing works in terms of the requirement of a global authority, the access structure type and the limitations of collude users it can tolerate.

TABLE I
SCALABILITY COMPARISON

Scheme	Requirement of Global Authority	Support of Policy Type	Number of Collude User
Our	No	Any LSSS	Any
[7]	Yes	Only 'AND'	Any
[8]	Yes	Any LSSS	Any
[9]	No	Only 'AND'	Any
[24]	No	Any LSSS	Up to m^*
[10]	No	Any LSSS	Any

* m is the threshold parameter.

From Table I, we can find that our scheme is much more scalable than most of the existing works. That is because: 1) our scheme does not require any global authority to involved in the key generation or management; 2) it can support any LSSS access structure and thus has high expressibility; 3) it can resist collusion attack with unlimited number of users. Although Lewko's scheme has the same scalability with our scheme, their scheme is not as efficient as our scheme. This will be analyzed in detail in the following performance analysis.

C. Performance Analysis

From Table I, we find that only Lewko's scheme [10] among the existing multi-authority CP-ABE schemes has the same scalability with our scheme. However, the first multi-authority CP-ABE scheme proposed by Lewko *et al.* is constructed in the composite order bilinear groups, it incurs much more computation cost than the second one constructed in the prime order bilinear groups. And our scheme is constructed in the prime order bilinear group, thus we choose the Lewko's second scheme for comparison. In this section, we analyze the performance of our scheme by comparing with the Lewko's scheme in terms of storage overhead, communication cost and computation efficiency.

We conduct the comparison under the same security level. That is the group order in our scheme has the same length with the group order of G in Lewko's scheme. Let $|p|$ denote the length of the element in $\mathbb{Z}_p$. Let $|G|$ and $|G_T|$ be the element size in $\mathbb{G}$ and $\mathbb{G}_T$ respectively. Let I_A denote the universe set of all the authorities in the system and each authority manages $n_k (k \in I_A)$ attributes. For a user with UID, let $n_{k,UID}$ denote the number of attributes obtained from the authority with $AID_k (k \in I_A)$. Let l be the total number of attributes used for encryption.

TABLE II
COMPARISON OF EACH COMPONENT

Component	Our Scheme	Lewko's Scheme
Authority Key	$\|p\|$	$n_k(\|p\|+\|p\|)$
Public Key	$\sum_{k\in I_A}(n_k\|G\|+\|G_T\|)$	$\sum_{k\in I_A} n_k(\|G_T\|+\|G\|)$
Secret Key	$\|G\|+\sum_{k\in I_A} n_{k,UID}\|G\|$	$\sum_{k\in I_A} n_{k,UID}\|G\|$
Ciphertext	$\|G_T\|+(l+1)\|G\|$	$(l+1)\|G_T\|+2l\|G\|$

First, we compare each component involved in our scheme and Lewko's scheme, as described in Table II. The authority key is the secret key that each authority keeps and the public key is used for encryption by the owner. The secret key is used by the user for decryption.

1) *Storage Overhead*

Table III describes the comparison of storage overhead on each entity between our scheme and Lewko's scheme. In our scheme, the storage overhead on the authority is just the version key, while in Lewko's scheme the authority's secret keys contribute to the storage overhead on it. The storage overhead of each owner comes from the public keys from multiple authorities, and the user's storage overhead also comes from the secret keys from multiple authorities. The storage overhead on the server is the size of ciphertext. From Table III, it is obvious that the storage overhead on the authority, the owner and the server in our scheme is much less than the one in Lewko's scheme. In our scheme, the storage overhead on each user is almost the same as the one in Lewko's scheme. Note that if more authorities involved in the system, our scheme incurs more less storage overhead than Lewko's scheme.

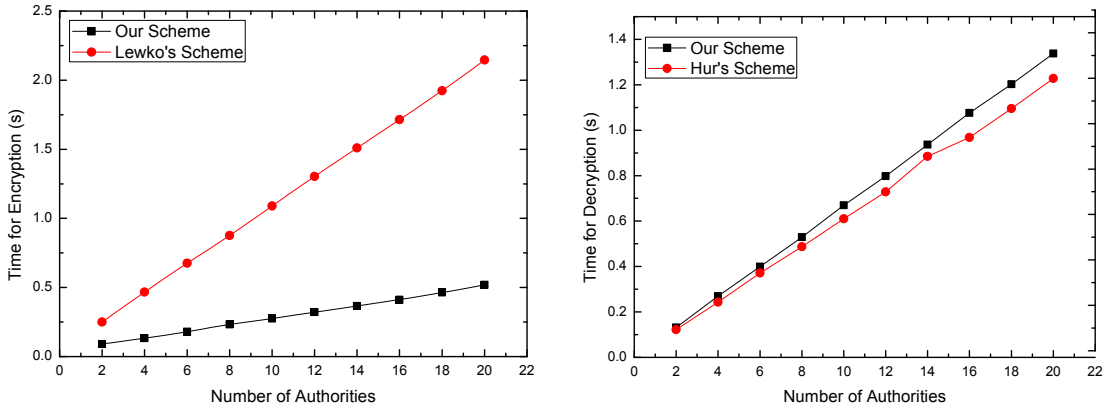
TABLE III
COMPARISON OF STORAGE OVERHEAD

Entity	Our Scheme	Lewko's Scheme
AA	$\|p\|$	$2n_k\|p\|$
Owner	$2\|p\|+\sum_{k\in I_A}(n_k\|G\|+\|G_T\|)$	$\sum_{k\in I_A} n_k(\|G_T\|+\|G\|)$
User	$\|G\|+\sum_{k\in I_A} n_{k,UID}\cdot\|G\|$	$\sum_{k\in I_A} n_{k,UID}\cdot\|G\|$
Server	$\|G_T\|+(l+1)\|G\|$	$(l+1)\|G_T\|+2l\|G\|$

2) *Communication Cost*

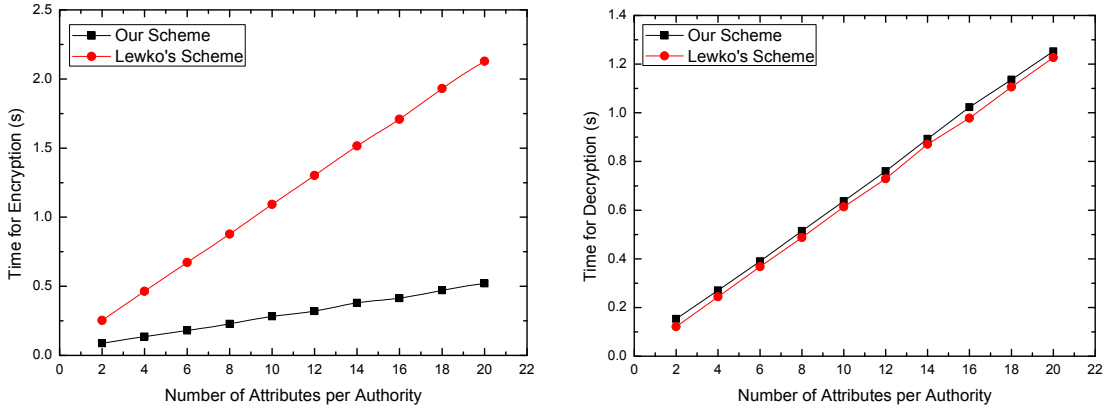
We conclude the communication cost for each owner and each user in the following Table IV.

The communication cost in the system is mainly caused by the keys and the ciphertexts. The communication cost between the authority and the user comes from the user's secret keys. Because the size of user's secret keys in our scheme is almost the same with the one in Lewko's scheme, the communication cost between the authority and the user in our scheme is also the same with the one in Lewko's scheme. The communication cost between the authority and the owner mainly comes from the public keys. In our scheme, the communication cost between the authority and the owner



(a) Encryption (b) Decryption

Fig. 3. Comparison of Time Consumption with Different Number of Authorities



(a) Encryption (b) Decryption

Fig. 4. Comparison of Time Consumption with Different Number of Attributes per Authority

TABLE IV
COMPARISON OF COMMUNICATION COST

Communication Cost between	Our Scheme	Lewko's Scheme
AA&User	$\|G\| + \sum_{k \in I_A} n_{k,UID}\|G\|$	$\sum_{k \in I_A} n_{k,UID}\|G\|$
AA&Owner	$\sum_{k \in I_A} (n_k\|G\| + \|G_T\|)$	$\sum_{k \in I_A} n_k(\|G_T\| + \|G\|)$
Server&User	$\|G_T\| + (l+1)\|G\|$	$(l+1)\|G_T\| + 2l\|G\|$
Server&Owner	$\|G_T\| + (l+1)\|G\|$	$(l+1)\|G_T\| + 2l\|G\|$

is less than the one in Lewko's scheme, because Lewko's scheme requires much larger size of public keys than our scheme, as described in Table II. The ciphertexts contribute the main communication cost between the server and the owner. Because the size of ciphertext in our scheme is much smaller than the one in Lewko's scheme, our scheme incurs less communication cost between the server and the owner than Lewko's scheme. Similarly, the communication cost between the server and the user in our scheme is much less than the one in Lewko's scheme. From the above analysis, it is easy to find that our scheme incurs less communication cost than Lewko's scheme.

3) *Computation Efficiency*

We implement our scheme and Lewko's scheme on a Linux system with an Intel Core 2 Duo CPU at 3.16GHz and 4.00GB RAM. The code uses the Pairing-Based Cryptography (PBC) library version 0.5.12 to implement the access control schemes. We use a symmetric elliptic curve α-curve, where the base field size is 512-bit and the embedding degree is 2. The α-curve has a 160-bit group order, which means p is a 160-bit length prime. The size of the plaintext is set to be 1 KByte. All the simulation results are the mean of 20 trials.

We compare the computation efficiency of both encryption and decryption in two criteria: the number of authorities and the number of attributes per authority. Fig.3(a) describes the comparison of encryption time with different number of authorities, where the involved number of attributes per authority is set to be 5. Fig.4(a) gives the encryption time comparison with different number of attributes per authority,

where the involved number of authority is set to be 5. It is easy to find that our scheme requires less time for encryption than Lewko's scheme.

Fig.3(b) shows the comparison of decryption time with different number of authorities, where the number of attributes the user holds from each authority is set to be 5. Suppose the user has the same number of attributes from each authority, Fig.4(b) describes the decryption time comparison with different number of attributes the user holds from each authority. In Fig.4(b), the number of authority for the user is fixed to be 5. It is not difficult to find that the time for decryption in our scheme is a little more than the one in Lewko's scheme. That is because there are more components in the ciphertext of Lewko's scheme than our scheme (that is the ciphertext in Lewko's scheme give more information than the ciphertext in our scheme), such that Lewko's scheme requires less computation cost for decryption. In fact, there is a tradeoff between the information given in the ciphertext and the time cost for decryption.

VII. Conclusion

In this paper, we defined a new access control framework for multi-authority systems in cloud storage and proposed an efficient and secure multi-authority access control scheme. We first designed an efficient multi-authority CP-ABE scheme that does not require a global authority and can support any LSSS access structure. Then, we proved that our multi-authority CP-ABE scheme is provably secure in the random oracle model. We also proposed a new technique to solve the attribute revocation problem in multi-authority CP-ABE systems. The analysis and simulation results showed that our proposed access control scheme is scalable and efficient. In the future work, we will remove the random oracle and extend our work to be provably secure in the standard model.

References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, Tech. Rep., 2009.

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

[3] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," *Public Key Cryptography–PKC 2011*, pp. 53–70, 2011.

[4] V. Goyal, A. Jain, O. Pandey, and A. Sahai, "Bounded ciphertext policy attribute based encryption," *Automata, Languages and Programming*, pp. 579–591, 2008.

[5] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 195–203.

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

[7] M. Chase, "Multi-authority attribute based encryption," *Theory of Cryptography*, vol. 4392, pp. 515–534, 2007.

[8] S. Müller, S. Katzenbeisser, and C. Eckert, "Distributed attribute-based encryption," *Information Security and Cryptology*, pp. 20–36, 2009.

[9] M. Chase and S. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 121–130.

[10] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," *Advances in Cryptology–EUROCRYPT 2011*, pp. 568–588, 2011.

[11] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*. ACM, 2010, pp. 261–270.

[12] J. Hur and D. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Transactions on Parallel and Distributed Systems*, 2010.

[13] S. Jahid, P. Mittal, and N. Borisov, "Easier: encryption-based access control in social networks with efficient revocation," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*. ACM, 2011, pp. 411–415.

[14] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable secure file sharing on untrusted storage," in *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*. Berkeley, CA, USA: USENIX Association, 2003, pp. 29–42.

[15] E. Goh, H. Shacham, N. Modadugu, and D. Boneh, "SiRiUS: Securing remote untrusted storage," in *Proceedings of the Tenth Network and Distributed System Security (NDSS) Symposium*. Citeseer, 2003, pp. 131–145.

[16] D. Naor, M. Naor, and J. Lotspiech, "Revocation and tracing schemes for stateless receivers," in *Advances in Cryptology–CRYPTO 2001*. Springer, 2001, pp. 41–62.

[17] D. Li, X. Du, X. Hu, L. Ruan, and X. Jia, "Minimizing number of wavelengths in multicast routing trees in wdm networks," *Networks*, vol. 35, no. 4, pp. 260–265, 2000.

[18] S. Di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Over-encryption: management of access control evolution on outsourced data," in *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment, 2007, pp. 123–134.

[19] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "A data outsourcing architecture combining cryptography and access control," in *Proceedings of the 2007 ACM workshop on Computer security architecture*, ser. CSAW '07. New York, NY, USA: ACM, 2007, pp. 63–69.

[20] W. Wang, Z. Li, R. Owens, and B. Bhargava, "Secure and efficient access to outsourced data," in *Proceedings of the 2009 ACM workshop on Cloud computing security*. ACM, 2009, pp. 55–66.

[21] A. Sahai and B. Waters, "Fuzzy identity-based encryption," *Advances in Cryptology–EUROCRYPT 2005*, pp. 457–473, 2005.

[22] V. Goyal, O. Pandey, A. Sahai, and 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*. ACM, 2006, pp. 89–98.

[23] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proceedings of the 29th conference on Information communications*. IEEE Press, 2010, pp. 534–542.

[24] H. Lin, Z. Cao, X. Liang, and J. Shao, "Secure threshold multi authority attribute based encryption without a central authority," *Information Sciences*, vol. 180, no. 13, pp. 2618–2632, 2010.

[25] J. Li, Q. Huang, X. Chen, S. S. M. Chow, D. S. Wong, and D. Xie, "Multi-authority ciphertext-policy attribute-based encryption with accountability," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '11. New York, NY, USA: ACM, 2011, pp. 386–390.

[26] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, "Secure attribute-based systems," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 99–112.

[27] X. Liang, R. Lu, and X. Lin, "Ciphertext policy attribute based encryption with efficient revocation," University of Waterloo, Tech. Rep., 2011.

[28] N. Attrapadung and H. Imai, "Conjunctive broadcast and attribute-based encryption," *Pairing-Based Cryptography–Pairing 2009*, pp. 248–265, 2009.

[29] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in cryptology*. Springer, 1985, pp. 47–53.

[30] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology–CRYPTO 2001*. Springer, 2001, pp. 213–229.

[31] C. Cocks, "An identity based encryption scheme based on quadratic residues," *Cryptography and Coding*, pp. 360–363, 2001.