Attribute-based Fine-Grained Access Control with Efficient Revocation in Cloud Storage Systems

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

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

Kui Ren
University at Buffalo
State University of New York
NY, USA
kuiren@buffalo.edu

ABSTRACT

Cloud storage service allows data owner to host their data in the cloud and through which provide the data access to the users. Because the cloud server is not trustworthy in the cloud storage system, we cannot rely on the server to conduct data access control. To achieve data access control on untrusted servers, traditional methods usually require the data owner to encrypt the data and deliver decryption keys to authorized users. In these methods, however, the key management is very complicated and inefficient. In this paper, we design an access control framework in cloud storage systems and propose a fine-grained access control scheme based on Ciphertext-Policy Attribute-based Encryption (CP-ABE) approach. In our scheme, the data owner is in charge of defining and enforcing the access policy. We also propose an efficient attribute revocation method for CP-ABE systems, which can greatly reduce the attribute revocation cost. The analysis shows that our proposed access control scheme is efficient and provably secure in the random oracle model.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Access controls, Cryptographic controls

General Terms

Security, Theory

Keywords

Access Control, Cloud Storage, Attribute Revocation, CP-ABE

1. INTRODUCTION

Cloud storage is an important service of cloud computing [4]. It allows data owners to host their data in the cloud that provides "24/7/365" data access to the users (data consumers). Cloud storage service separates the roles of the data owner from the data service provider, and the data owner does not interact with the user directly for providing data access service, which makes data access control a challenging issue. Existing methods delegate data access control to a trusted server and let it be in charge of defining and enforcing access policies [6]. However, the cloud server cannot be fully trusted by data owners, because the cloud server may give data access to unauthorized users for profit making (e.g., the competitor of a company). Thus, traditional server-based data access control methods are no longer suitable for cloud storage systems. The aim of this paper is to study the data access control issue in cloud storage system that does not rely on the cloud server.

The Ciphertext-Policy Attribute-based Encryption (CP-ABE) [1, 7] 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 and the policy checking occurs "inside the cryptography". However, due to the *attribute revocation problem*, it is very costly to apply the CP-ABE approach to the access control in cloud storage systems. We call a user whose attribute is revoked as a revoked user. There are two basic requirements for the attribute revocation in cloud storage systems: 1) *Backward Security* The revoked user cannot decrypt any new published ciphertext with its previous secret key. 2) *Forward Security* The newly joined user who has sufficient attributes can still be able to decrypt the ciphertexts which were published before it joined the system. Existing attribute revocation methods proposed for CP-ABE systems cannot be applied into the cloud storage systems. That is because they rely on the server to do auxiliary access control during the attribute revocation, which requires the server to be fully trusted. Thus, in cloud storage systems, the attribute revocation is still an open problem in the design of attribute-based data access control schemes.

In this paper, we design an attribute-based access control framework for cloud storage systems and propose a fine-grained access control scheme with efficient attribute revocation. Our scheme does not require the server to do any auxiliary access control and data owners are not required to be online all the time. The revocation is conducted efficiently on attribute level rather than on user level.

The main contributions of this work are summarized as follows.

1) We propose a fine-grained access control scheme for cloud storage systems, where the data owner is in charge of defining and enforcing the access policy without relying on any auxiliary access control by the server. Moreover, our scheme is provably secure in the random oracle model.

2) We propose a secure and efficient attribute revocation method for CP-ABE systems. It is secure in the sense that it can achieve both forward security and backward security, and it is efficient in the sense that it incurs less computation cost and communication overhead.

ASIA CCS'13, May 8–10, 2013, Hangzhou, China.

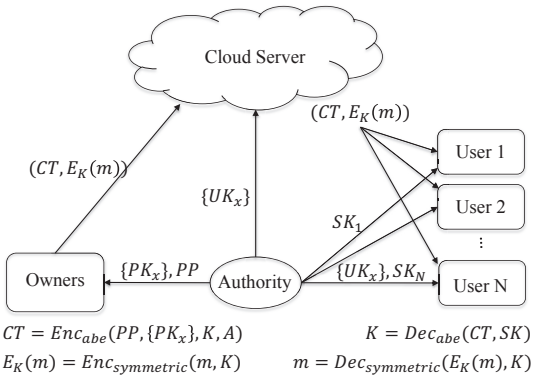


Figure 1: System Model of Access Control in Cloud Storage

3) We further provide the security analysis and performance analysis to show that our scheme is secure in the random oracle model and efficient to be applied into practice.

The remaining of this paper is organized as follows. Section 2 describes the definition of system model and security model. In Section 3, we describe the attribute-based fine-grained access control scheme and the efficient attribute revocation method. Section 4 gives the analysis of our scheme in terms of security and efficiency. In Section **??**, we give the related work on data access control and attribute revocation in ABE system. Finally, the conclusion is given in Section 5.

2. SYSTEM AND SECURITY MODEL

2.1 Definition of System Model

We consider an access control system for cloud storage service, as described in Figure 1. There are four entities in the system: authority, data owners (owner), cloud server (server) and data consumers (users).

The authority is responsible for entitling/revoking/re-granting attributes to/from/to users according to their role or identity in the system. It assigns secret keys to users when they are entitled attributes and maintains a version number of each attribute. When an attribute revocation happens, the authority will update the version number of the revoked attribute, and generate an update key. It then sends the update key to all the non-revoked users (for secret key update) and the cloud server (for ciphertext update).

The owners determine the access policies and encrypt their data under the policies before hosting them in the cloud (For simplicity, the data here means the content key[1]).

The cloud server stores the owners' data and provides data access service to users. But the server does not engage in the data access control. Instead, we assume the ciphertext may be accessed by all the legal users in the system. But, the *access control happens inside the cryptography*. That is only the users who possess eligible attributes (satisfying the access policy) can decrypt the ciphertext.

Each user is entitled a set of attributes according to its roles or identity in the system. However, the user's attribute set may dynamically change due to the role changed of the user in the system. For example, when a user is degraded from the manager to the normal worker, some of its attributes should be revoked, while sometimes the revoked attribute need to be re-granted to the user. The user can decrypt the ciphertext only when he/she has sufficient attributes satisfying the access policy associated with the ciphertext.

[1]In practical, the data is encrypted with a content key by using symmetric encryption method, and the content key is encrypted by using CP-ABE.

2.2 Definition of Framework

The framework of the data access control is defined as follows.

DEFINITION 1 (ACCESS CONTROL SCHEME). *An access control scheme is a collection of the following algorithms:* Setup, SKeyGen, Encrypt, Decrypt, UKeyGen, SKUpdate *and* CTUpdate, *where* UKeyGen, SKUpdate *and* CTUpdate *are used for attribute revocation.*

Setup$(1^\lambda) \rightarrow (MK, PP, \{PK_x\})$. The setup algorithm takes no input other than the implicit security parameter λ. It outputs a master key MK, the public parameters PP and a set of all the public attribute keys $\{PK_x\}$.

SKeyGen$(MK, S, \{VK_x\}_{x \in S}) \rightarrow SK$. The key generation algorithm takes as inputs the master key MK, a set of attributes S that describes the secret key, and the corresponding set of attribute version keys $\{VK_x\}_{x \in S}$. It outputs the user's secret key SK.

Encrypt$(PP, \{PK_x\}, m, \mathbb{A}) \rightarrow CT$. The encryption algorithm takes as inputs the public parameters PP, the set of public attribute key $\{PK_x\}$, a message m and an access structure $\mathbb{A}$ over the universe of attributes. The algorithm will encrypt m such that only a user who possesses a set of attributes satisfying the access structure will be able to decrypt the message. It outputs a ciphertext CT.

Decrypt$(CT, SK) \rightarrow m$. The decryption algorithm takes as inputs the ciphertext CT which contains an access structure $\mathbb{A}$ and the secret key SK for a set of attributes S. If the set of attributes S satisfies the access structure $\mathbb{A}$, then the algorithm will decrypt the ciphertext and return a message m.

UKeyGen$(MK, VK_{x'}) \rightarrow (\widetilde{VK}_{x'}, UK_{x'})$. The update key generation algorithm takes as inputs the master key MK and the current version key $VK_{x'}$ of the revoked attribute x'. It outputs a new version key $\widetilde{VK}_{x'}$ of the revoked attribute x' and an update key $UK_{x'}$.

SKUpdate$(SK, UK_{x'}) \rightarrow \widetilde{SK}$. The secret key update algorithm takes as inputs the current secret key SK and the update key $UK_{x'}$ of the revoked attribute x'. It outputs a new secret key $\widetilde{SK}$.

CTUpdate$(CT, UK_{x'}) \rightarrow \widetilde{CT}$. The ciphertext update algorithm takes as inputs the ciphertext CT and the update key $UK_{x'}$. It outputs a new ciphertext $\widetilde{CT}$.

2.3 Definition of Security Model

In cloud storage systems, we assume that: 1) The server may give access permission to the users who are not supposed to. 2) The server is curious but honest. It is curious about the content of the encrypted data or the received message, but will execute correctly the task assigned by the authority. 3) The users, however, are dishonest and may collude to obtain unauthorized access to data.

We now describe the security model for CP-ABE systems by the following game between a challenger and an adversary as follows.

Setup. The challenger runs the Setup algorithm and gives the public parameters, PK to the adversary.

Phase 1. The adversary is given oracle access to secret keys SK that corresponding to sets of attributes $S_1, S_2, \cdots, S_{q1}$ and the update keys UK.

Challenge. The adversary submits two equal length messages M_0 and M_1. In addition, the adversary gives a challenge access structure $\mathbb{A}^*$ such that none of the sets $S_1, \cdots, S_{q1}$ from Phase 1 satisfy the access structure. The challenger flips a random coin b, and encrypts M_b under the access structure $\mathbb{A}^*$. Then, the ciphertext CT^* is given to the adversary.

Phase 2. Phase 1 is repeated with the restrictions: 1) none of sets of attributes $S_{q1+1}, \cdots, S_q$ satisfy the access structure corre-

sponding to the challenge; 2) none of the updated secret keys $\widetilde{SK}$ (generated by the queried SK and update keys UK) can decrypt the challenge ciphertext.

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

The advantage of an adversary $\mathcal{A}$ in this game is defined as $Pr[b' = b] - 1/2$. This security model can easily be extended to handle chosen-ciphertext attacks by allowing for decryption queries in Phase 1 and Phase 2.

DEFINITION 2. *A revocable CP-ABE scheme is secure if all polynomial time adversaries have at most a negligible advantage in the above game.*

3. ATTRIBUTE-BASED ACCESS CONTROL WITH EFFICIENT REVOCATION

In this section, we first give an overview of our method and then propose the detailed construction of access control scheme. After that, we describe our attribute revocation method for CP-ABE.

3.1 Overview of Challenges and Our Solutions

To achieve fine-grained access control, the owner first divides the data into several components according to the logic granularities and encrypts each data component with different content keys by using symmetric encryption techniques. Then, the owner applies our proposed 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 same data.

To solve the attribute revocation problem, we assign a version number for each attribute. When an attribute revocation happens, only those components associated with the revoked attribute in secret keys and ciphertexts need to be updated, instead of all the components in the secret keys and ciphertexts. When an attribute is revoked from a user, the authority generates a new version key for this revoked attribute and generate an update key for it. With the update key, all the users, except the revoked user, who hold the revoked attributes can update its secret key (Backward Security). By using the update key, the components associated with the revoked attribute in the ciphertext can also be updated to the current version. To improve the efficiency, we delegate the workload of ciphertext update to the server by using the proxy re-encryption method, such that the newly joined user is also able to decrypt the previous published data, which are encrypted with the previous public keys (Forward Security). Moreover, all the users need to hold only the latest secret key, rather than keep records on all the previous secret keys.

3.2 Construction of Our 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 $H : \{0,1\}^* \to \mathbb{G}$ 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 by Authority, Data Encryption by Owners* and *Data Decryption by Users*.

Phase 1: System Initialization

The authority initializes the system by running the Setup algorithm. It randomly chooses $\alpha, \beta, \gamma, a \in \mathbb{Z}_p$ as the master key $MK = (\alpha, \beta, \gamma, a)$. Then, it generates the public parameters PP as

$$PP = (\ g,\ g^a,\ g^{1/\beta},\ g^{\beta},\ e(g,g)^{\alpha}\ )$$

For each attribute x, the authority generates a random number $v_x \in \mathbb{Z}_p$ as the initial attribute version number $VK_x = v_x$ and then applies it to generate a public attribute key PK_x as

$$PK_x = (\ PK_{1,x} = H(x)^{v_x},\ PK_{2,x} = H(x)^{v_x\gamma}\ ).$$

All the public parameters PP and the public attribute keys $\{PK_x\}$ are published on the public bulletin board of the authority, such that all the owners in the system can freely get them.

Phase 2: Secret Key Generation for Users

When a user joins the system, the authority first assigns a set of attributes S to this user according to its role or identity in the system. Then, the authority generates the secret key SK for this user by running the key generation algorithm SKeyGen. It takes as inputs the master key MK, a set of attributes S that describes the secret key, and the corresponding set of attribute version keys $\{VK_x\}_{x \in S}$. It then chooses a random number $t \in \mathbb{Z}_p$ and generates the user's secret key as

$$SK = (\ K = g^{\frac{\alpha}{\beta}} \cdot g^{\frac{at}{\beta}}, L = g^t, \forall x \in S : K_x = g^{t\beta^2} \cdot H(x)^{v_x t\beta}\ ).$$

The authority then sends SK to the user via a secure channel.

Phase 3: Data Encryption by Owners

Before hosting the data M to the cloud servers, the owner processes the data as follows. 1) It first divides the data into several data components as $M = \{m_1, \cdots, m_n\}$ according to the logic granularities. For example, the person record data may be divided into {name, address, security number, employer, salary}; 2) It encrypts each data component m_i with different content keys $k_i (i = 1, \cdots, n)$ by using the symmetric encryption techniques; 3) For each content key $k_i (i = 1, \cdots, n)$, the owner defines the access structure $\mathcal{M}$ over the universe of attributes $\mathbb{S}$ and then encrypts k_i under this access structure by running the encryption algorithm Encrypt.

The encryption algorithm $\textbf{Encrypt}(PP, \{PK_x\}, k, (\mathcal{M}, \rho)) \to CT$ can be constructed as follows. It takes as inputs the public parameters PP, a set of public attribute key $\{PK_x\}$, a content key k and a LSSS access structure $(\mathcal{M}, \rho)$. 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. It first chooses a random encryption exponent $s \in \mathbb{Z}_p$ and 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 $\mathcal{M}$. Then, it randomly chooses $r_1, r_2, \cdots, r_l \in \mathbb{Z}_p$ and computes the ciphertext as

$$CT = (\ C = ke(g,g)^{\alpha s},\ C' = g^{\beta s},\ C_i = g^{a\lambda_i}(g^{\beta})^{-r_i} H(\rho(i))^{-r_i v_{\rho(i)}},$$
$$D_{1,i} = H(\rho(i))^{v_{\rho(i)} r_i \gamma},\ D_{2,i} = g^{\frac{r_i}{\beta}}\ (i = 1, \cdots, l)\ ).$$

The owner then uploads the encrypted data to the server in the format as described in Figure 2.

CT_1	$E_{K_1}(m_1)$	$\cdots$	$\cdots$	CT_n	$E_{K_n}(m_n)$

Figure 2: Data Format on Cloud Server

Phase 4: Data Decryption by Users

Upon receiving the data from the server, the user runs the decryption algorithm Decrypt to obtain the corresponding content keys and use them to further decrypt data components. Only the attributes that the user possesses satisfy the access structure defined in the ciphertext CT, the user can decrypt the content key and then use it to decrypt the data component. Because different users may have different attributes, they are able to decrypt different number of data components, such that they can get different granularities of information from the same data.

The decryption algorithm $\textbf{Decrypt}(CT, SK) \rightarrow m$ is constructed as follows. It takes as inputs a ciphertext CT attached with the access structure $(\mathcal{M}, \rho)$ and the secret key for a set of attributes S. Suppose that the user's attribute set S satisfies the access structure and let $I \subset \{1, 2, \cdots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. 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

$$\frac{e(C', K)}{\prod_{i \in I}(e(C_i, L)e(D_{2,i}, K_{\rho(i)}))^{w_i}} = e(g, g)^{\alpha s}. \quad (1)$$

It can then decrypt the content key as $k = C/e(g, g)^{\alpha s}$. The user then uses the content keys to further decrypt the data.

3.3 Efficient Attribute Revocation for CP-ABE

When a user is leaving the system, the user should not be able to decrypt any data stored on the server. Thus, the access right of this user should be revoked, which is called the *User Revocation*. Another scenario is that a user is degraded in the system, some attributes should be removed from the previous set of attributes it possesses, which is called the *Attribute Revocation*.

In order to satisfy the requirements of attribute revocation, our revocation method includes three phases: *Update Key Generation by Authority*, *Secret Key Update for non-revoked Users* and *Ciphertext Update by Cloud Server*. Suppose an attribute x' is revoked from a user μ. The attribute x' is denoted as the *Revoked Attribute* and the user μ is denoted as the *Revoked User*. We also use the term of *Non-revoked Users* to denote the set of users who possess the revoked attribute x' but has not been revoked.

Phase 1: Update Key Generation by Authority

When there is an attribute revocation, the authority runs the update key generation algorithm $\textbf{UKeyGen}(MK, VK_{x'}) \rightarrow (\widetilde{VK}_{x'}, UK_{x'})$. It takes the master key MK and the current version key $VK_{x'}$ of the revoked attribute x' as inputs. It generates a new attribute version key $\widetilde{VK}_{x'}$ by randomly choosing a number $\tilde{v}_{x'} \in \mathbb{Z}_p (\tilde{v}_{x'} \neq v_{x'})$. Then, the authority computes the update key as

$$UK_{x'} = (\ UK_{1,x'} = \frac{\tilde{v}_{x'}}{v_{x'}},\ UK_{2,x'} = \frac{v_{x'} - \tilde{v}_{x'}}{v_{x'}\gamma}\ ).$$

It outputs a new version key $\widetilde{VK}_{x'}$ of the attribute x' and an update key $UK_{x'}$ that can be used for updating the secret keys of non-revoked users and the ciphertexts that are associated with the revoked attribute x'. Then, the authority sends the update key $UK_{x'}$ to all the non-revoked users (for secret key updating) and the cloud server (for ciphertext updating) via secure channels.

The authority also updates the public attribute key of the revoked attribute x' as $\widetilde{PK}_{x'} = (\ \widetilde{PK}_{1,x'} = H(x')^{\tilde{v}_{x'}},\ \widetilde{PK}_{2,x'} = H(x')^{\tilde{v}_{x'}\gamma}\ )$. After that, the authority broadcasts a message to all the owners that the public attribute key of the revoked attribute x' is updated. Then, all the owners can obtain the new public attribute key of the revoked attribute from the public bulletin board of the authority.

Phase 2: Secret Key Update for Non-revoked Users

Each non-revoked user submits two components $L = g^t$ and $K_{x'}$ of the secret key SK to the authority. Upon receiving these components, the authority runs the **SKUpdate** to compute a new component $\tilde{K}_{x'}$ associated with the revoked attribute x' as

$$\tilde{K}_{x'} = (K_{x'}/L^{\beta^2})^{UK_{1,x'}} \cdot L^{\beta^2} = g^{t\beta^2} \cdot H(x')^{\tilde{v}_{x'} t\beta}.$$

Then, it returns the new component $\tilde{K}_{x'}$ to the non-revoked user. The user's secret key is updated by replacing the component $K_{x'}$ associated with the revoked attribute x' with the new one $\tilde{K}_{x'}$:

$$\widetilde{SK} = (\ K,\ L,\ \tilde{K}_{x'},\ \forall x \in S \backslash \{x'\} : K_x\ ).$$

Note that only the component associated with the revoked attribute x' in the secret key needs to be updated, while all the other components are kept unchanged.

Phase 3: Ciphertext Update by Cloud Server

To ensure that the newly joined user who has sufficient attributes can still decrypt those previous data which are published before it joined the system, all the ciphertexts associated with the revoked attribute are required to be updated to the latest version. Intuitively, the ciphertext update should be done by data owners, which will incur a heavy overhead on the data owner. To improve the efficiency, we move the workload of ciphertext update from data owners to the cloud server, such that it can eliminate the huge communication overhead between data owners and cloud server, and the heavy computation cost on data owners. The ciphertext update is conducted by using proxy re-encryption method, which means that the server does not need to decrypt the ciphertext before updating.

Upon receiving the update key UK_x from the authority. The cloud server runs the ciphertext update algorithm $\textbf{CTUpdate}(CT, UK_{x'}) \rightarrow \widetilde{CT}$ to update the ciphertext associated with the revoked attribute x'. It takes as inputs the ciphertext CT and the update key $UK_{x'}$. It updates the ciphertext associated with x' as

$$\begin{aligned}\widetilde{CT} = (\ & \tilde{C} = C,\ \tilde{C}' = C',\ \forall i = 1\ to\ l :\ \tilde{D}_{2,i} = D_{2,i},\\ & if\ \rho(i) \neq x' :\ \tilde{C}_i = C_i,\ \tilde{D}_{1,i} = D_{1,i},\\ & if\ \rho(i) = x' :\ \tilde{C}_i = C_i \cdot (D_{1,i})^{UK_{2,x'}},\ \tilde{D}_{1,i} = (D_{1,i})^{UK_{1,x'}}\ )\end{aligned}$$

It is obvious that our scheme only requires to update those components associated with the revoked attribute in the ciphertext, while the other components are not changed. In this way, our scheme can greatly improve the efficiency of attribute revocation.

The ciphertext update can not only guarantee the forward security of the attribute revocation, but also can reduce the storage overhead on the users (i.e., all the users need to hold only the latest secret key, rather than to keep records on all the previous secret keys). The cloud server in our system is required to be semi-trusted. Even when the cloud server is not semi-trusted in some circumstance, which means that the server will not update the ciphertexts correctly. The forward security cannot be guaranteed, but our system can still achieve the backward security (i.e., the revoked user cannot decrypt the new published ciphertexts encrypted with the new public attribute keys).

4. ANALYSIS OF OUR PROPOSED ACCESS CONTROL SCHEME

4.1 Security Analysis

We conclude the security analysis as the following Theorems:

THEOREM 1. *When the decisional q-parallel 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. Under the constraint that none of the updated secret keys $\widetilde{SK}$ (generated by both the queried secret keys SKs and update keys UKs) can decrypt the challenge ciphertext, we can build a simulator $\mathcal{B}$ that plays the decisional q-parallel BDHE problem with non-negligible advantage. The detailed proof will be shown in the full version of our work. □

THEOREM 2. *Our proposed access control scheme is secure against the unauthorized access.*

Table 1: Comparison of Storage Overhead

Entity	Our Scheme	[3]
Authority	$(4+n_a)\cdot\|p\|$	$2\|p\|$
Owner	$(2+n_a)\|g\|+\|g_T\|$	$2\|g\|+\|g_T\|$
Server	$\|g_T\|+(3l+1)\|g\|$	$2\|g_T\|+(3l+3)\|g\|+\frac{l\cdot\|n_u\|\cdot\|p\|}{2}$
User	$(2+n_{a,i})\cdot\|g\|$	$(2n_{a,i}+1)\|g\|+\log(n_u+1)\|p\|$

Table 2: Comparison of Communication Cost

Communication Cost between	Our Scheme	[3]
Auth.&User	$4\|g\|+n_{a,i}\|g\|$	$\|g\|+2n_{a,i}\|g\|$
Auth.&Owner	$2\|g\|+\|g_T\|+n_a\|g\|$	$2\|g\|+\|g_T\|$
Server&User	$\|g_T\|+(3l+1)\|g\|$	$\|g_T\|+(2l+1)\|g\|+$ $(l\cdot\|n_u\|/2+\log(n_u+1))\|p\|$
Server&Owner	$\|G_T\|+(3l+1)\cdot\|G\|$	$(l+1)\|G_T\|+2l\|G\|$

PROOF. From the definition of the unauthorized access, there are two scenarios: 1) Users who do not have sufficient attributes satisfying the access structure may try to access and decrypt the data. 2) When one or some attributes of the user are revoked, the user may still try to access the data with his/her previous secret key.

For the first scenario, the users who do not have sufficient attributes cannot decrypt the ciphertext by using their own secret keys. We also consider the collusion attack from multiple users, in our scheme, the user's secret key is generated with a random number, such that they may not be the same even if the users have the same set of attributes. Thus, they cannot collude their secret keys together to decrypt the ciphertext.

For the second scenario, suppose one attribute is revoked from a user, the authority will choose another version key to generate the update key and sends it to the server for updating all the ciphertexts associated with the revoked attribute, such that the ciphertexts are associated with the latest version key of the revoked attributes. Due to the different values of the version key in the ciphertext, the revoked user is not able to use the previous secret key to decrypt the ciphertext. □

4.2 Performance Analysis

We give the analysis of our scheme by comparing with [3] in terms of storage overhead, communication cost and computation efficiency. Let $|p|$ be the size of elements in $\mathbb{Z}_p$. Let $|g|$ and $|g_T|$ be the element size in $\mathbb{G}$ and $\mathbb{G}_T$ respectively. Let n_a and n_u denote the total number of attributes and users in the system respectively. Let $n_{a,i}$ denote the number of attributes the user i possesses and let l denote the number of attributes associated with the ciphertext.

4.2.1 Storage Overhead

Table 2 shows the comparison of storage overhead on each entity in the system. The main storage overhead on the authority comes from the master key in [3]. Besides the master key, in our scheme, the authority needs to hold a version key for each attribute. Both the public parameters and the public attribute keys contribute the storage overhead on the owner in our scheme, which is linear to the total number of attributes in the system. Although the data is stored on the server in the format as shown in Figure 2, we do not consider the storage overhead caused by the encrypted data, which are the same in both our scheme and [3]. Our scheme only requires the server to store the ciphertext, while the server in [3] needs to store both the message head and the ciphertext which is also linear to the number of users in the system. The storage overhead on each user in our scheme is associated with the number of attributes it possesses, while in [3] the storage overhead on each user is not only linear to the number of attributes it possesses but also linear to the number of users in the system. Usually, the number of users are much larger than the number of attributes in the system, which means that our scheme incurs less storage overhead.

4.2.2 Communication Cost

As illustrated in Table 3 the communication cost in the system is mainly caused by the keys and ciphertexts. In our scheme, the communication cost between the authority and the user comes from both the user's secret keys and the update keys, while in [3] only the secret key contributes the communication cost between the authority and the user. The communication cost between the authority and the owner mainly comes from the public keys. In our scheme, when there is an attribute revocation, the owner needs to get the latest public attribute key of the revoked attributes, which also contributes the communication between the authority and the owner.

In our scheme, the communication cost between the server and the user comes from the ciphertext. But in [3], besides the ciphertext, the message head (which contains the path keys) also contributes the communication cost between the server and the users, which is linear with the number of all the users in the system. Thus, our scheme incurs less communication cost between the server and the user than [3]. The ciphertext contributes 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 [3], the communication cost between the sever and the owner is much less than the one in [3].

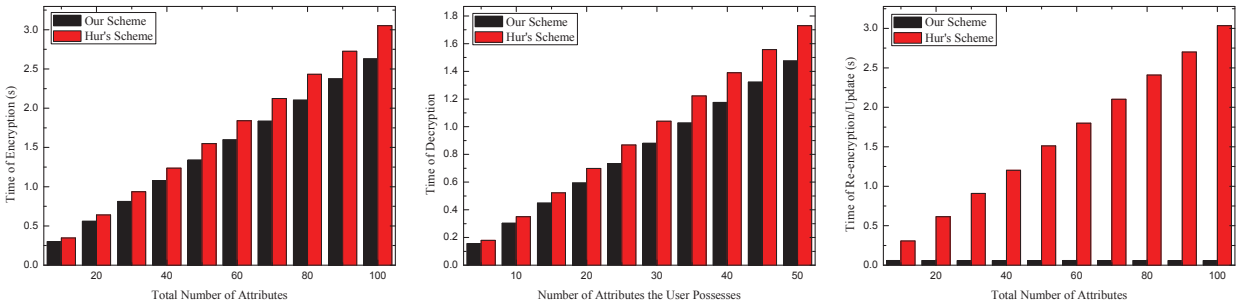
4.2.3 Computation Efficiency

We implement our scheme and [3] 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 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 between our scheme and [3] in terms of encryption, decryption and re-encryption [2]. From the Figure 3(a), we can see that the time of encryption is linear with the total number of attributes in the system. The encryption phase in our scheme is more efficient than the one in [3]. That is because, in [3], the owner first encrypts the data by using the CP-ABE scheme and sends the ciphertext to the server. Upon receiving the ciphertext from the owner, the server will re-encrypt the ciphertext with a randomly generated encryption exponent. Then, the server encrypts this exponent with a set of attribute group keys by using the broadcast encryption approach. Correspondingly, in the phase of decryption, the user should first decrypt the exponent with its own path key and uses it to decrypt the data together with the secret key. In our scheme, however, the user only needs to use the secret key to decrypt the data, which is more efficient than the [3] as illustrated in the Figure 3(b).

During the attribute revocation, our scheme only requires to update those components associated with the revoked attribute of the ciphertext, while the [3] should re-encrypt all the components of the ciphertext. Besides, the re-encryption in [3] should generate a new

[2]Note that we do not consider the computation of symmetric encryption for data components since they are the same in both our scheme and [3].



(a) Encryption (b) Decryption (c) Re-encryption

Figure 3: Computation Cost Comparison

encryption exponent and encrypt this new exponent with the new set of attribute group key by using broadcast encryption approach. Thus, as illustrated in Figure 3(c), the time of re-encryption phase in [3] is linear with the total number of attributes, while the time of ciphertext update in our scheme is constant to the number of revoked attributes.

5. RELATED WORK

The attribute-based encryption (ABE) technique [1, 2, 5, 7] is regarded as one of the most suitable technologies for data access control in cloud storage systems.There are two complementary forms of ABE, Key-Policy ABE (KP-ABE) [2] and Ciphertext-Policy ABE (CP-ABE) [1,7]. In KP-ABE, attributes are used to describe the encrypted data and access policies over these attributes are built into user's secret keys; while in CP-ABE, attributes are used to describe the user's attributes and the access policies over these attributes are attached to the encrypted data.

In [8], the authors proposed a fine-grained data access control scheme based on the KP-ABE approach [2]. 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 with 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 [1, 3], since CP-ABE is considered to be more suitable for data access control in cloud storage systems than KP-ABE. It allows data owners to define an access structure on attributes and encrypt the data under this access structure, such that data owners can define the attributes that the user needs to possess in order to decrypt the ciphertext. However, the revocation issue in CP-ABE is still an open problem.

To deal with the attribute revocation issue in ABE system, Yu *et al.* [9] proposed an attribute revocation method for CP-ABE, but they require the server to decide which users can update their secret keys according to the revoked user identity list, such that the server is required to be fully trusted. Hur *et al.* [3] also proposed an attribute revocation scheme in CP-ABE by allowing the server to re-encrypt the ciphertext with a set of attribute group keys. It can conduct the access right revocation on attribute level rather than on user level. During the attribute revocation, the server needs to change the attribute group key for the attribute which is affected by the membership change and re-encrypts the ciphertext with the new set of group attribute keys. This may incur high computation cost on the server. Also the server should be fully trusted. However, the server in cloud storage systems cannot be trusted and thus [3] cannot be applied in our problem. Therefore, the attribute revocation is still an open problem in attribute-based data access control.

6. CONCLUSION

In this paper, we proposed a fine-grained data access control scheme based on CP-ABE approach, where the owner was in charge of defining and enforcing the access policy. We also proposed an efficient attribute revocation method for CP-ABE, which can greatly reduce the cost of attribute revocation. Although this work is specific to cloud storage systems, but it is true that an untrusted remote storage system is an application of the work.

7. ACKNOWLEDGMENT

This work is supported by Research Grants Council of Hong Kong [Project No. CityU 114112] and in part by US National Science Foundation under grants CNS-1262277 and CNS-1116939.

8. REFERENCES

[1] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *S&P'07*, pages 321–334. IEEE Computer Society, 2007.

[2] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS'06*, pages 89–98. ACM, 2006.

[3] J. Hur and D. K. Noh. Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Trans. Parallel Distrib. Syst.*, 22(7):1214–1221, 2011.

[4] P. Mell and T. Grance. The NIST definition of cloud computing. Technical report, National Institute of Standards and Technology, 2009.

[5] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *EUROCRYPT'05*, pages 457–473. Springer, 2005.

[6] K. Sohr, M. Drouineaud, G.-J. Ahn, and M. Gogolla. Analyzing and managing role-based access control policies. *IEEE Trans. Knowl. Data Eng.*, 20(7):924–939, 2008.

[7] B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *PKC'11*, pages 53–70. Springer, 2011.

[8] S. Yu, C. Wang, K. Ren, and W. Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *INFOCOM'10*, pages 534–542. IEEE, 2010.

[9] S. Yu, C. Wang, K. Ren, and W. Lou. Attribute based data sharing with attribute revocation. In *ASIACCS'10*, pages 261–270. ACM, 2010.