26th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing

Enabling Dynamic and Efficient Data Access Control in Cloud Computing based on Attribute Certificate Management and CP-ABE

Somchart Fugkeaw
Thai Digital ID Co., Ltd.
Bangkok, Thailand
somchartf@yahoo.com

Hiroyuki Sato
Department of Electrical Engineering
and Information Systems,
The University of Tokyo, Japan
schuko@satolab.itc.u-tokyo.ac.jp

***Abstract*—In this paper, we propose an access control model featured with the efficient key update function in data outsourcing environment. Our access control is based on the combination of Ciphertext Policy – Attribute-based Encryption (CP-ABE) and Role-based Access Control (RBAC). The proposed scheme aims to improve the attribute and key update management of the original CP-ABE. In our scheme, a user's key is incorporated into the attribute certificate (AC) which will be used to decrypt the ciphertext encrypted with CP-ABE policy. If there is any change (update or revoke) of the attributes appearing in the key, the key in the AC will be updated upon the access request. This significantly reduces the overheads in updating and distributing keys of all users simultaneously compared to the existing CP-ABE based schemes. Finally, we conduct the experiment to evaluate the performance of our proposed scheme to show the efficiency of our proposed scheme.**

Keywords- access control; attribute certificate; CP-ABE, key update;performance

I. Introduction

Outsourcing data to a third party such as a cloud provider is becoming more convincing due to economy of scale and efficiency of resource management. However, security and privacy requirement generally stays as one of the top concerns for adopting data outsourcing service such as cloud computing. Traditional access control models such as mandatory access control (MAC), discretionary access control (DAC), and role-based access control model (RBAC) are not suitable for data outsourcing scenario since these models are designed to enforce the policy in the organizations where the resources housed at data owners. Generally, cloud provider is considered honest but curious, the privacy of data outsourced is still in risk of being compromised. Encryption is basically required to support the confidentiality of data in outsourcing environment.

Even though, public key encryption offers strong encryption, it would render the multiple copies of encrypted data as it uses each public key of all users to encrypt the data, while the symmetric encryption encounters the maintenance cost and key distribution problem when there are a large number of users accessing the shared data. Existing solutions for data access control in cloud computing generally apply attribute-based encryption to achieve both secure access control and privacy of data. Typically, symmetric and public key encryption can be applied to work with the ABE scheme to support fast encryption and digital signing [11, 15].

Ciphertext-policy attribute-based encryption (CP-ABE) is a kind of attribute-based encryption (ABE) approach proposed by Bethencourt et al. [1]. In CP-ABE, a set of attributes is assigned to each user in the way they are embedded into the user's secret key. The ciphertext is associated with the access policy structure in which encryptors or data owners can define the access policy by their own control. Users are able to decrypt a ciphertext if their attributes satisfy the ciphertext access structure. To date, several CP-ABE based solutions [2,4,5,7-9] have been proposed to reduce key management cost, optimize encryption and encryption performance, and address user and attribute revocation. However, these solutions adhere to the additional cryptographic-based enhancement to the CP-ABE process while the policy expressiveness in specifying user privilege has got less concerned. For the revocation problem, most research works favorably employ full scale proxy re-encryption (PRE) [10,12,13,16] to alleviate the computation cost burden at data owner side. Nevertheless, simultaneous key update and key distribution are also the expensive overheads arisen from the attribute revocation or policy update. These problems are not supported by the existing PRE techniques and most research works have overlooked these aspects.

In this paper, we propose a novel access control model called an Attribute Certificate - Ciphertext Policy – Attribute Role-based Encryption (AC-CP-ARBE) model to improve the flexibility and efficiency of key management of the existing CP-ABE based scheme[1-4, 11]. To this end, we employ attribute certificate (AC) used in Privilege Management Infrastructure (PMI) to express the readable user's attribute profiles and to support the authorization function of the CP-ABE scheme in the distributed systems. In our design, the AC is tailored to accommodate a well organized attribute-role schema and associated key for decrypting the file encrypted by the CP-ABE access policy. With the AC assigned to the individual user, the computation and communication cost at data owner or attribute authority (AA) in updating and distributing the keys to all existing users if there is attribute revocation is greatly reduced.

The contributions of this paper can be summarized as follows.

1. Our proposed AC-CP-ARBE access control model provides a practical linkage between data encryption based on CP-ABE and user decryption key management based on the attribute certificate (AC).
2. The proposed model is fully compatible with public key infrastructure where X.509 public key certificate is used for authentication and it can be used to verify the AC's issuer. Hence, the seamless integration of PKI and PMI can be established to enhance the trust of access control.
3. Compared to the original CP-ABE scheme, our AC-CP-ARBE provides no key distribution cost and no expensive computation of updating keys for all existing users.
4. We design and develop attribute certificate management system to support automatic attribute certificate enrollment and update in a highly secure and scalable manner.

II. Related work

Existing access control solutions for data outsourced in cloud computing are usually based on attribute-based encryption (ABE) which is generally classified into two types; key-policy attribute-based encryption (KP-ABE) [3,5] and Ciphertext Policy Attribute Based Encryption (CP-ABE) [1]. In KP-ABE, the ciphertext is associated with a set of attributes and user secret key is constructed to associate with the access structure. Thus, the ciphertext is independent of the access policy; instead it can be simply encrypted with a set of attributes that can be done by anyone. With this fact, the data owner lacks a full control over the access policy. In contrast, CP-ABE uses access policy structure defined by the data owner to encrypt the data. Users are able to decrypt a ciphertext if their attributes satisfy the ciphertext access structure.

To support a more complex environment of cloud computing where there is multi-owner and multi-authority, a multi-authority attribute based encryption (MA-ABE) has been proposed by several works [2-4, 7, 8, 11]. These works usually concentrate on security of cryptographic functions, optimization of encryption and decryption cost, and addressing revocation problem.

In [2], the authors propose hierarchical attribute-set-based encryption (HASBE) by extending ciphertext-policy attribute-set-based encryption (ASBE) with a hierarchical structure of users. In this scheme, a trusted authority is responsible for generating and distributing system parameters and root master keys as well as authorizing the top-level domain authorities. However, to serve the revocation and re-encryption process, the data owner needs to be available during the period agreed with users.

In [3], the authors propose a fine-grained and scalable data access control for personal health records (PHRs) by using attribute-based encryption. The paper addresses the access control in the multi-owners and multi-authority setting. This scheme supports the write access in the way that the requestors who have an authorization to write the data are granted a time-related signature. However, the revocation management relies on the universal CP-ABE where the cost of file re-encryption and key re-generation linear to number of users and policy size.

Kan Yang et al [4] proposes DAC-MACS (Data Access Control for Multi-Authority Cloud Storage model. The authors apply CP-ABE to construct an access control model where there are several multi-authority issuing the attributes. The proposed scheme improves the decryption process and solves revocation problem in ABE by designing the decryption token and key update and ciphertext update algorithms. However, the revocation cost for ciphertext and key update could be problematic if there are a large number of users as well as frequent and a high number of revoked attributes.

Specifically, the cost for key re-generation, key re-distribution, and file re-encryption caused from user and attribute revocation always is a cumbersome to degrade the efficiency of attribute-based access control deployment. Most recent research works [10,12,13,16] attempt to alleviate this problem by delegating the re-encryption process to a semi-trusted agent usually located in the outsourcing environment such as cloud. This concept is referred as Proxy-based Re-encryption (PRE) [6]. However, the PRE approach mainly resolves the re-encryption cost but the key update cost is another overhead that has not been solved by the existing PRE schemes.

Concretely, existing CP-ABE based schemes rely on the cryptographic operations executed at data owner or AA in re-generating and distributing keys to users when there is a policy update or revocation. These expensive overheads will exponentially grow if there are a large number of users.

III. backgrounD

A. *Ciphertext Policy Attribute-based Encryption (CP-ABE)*

Bethencourt et al. [1] have proposed ciphertext policy attribute-based encryption as another kind of attribute-based encryption (ABE) for access control. Basically, the concept of cryptographic construction for both ABE is based on the bilinear maps. A description of the formal definition of bilinear maps is shown below.

Bilinear Maps

Let G_1 and G_2 be two multiplicative cyclic groups of prime order p and e be a bilinear map $e : G_1 \times G_1 \rightarrow G_2$. Let g be a generator of G_1. Let H: $\{0,1\}^* \rightarrow G_1$ be a hash function that the security model is in random oracle.

The bilinear map e has the following properties:

1. Bilinearity: for all $u, v \in G_1$ and $a, b \in Z_p$, $e(u^a, v^b) = e(u, v)^{ab}$
2. Non-degeneracy: $e(g, g) \neq 1$.

Definition 1:Access Tree $\mathcal{T}$ [11]. Let $\mathcal{T}$ be a tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value. If num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_x \leqslant num_x$. When $k_x = 1$, the threshold gate is an OR gate and when $k_x = num_x$, it is an AND gate. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$. The *k of n* threshold gate is also allowed in $\mathcal{T}$, in this case $k_x = k$ where k is the threshold value determined in the *k of n* gate.

B. *Role-based Access Control*

RBAC is an access control model that provides the relationships of user`s role, permission, and objects or resources.

Definition 2: RBAC is a tuple of (*U, R, P, UA, PA, RH*) where:

(1) U, R, P are given sets that represent the set of users, roles, and permissions, respectively.

(2) UA ⊆U × R, a many-to-many user-to-role assignment relations;

(3) PA ⊆ P × R, a many-to-many permission-to-role assignment relation;

(4) RH ⊆ R × R is a partial order on R representing the role hierarchy.

In RBAC model, it supports role hierarchy where a role can be structured in hierarchy. The top role is on the top of hierarchy while less-powerful roles lay on the lower level.

C. *Extension of Access Tree*

To integrate the RBAC model into CP-ABE to enhance the expressiveness and scalability, we define attribute-role based (A-RBAC) access control model as follows:

Definition 3: A-RBAC is a tuple of (*U, R, P, UA, RH, D, APA, Attr*) where:

- *U, R, P, UA, RH* are as in the RBAC model,
- *D* ⊆ *R* × *Attr* is a many-to-many permission-to-role assignment relation.
- *APA* ⊆ *Attr* × *P* is a permission-to-attribute assignment.

We define *PA* ⊆ *R* × *P* as for r ∈ *R*, p ∈ *P*, *PA*(*r*,*p*) iff there exists an attribute *a* such that *D*(*r*,*a*) and *APA*(*a*,*p*). Hence, A-RBAC is an extension of RBAC.

In cloud computing environment, a user *u* (∈ *U*) is an entity to whom assigned a specific role r_u (∈*R*) and she requests to access the resource with a permission p (∈ *P*) i.e. read or write. Attributes *Attr* are a set of attributes used to characterize a role r_u. A set of attributes is issued by attribute authority AA.

In our cryptographic access control model, the extended access tree is referred to as an access control policy (ACP) used to encrypt the data files.

Figure 1 shows an example of an access control policy tree expressed in a patient treatment system. In the policy, there are three major roles pharmacist, nurse, and MD (medical doctor) who are all allowed to access the prescription records with different privileges.

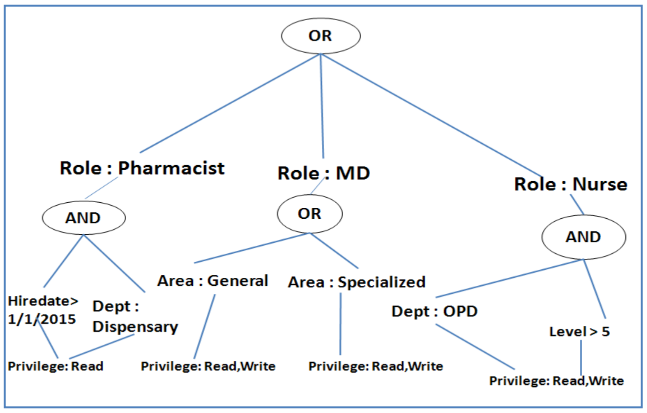

Fig. 1. Access Policy Structure

The above access structure is used to encrypt the prescription record. As of Figure 1, only the MD and nurse role are able to write this file. Users belonging to the MD role are supposed to access this policy in order to use it for encrypting prescription records after they update the file.

D. *Privilege Management Infrastructure (PMI)*

Privilege management infrastructure (PMI) is the term used to describe the process of authorization based on the ITU-T Recommendation X.509. PMI uses attribute certificate (AC) to hold user privileges in the form of attributes. Attribute certificates (ACs) are used to hold authorization information (privileges) while public key certificates (PKCs) hold authentication information [14].

PKCs are normally signed by the certification authority (CA) while AC has to be digitally signed by the issuing attribute authority (AA), and the PKC is used to validate the AA's signature. Also, the PKC is used to authenticate the identity of the AC holder when asserting privileges with this attribute certificate. Hence, the verification of an AC is synergistically supported by the presence of the PKC. Due to the possible change on attributes, a set of attributes in AC can be updated. AC comprises a digitally signed sequence of:

– Version Number of AC
– AC Holder
– AC Issuer
– Algorithm used to sign this AC
– Serial No.
– the validity period of this AC
– the sequence of attributes being bound to the holder any optional extensions.
– Extensions used to specify additional information (including time, targets, policies, and users), privilege revocation, roles, source of authority, and delegation of authority [14].

IV. OUR PROPOSED ACCESS CONTROL MODEL AC-CP-ARBE

A. *Overview of An Authorization Model*

In this paper, we incorporate attribute certificate to enhance the user key management of CP-ABE. At a core of the model, RBAC is integrated into CP-ABE in order to enhance the expressiveness of CP-ABE policy specification and manage a group attributes in the CP-ABE mechanism. Figure 2 illustrates the interplay of AC-CP-ARBE components.

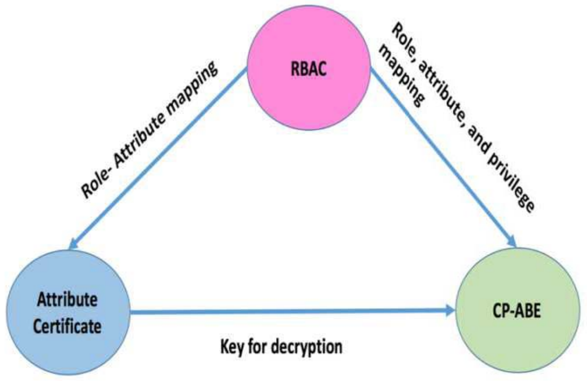

Fig.2. An Authorization Model combining CP-ABE, RBAC, and AC.

In essence, as the way of CP-ABE key management, the overhead for key update when there is attribute update or revocation is non-trivial, especially in large-scale CP-ABE deployment. In addition, the presence of attribute information may be necessary for proof of identity in certain applications. To this end, we propose and implement the extended model AC-CP-ARBE to support a more scalable and usable authorization in multi-authority cloud. In the AC-CP-ARBE scheme, a key generated by the AA or data owner is embedded in the AC instead of distributing the encrypted keys to users as existing CP-ABE schemes do. Similar to the CP-ABE, the authorization is enforced through the CP-ABE access policy used for data encryption.

B. *A Design of Attribute Certificate for Storing User Decryption Key*

The AC schema is based on the RFC 3281 [20]. Table 1 shows the AC details specifically designed for supporting role-based authorization and CP-ABE model.

In the AC structure, a set of attributes is mapped to a specific role and are shown in the attribute field. Attributes shown in the AC can be updated anytime and if there any changes on attribute value, the corresponding user decryption key (UDK) field will be updated.

Table 1. Attribute Certificate

Certificate Holder: John Smith
Issuer: AuthorityHos1 CA (AA1)
Algorithm: SHA2RSA
Serial No. 25ACB56
Validity 2016/01/20 – 2017/01/19
Attributes Information
Role: Doctor
Attributes: Name : John Smith
Office: Tokyo
Hospital: A
Level: 8
Position: Full MD
Professor: yes
MD Type: General
Specialty: Cardiologist
Hire date: 01/10/2008
Extension:
EDK: [xs44454754fdsefsd4f24]
Random Key Component :(E[$r \in_R Z_p$, D, g^r], Exp.Time)

A specific role and an associated set of attributes are issued by AA and they are used to determine the key value used to decrypt the data encrypted by CP-ABE policy. If a set of attributes contained in AC satisfies the ACP, the privilege of the authorized role in ACP will be enabled and the user can access the file with the assigned privilege. According to our model, there is no key distribution cost as the user can register for AC and gets the key embedded in the AC. The extension field contains two items: (1) encrypted UDK which is the key value associated to the set of attributes appearing in the AC, and (2) encrypted key components used to construct the UDK. The random key component consists of the random $r \in Z_p$ as defined in the bilinear map, a key component D, and key generator g. The computation of these components is shown in the next section AC-CP-ARBE cryptographic model, under Phase 2 UserKeyGen.

C. *AC-CP-ARBE Cryptographic Model*

Basically, the cryptographic construction of AC-CP-ARBE is based on bilinear maps as the original CP-ABE scheme [1] does.
Basically, the cryptographic construction of AC-CP-ARBE is extended from our C-CP-ARBE model [11]
The algorithms include System Setup, Key Generation, Attribute Certificate Issuance, Encryption, Decryption, and Revocation. Table 2 presents a list of notations used in our proposed model.

Table 2: Notations used in our model

Notation	Description
$S_{uid.k}$	Set of all attributes issued to user *uid* and managed by authority *k*.
SK_k	A secret key which belongs to authority *k*.
PK_k	Public key which belongs to authority *k*.
$PK_{x.k}$	Public attribute key issued by authority *k*.
GSK_{uid}	A global secret key of a user *uid*. GSK is a private key issued by the certification authority CA.
$Cert_{uid}$	A public key certificate containing user's public key issued by a certification authority CA.
$(PubK_k, PrivK_k)$	A PKI Key pair consisting of public key and private key issued by CA. This key pair is issued to the attribute authority *k*.
R_{id}	A role (identified with *id*) available in the system.
UL	A database that contains a set of user list of each role ($UL_{R_{id}}$)
$UDK_{uid.k}$	User Decryption key issued by an authority *k*
$EDK_{uid.k}$	EDK is an encrypted form of a UDK which is encrypted by a user public key.
GRP	Group role parameter is a list of user lists for all roles.
SS	Secret seal is a symmetric key created from the AES algorithm together with the GRP.
ACP	An extended access tree (referred to as an access control policy) used to encrypt the data files.
SCT	A sealed ciphertext which is a ciphertext encrypted with the SS

Phase 1: System Setup

This phase consists of the following five algorithms run by the AA or data owner.

1. **CreateAttributeAuthority**(k)→PK_k, SK_k, $PK_{x.k}$. This algorithm takes as input an attribute authority ID(k). It outputs the authority public key PK_k, SecretKey (SK_k), and public attribute keys ($PK_{x,k}$) for all attributes issued by the A_k.
2. **CreateRole**(SK_k, R_{id},) → $UL_{R_{id}}$. The CreateRole algorithm takes as inputs attribute authority's secret key (SK_k) , Role (R_{id}), It returns user list ($UL_{R_{id}}$) of users who qualify to the role R_{id} and stored in the database *UL*.
3. **UserRegister**(*uid*, $Cert_{uid}$). The UserRegister algorithm takes as input userID (*uid*) and user's certificate ($Cert_{uid}$), then it updates UL so that if *uid* has a role R_{id}, then *uid* is contained in $UL_{R_{id}}$.
4. **CreateGrouproleParameter** ()→ GRP. The Create GroupRole parameter algorithm first collects all user list $UL_{R_{id}}$ and concatenates them. GRP will be recomputed by either taking a new user *uid* to the R_{id} (add a new user) or remove a revoked user *uid* from R_{id} (revoke a user).

Phase 2: Key Generation

This phase is also run by the AA and it consists of two algorithms as follows:

5. **UserKeyGen**($S_{uid,k}$, SK_k, $Cert_{uid}$)→ $EDK_{uid,k}$. The KeyGen algorithm consists of two sub-modules:
   (1) UDKGen. It takes as input a set of attributes ($S_{uid,k}$) that describes the *uid*'s user decryption key, attribute authority's secret key (SK_k), and $Cert_{uid}$ of user *uid* issued by a CA, then it returns the set of user decryption key (UDKs).

   For each user *uid*, the AA A_k chooses a random r *and* $r_j \in \mathbb{Z}_p$, for each attribute $j \in S_{A_k}$. Then the user decryption key ($UDK_{uid.k}$) is computed as:

   $$UDK_{j,k} = (D = g^{(\alpha_k + r)/\beta_k}, A_i \in S : D_i = g^r . H(i)^{r_i}, D'_i = g^{r_i}).$$

   (2) EDKGen. The algorithm takes public key certificate of users ($Cert_{uid}$) issued by CA to encrypt the $UDK_{uid.k}$ and ($r \in_R Z_p$, D, g^r). Then $EDK_{uid.k}$ and encrypted key component (E[$r \in_R Z_p$, D, g^r] are produced. They will be used for embedding in the AC. The encryption function of EDKGen and key components are done by public key encryption function (ENC_{RSA}) which is expressed as:

   $ENC_{RSA}(Cert_{uid}, UDK_{uid,k}) \equiv EDK_{uid,k}$
   $ENC_{RSA}(Cert_{uid}, (r \in_R Z_p , D, g^r)) \equiv E(r \in_R Z_p , D, g^r)$

Phase 3: Attribute Certificate Issuance

This phase is run by the AA to issue and sign AC. This phase contains two algorithms as follows.

6. **IssueAC**(*param*, $PrivK_k$, $Cert_{uid}$, $S_{uid,k}$, $EDK_{uid,k}$, (E[$r \in_R Z_p$, D, g^r], Exp.Time))→ AC. This algorithm takes as input a set of attributes $S_{uid,k}$, encrypted $UDK_{uid.k}$ ($EDK_{uid.k}$), and encrypted set of key components (E[$r \in_R Z_p$, D, g^r], *Exp.Time*), then the AC is signed by the AA's key $PrivK_k$. Finally, the AC is issued encrypted by user's PKC ($Cert_{uid,}$) before it is sent to the respective user.

Phase 4: Encryption

The encryption function is either performed by data owners or users who have the write access privilege. This phase performs two encryption layers and secret seal encryption. It accommodates following two algorithms:

7. **ENC**(PK_k, *SS*, *M*, *ACP*, $Cert_{uid}$)→*SCT*. The encryption algorithm performs two consecutive steps as follows:

(1) Encrypt Message *M*: the algorithm takes as inputs authority public key PK_k, access control policy ACP, and data M. Then it returns a ciphertext CT.

(2) Encrypt *CT*: The algorithm takes secret seal (*SS*) calculated from the hash value of *GRP* to encrypt the ciphertext. Then, it returns a sealed ciphertext (SCT).

As the final encryption step, *SS* is encrypted with $Cert_{uid}$ and the encrypted secret seal (ESS) is produced.

Phase 5: Decryption
The decryption phase is usually done by users who want to access the encrypted file.

8. **DEC**(SCT, ESS, GSK_{uid}, $EDK_{uid\cdot k}$) → *M*.

The decryption algorithm performs two continuous steps as follows:

(1) Decrypt *SS* and *SCT*

The algorithm takes user's global secret key GSK_{uid} and it returns the session key *SS*. Then, the algorithm takes *SS* to decrypt *SCT* and gets the *CT*.

(2) Decrypt *CT*

The user *uid* takes a private key GSK_{uid} to decrypt $EDK_{uid,aid}$. Then, $UDK_{uid,k}$ is derived and it is used to decrypt the ciphertext. If the set of attribute *S* in $UDK_{uid,k}$ (stored in AC) satisfies the ACP structure encrypting the *CT*, the algorithm returns the message *M*.

Phase 6: Revocation
To revoke the attributes, the following algorithm will be executed.

9. **RevokeAtt**(SK_k , $\boldsymbol{Att_{rev}}$, *ACPs*)→ ACP′. The RevokeAtt algorithm takes as inputs AA's secret key SK_k, a set of attribute revoked $\boldsymbol{Att_{rev}}$, and all access control policies *ACPs* where the revoked attribute is included. The attribute revoked is removed from the affected ACPs. Then, the algorithm returns the updated ACP, ACP′. Then, a new ACP′ is used to re-encrypt the ciphertext. For the process of ciphertext re-encryption, proxy re-encryption (PRE) technique proposed in [16] is employed.

As for the attribute revocation effect, user decryption key update is done by updating the AC by each affected user. The process for UDK update is described in the next section.

D. User decryption key update: AC Update Process supporting attribute revocation

To support the automatic AC update, we develop an Attribute certificate management system (ACMS) by using IAIK java code [18] and attribute certificate management tool of Open PERMIS [19].

The ACMS is installed at the AA or data owner site to support AC enrollment or AC update. AC is signed by the authority or data owner key. Figure 3 exhibits the AC update process according to the attribute revocation. The trust of all key players and communication is assured by the PKI.

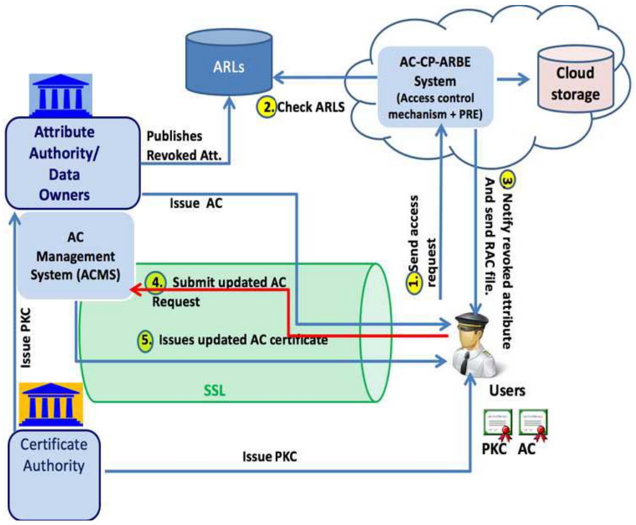
ARLs
AC-CP-ARBE System (Access control mechanism + PRE)
Cloud storage
2. Check ARLS
Attribute Authority/ Data Owners
Publishes Revoked Att.
Issue AC
AC Management System (ACMS)
1. Send access request
3. Notify revoked attribute And send RAC file.
4. Submit updated AC Request
5. Issues updated AC certificate
SSL
Issue PKC
Users
PKC
AC
Certificate Authority
Issue PKC

Fig. 3. AC Update Process

The AC and UDK update process consists of the following pro cedures.

1. A user makes a request for data access by presenting her PKC and AC as the credentials for authentication and authorization validation. The AC-CP-ARBE access control system consists of a collection of algorithms used to support cryptographic functions necessary for fine-grained access control. Also, the PRE mechanism [16] is embedded in our access control system to support ciphertext re-encryption.
2. If the authentication is successful, the system will check whether user attributes are all valid by checking the attribute revocation list (ARLs) published by AA or data owner. ARLs store the list of revoked attributes in a tuple <revoked attribute$_{aa.id}$ [name, value], timestamp>.
3. If the attribute is revoked, AA or data owner will notify the revoked attribute to user with the revoked attribute configuration (RAC) file. The RAC file contains attributes revoked for the particular user and it is signed by the data owner's key.

4. User logs in to the ACMS for updating key by submitting existing AC and the RAC file.
5. The ACMS will ask the user to decrypt the UDK (contained in the AC) and key components encrypted by his public key.
6. ACMS updates key by performing the key update procedure described in Table 3 as follows.

Table3: Key update procedure

KeyUpdate(due to attribute revocation)
Input: *SK_k, $UDK_{uid.k}$*, RAC file **Output:** Updated *$UDK_{uid.k}$* of user *uid*
1. AA checks the validity of Exp.Time of the key components $r \in_R Z_p$, D , g^r . *If expired*, AA will generate a new $r \in_R Z_p$, D and g^r . 2. AA checks a particular revoked attribute from RAC file, if it is found at i , then it removes the revoked attribute from $UDK_{uid.k}$ of a user *uid* and changse pointer of (D_{i-1} and D_{i-1}') to (D_{i+1} and D_{i+1}') as the active sequence and regenerates a new $\mathrm{UDK}_{\mathrm{uid.k}}$($UDK'_{uid.k}$). 3. AA outputs a new $UDK'_{uid.k}$ and a new set of key components (if $r \in_R Z_p$, D and g^r are regenerated). 4. AA takes a user's PKC to encrypt a UDK′ and key components, and update the encrypted value in the corresponding AC extension fields.

7. The ACMS will then sign the updated AC and return to the user. Then, the user can use her present AC where the updated key is embedded to access the encrypted data.

In our key update scheme, data owners or AA do not need to re-generate all affecting keys and re-distribute them to the users. Instead, users will get the notification for key update required, they can then update the key by their own convenience. This reduces the computation and communication cost at data owners or AA side as well as to give the flexibility for attribute revocation management. Besides, the key update component including random number has been assigned with the valid period of time, it will be only re-generated when the expiration is detected.

V. SECURITY ANALYSIS

A. Security Model

In the cloud computing environment, we assume that data owners are fully trusted. The users are assumed to be dishonest, i.e., they may collude to access unauthorized data. The AA can be corrupted or compromised by the attackers. The security model allows the adversary to request for any secret key, but they cannot be used to decrypt the challenge ciphertext. Since our cryptographic scheme is based the original CP-ABE, the detailed proof and security game are thus referred to [1].

B. Security Features of AC-CP-ARBE

- *Strong Authentication and authorization based on PKI and PMI*

Our system requires all entities to have the key pairs signed by the CA. All communication and access to the systems are authenticated by using the pubic key certificate and digital signature. In addition, the authentication model is linked to the authorization ticket, which is presented by an attribute certificate. The verification of AC can be verified through the issuer's signature.

- *Secure data access control*

Our cryptographic scheme is based the CP-ABE which is proven to be secure against collusion attack and secure under general security model. The detailed proof and security game can be found in [1,11].

- *Secure decryption key generation and update*

User key generation is securely supported by ACMS service located at data owner or attribute authority in an automatic manner. The request of AC update is validated with the PKI authentication and signature of the authority issuing the attributes. The process of AC signing request and key generation, and AC delivery is secure through the SSL communication. For the user decryption key generated, it is cryptographically protected by the respective user's public key (RSA key).

VI. EVALUATION AND EXPERIMENTS

In this section, we conduct the experiments to show the performance of our newly proposed scheme and the original CP-ABE. In our experiments, we implement three parts of access control function including cryptographic function, certification authority (CA), and ACMS. Basically, we use CP-ABE library [17] together with our proposed algorithms to implement a core of access control application service. The service is based on PHP and Java language which run on the Apache Sever. For the CA service, we use Open SSL to generate key pairs to users and system entities.
All services are run on Intel(R)Xeon(R)-CPU E5620, 2.40 GHz. with Ubuntu Linux.

To evaluate the efficiency of our proposed model, we compare key re-generation/key update performance of three schemes including our extended scheme AC-CP-RBE and the original CP-ABE. In the test scenario, an attribute is revoked from the amount of five attributes. To evaluate the worst-case scenario, we ran up to 800 simultaneous non-revoked clients requesting for key update /key re-generation to both systems.

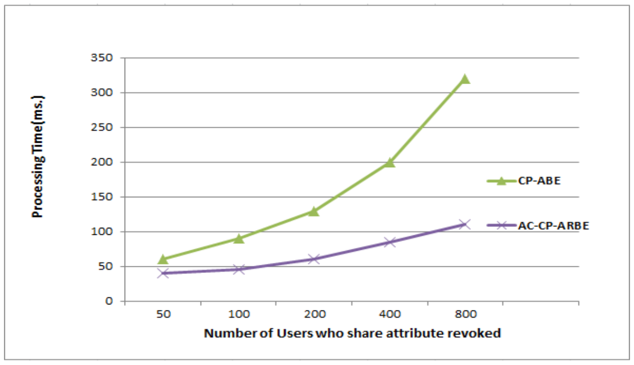

Fig.4. Comparison of key update cost

The graph shown in Fig.4 indicates that our proposed AC-CP-ARBE provides less cost for key update when there is attribute revocation than the original CP-ABE. The major overhead of CP-ABE is that it requires the computation cost for re-generating CP-ABE keys for all non-revoked users instantly. Also, all generated keys are encrypted and sent to the all affected users. In contrast, AC-CP-ARBE uses key update strategy which requires single computation cost $O(1)$ instead of key re-generation. Hence, the key is updated upon the active list of attributes contained in the AC. Also, the process of key distribution is not required. The key update algorithm is executed once the user submits the key update request. Hence, our proposed scheme provides less computation overhead for this portion of attribute revocation effect.

VII. Conclusion

We have presented the access control model based on the combination of CP-ABE, RBAC, and PMI to support flexible and scalable collaborative data sharing in cloud storage systems. In our design, AC does not only specify the authorization profile of the users, it enables key updating to be more practical and efficient when there is a case of attribute revocation. Our designed process for user key update based on the AC specification enables the efficient management of attribute-based revocation in large-scale system. Users can update their key flexibly while the communication and computation cost at data owner side is significantly minimised.

For future work, we will deploy our proposed scheme in a real cloud environment and evaluate the performance with a larger scale of data sets in both users and attributes to confirm the scalability and efficiency of our AC-CP-ARBE model.

References

[1] J. Bethencourt, , A. Sahai, and B. Waters, Ciphertext-policy Attribute-based Encryption, Proc. of IEEE Symposium of Security and Privacy, Oakland, CA, USA, May 20-23 2007, pp.321-334.

[2] Z. Wan, J. Liu, and R. H. Deng, HASBE: A Hierarchical Attribute-Based Solution for Flexible and Scalable Access Control in Cloud Computing. IEEE Transactions on Information Forensics and Security, Vol. 7(2),2012, pp.743-754.

[3] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, Scalable and Secure Sharing of Personal Health Records in Cloud Computing using Attribute-based Encryption, IEEE Transactions on Parallel and Distributed Systems, January 2013, Vol.24, pp.131-143.

[4] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, DAC-MACS: Effective Data Access Control for Multiauthority Cloud Storage Systems, IEEE Transactions on Information Forensics and Security, 2013, Vol. 8(11): pp.1790-1801.

[5] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-Based Encryption for Fine-grained Access Control of Encrypted Data, Proc. of CCS'06, Alexandria, Virginia, USA, Nov. 2006, pp.89-98.

[6] M. Mambo and E. Okamoto. Proxy cryptosystems: Delegation of the power to decrypt ciphertexts. IEICETransactions, Vol. E80-A(1), 1997, pp.54- 63.

[7] M. Chase and S. S. M. Chow, Improving privacy and security in multi-authority attribute-based encryption, Proc. of the 16th ACM Conference on Computer and Communications Security (CCS'09), ACM, Nov. 2009, pp.121-130.

[8] A. B. Lewko and B. Waters, Decentralizing attribute-based encryption, Proc. of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology, EUROCRYPT'11, Springer, May 2011, pp.568-588.

[9] L. Zhou, V. Varadharajan, M. Hitchens, Achieving Secure Role-Based Access Control on Encrypted Data in Cloud Storage, IEEE Transactions on Information Forensics and Security, Vol. 8(12), 2013, pp. 1947-1960.

[10] P. K. Tysowski, M. A. Hasan, Hybrid Attribute- and Re-Encryption-Based Key Management for Secure and Scalable Mobile Applications in Clouds. IEEE Transactions on Cloud Computing , Vol.1(2), 2013, pp. 172-186.

[11] S. Fugkeaw and H. Sato, An extended CP-ABE based Access control model for data outsourced in the cloud, Proc. of IEEE 39th Annual Computer Software and Applications (COMPSAC 2015), Taichung, Taiwan, July 2015, pp.73-78.

[12] Y. Kawai, Outsourcing the Re-encryption Key Generation: Flexible Ciphertext-Policy Attribute-Based Proxy Re-Encryption, Proc. of International Conference on Information Security Practice and Experience (ISPEC 2015), Beijing, China, May 2015, pp.301-315.

[13] J. Lai, R. H. Deng, Y. Yangj, J. Weng, Adaptable Ciphertext-Policy Attribute-Based Encryption, In Paring 2013. LNCS, Vol.8365, 2013, pp.199-214.

[14] D. Chadwick, The X.509 Privilege Management Infrastructure, Proc. of the NATO Advanced Networking Workshop on Advanced Security Technologies in Networking, Bled, Slovenia. 2003, pp. 15-25.

[15] C.Wise, C. Friedrich, S. Nepal, S. Chen, and R. O. Sinnott, Cloud Docs: Secure Scalable Document Sharing on Public Clouds, Proc. of IEEE International Conference on Cloud Computing (CLOUD'15), New York USA. June 2015, pp. 532-539.

[16] S. Fugkeaw and H. Sato, Improved Lightweight Proxy Re-Encryption for Flexible and Scalable Mobile Revocation Management in Cloud Computing, Proc. of IEEE International Conference on Cloud Computing (CLOUD'16), San Francisco, USA, 27 June – 2 July 2016, pp. 894-899.

[17] http://acsc.cs.utexas.edu/cpabe/

[18] http://javadoc.iaik.tugraz.at/iaik_jce/current/iaik/x509/attr/AttributeCertificate.html

[19] Open PERMIS source code, http://www.openpermis.info/

[20] AC Profile RFC3281,https://www.ietf.org/rfc/rfc3281.txt