International Conference on New Frontiers of Engineering, Management, Social Science and Humanities
Mahratta Chamber of Commerce, Industries and Agriculture, Pune (India)
25th February 2018 www.conferenceworld.in ISBN : 987-93-87793-04-0

DATA ACCESS CONTROL IN PUBLIC CLOUD STORAGE SYSTEM USING "CP-ABE" TECHNIQUE

R.Mahesh Muthulakshmi[1], Karthiga. E[2] , Ramani.K[3] , Suguna.K[4]

[1]AP/CSE , [2,3,4] Final year CSE Students

Indira Gandhi College of Engineering & Technology for Women, Chengalpattu

ABSTRACT

Data access control is the challenging issues in public cloud storage systems. In our paper the data security is improved using multiple authorities. Cipher text-Policy Attribute-Based Encryption (CP-ABE) has been adopted as a promising techinque to provide continuous, flexible, fine-grained and secure data access control for cloud storage with honest but curious cloud servers.

Keywords: Access control, Auditing, Cloud storage, CPABE.

I. INTRODUCTION

Cloud storage is a promising and important service paradigm in cloud computing. Benefits of using cloud storage include greater accessibility, higher reliability, rapid deployment and stronger protection, to name just a few. Since cloud storage is operated by cloud service providers, who are usually outside the trusted domain of data owners, the traditional access control methods in the Client/Server model are not suitable in cloud storage environment. The data access control in cloud storage environment has thus become a challenging issue.

To address the issue of data access control in cloud storage, there have been quite a few schemes proposed, among which Cipher text-Policy Attribute-Based Encryption (CP-ABE*)* is regarded as one of the most promising techniques. A salient feature of CP-ABE is that it grants data owners direct control power based on access policies, to provide flexible, fine-grained and secure access control for cloud storage systems. In CP-ABE schemes, the access control is achieved by using cryptography, where an owner's data is encrypted with an access structure over attributes, and a user's secret key is labelled with his/her own attributes. Only if the attributes associated with the user's secret key satisfy the access structure, can the user decrypt the corresponding cipher text to obtain the plaintext. So far, the CP-ABE based access control schemes for cloud storage have been developed into two complementary categories, namely, single-authority scenario, and multiauthority scenario.

i) CP-ABE

CP-ABE access control schemes have a lot of attractive features; they are neither robust nor efficient in key generation. Since there is only one authority in charge of all attributes in single-authority schemes, offline/crash of this authority makes all secret key requests unavailable during that period. The similar problem exists in multi-authority schemes, since each of multiple authorities manages a disjoint attribute set.

ii) RAAC

In this paper, inspired by the heterogeneous architecture with single *CA* and multiple *RAs*, we propose a robust and auditable access control scheme (named RAAC) for public cloud storage to promote the performance while keeping the flexibility and fine granularity features of the existing CP-ABE schemes. In our scheme, we separate the procedure of user legitimacy verification from the secret key generation, and assign these two sub-procedures to two different kinds of authorities. There are multiple authorities (named attribute authorities, *A*s), each of which is in charge of the whole attribute set and can conduct user legitimacy verification independently. Meanwhile, there is only one global trusted authority (referred as Central Authority, *CA*) in charge of secret key generation and distribution. Before performing a secret key generation and distribution process, one of the *AA*s is selected to verify the legitimacy of the user's attributes and then it generates an intermediate key to send to *CA*. *CA* generates the secret key for the user on the basis of the received intermediate key, with no need of any more verification. In this way, multiple *AA*s can work in parallel to share the load of the time consuming legitimacy verification and standby for each other so as to remove the single-point bottleneck on performance. Meanwhile, the selected *AA* doesn't take the responsibility of generating final secret keys to users. Instead, it generates intermediate keys that associate with users' attributes and implicitly associate with its own identity, and sends them to *CA*.

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

1) To address the single-point performance bottleneck of key distribution existed in the existing schemes, we propose a robust and efficient heterogeneous framework with single *CA*(Central Authority) and multiple *AA*s (Attribute Authorities) for public cloud storage. The heavy load of user legitimacy verification is shared by multiple *AA*s, each of which manages the universal attribute set and is able to independently complete the user legitimacy verification, while *CA* is only responsible for computational tasks. To the best of our knowledge, this is the first work that proposes the heterogeneous access control framework to address the low efficiency and single-point performance bottleneck for cloud storage.

2) We reconstruct the CP-ABE scheme to fit our proposed framework and propose a robust and high-efficient access control scheme, meanwhile the scheme still preserves the fine granularity, flexibility and security features of CPABE.

3) Our scheme includes an auditing mechanism that helps the system trace an *AA*'s misbehaviour on user's legitimacy verification.

II . PRELIMINARIES AND DEFINITIONS

In this section, we first give a brief review of background information on bilinear maps and the security assumptions defined on it. Then we briefly introduce CP-ABE and LSSS which are the constituents in our scheme.

A. Bilinear Maps

Let G, G*T* be two multiplicative cyclic groups with the same prime order *p*, and *g* be a generator of G. A bilinear

map $e : G \times G \rightarrow G_T$ defined on G has the following three properties:

1) *Bilinearity*: $\forall a, b \in Zp$ and $g1, g2 \in G$, we have $e(ga1 , gb2) = e(g1, g2)^{ab}$.

2) *Non-degeneracy*: $\forall g1, g2 \in G$ such that $e(g1, g2) \neq 1$,which means the map does not send all pairs in G × G

to the identity in G*T* .

3) *Computability*: There is an efficient algorithm to compute $e(g1, g2)$ for all $g1, g2 \in G$.

Definition 1. Decisional q-parallel Bilinear Diffie-Hellman Exponent Assumption (decisional q-BDHE): The decisional q-BDHE problem is that, in a group G of prime order p, give

a, s, b1, b2, . . . , bq ∈ Zp, if an adversary is given:

$\vec{y}$ = (g, gs, ga, . . . , g(aq), g(aq+2), . . . , g(a2q)

$\forall 1 \leq j \leq q$, $gs \cdot bj , ga/bj , \ldots , g(aq/bj), g(aq+2/bj), \ldots , g(a2q/bj)$

$\forall 1 \leq j,l \leq q, l \neq j$, $ga \cdot s \cdot bl/bj , \ldots , gaq \cdot s \cdot bl/bj)$,

it must remain hard to distinguish $e(g, g)^{aq+1s} \in$ GT from a random element R in GT .

B. Cipher text

Policy Attribute-Based Encryption (CP-ABE) Although the definitions and constructions of different CPABE schemes are not always consistent, the uses of the access structure in Encrypt and Decrypt algorithms are nearly the same. Here we adopt the definition and construction from .A CP-ABE scheme consists of four algorithms: Setup, Encrypt, Key Generation (KeyGen), and Decrypt. Setup(λ,*U*) → (*PK*,*MSK*). The setup algorithm takes the security parameter λ and the attribute universe description *U* as the input. It outputs the public parameters *PK* and a master secret key *MSK*.Encrypt(*PK*,*M*,A) → *CT*. The encryption algorithm takes the public parameters *PK*, a message *M*, and an access structure A as input. The algorithm will encrypt *M* and produce a ciphertext *CT* such that only a user whose attributes satisfies the access structure will be able to decrypt the message. We will assume that the ciphertext implicitly contains A. KeyGen(*MSK*, *S*) → *SK*. The key generation algorithm takes the master secret key *MSK* and a set of attributes *S* as input. It outputs a secret key *SK*.Decrypt (*PK*,*CT*, *SK*)→*M*. The decryption algorithm takes the public parameters *PK*, a cipher text *CT* which contains an access

policy A, and a secret key *SK* as input, where *SK* is a secret key for a set *S* of attributes. If the set *S* of attributes satisfies the access structure A, the algorithm will decrypt the cipher text and return a message *M*.

III. SYSTEM MODEL

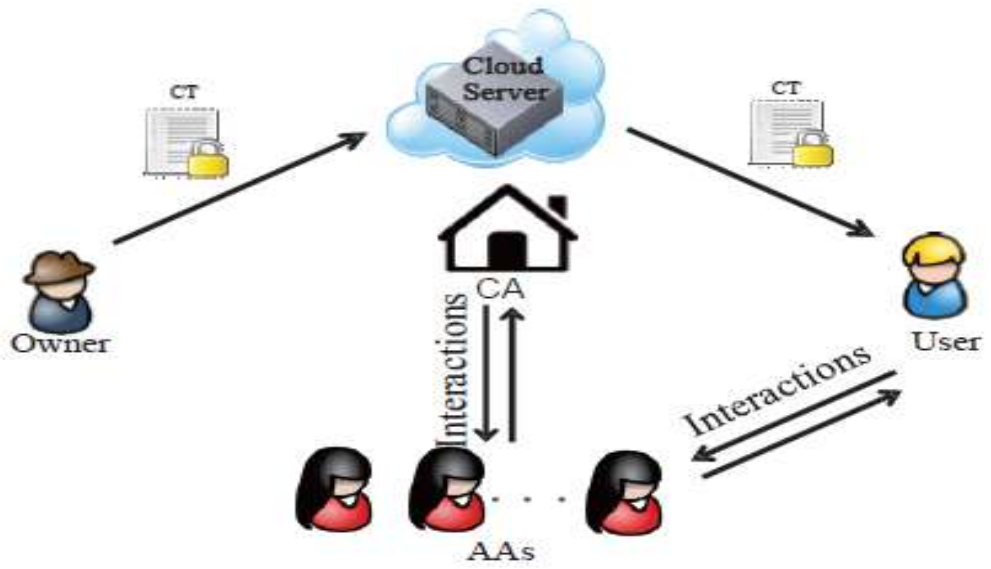

Fig. 1. System model

The system model of our design is shown in Fig. 1, which involves five entities: a *central authority (CA)*, multiple *attribute authorities (AAs)*, many *data owners (Owners)*, many *data consumers (Users)*, and a *cloud service provider with multiple cloud servers*(here, we mention it as *cloud server*.).

• **The central authority (CA)** is the administrator of the entire system. It is responsible for the system construction by setting up the system parameters and generating public key for each attribute of the universal attribute set. In the system initialization phase, it assigns each user a unique *Uid* and each attribute authority a unique *Aid*. For a key request from a user, *CA* is responsible for generating secret keys for the user on the basis of the received intermediate key associated with the user's legitimate attributes verified by an *AA*. As an administrator of the entire system, *CA* has the capacity to trace which *AA* has incorrectly or maliciously verified a user and has granted illegitimate attribute sets.

• **The attribute authorities (AAs)** are responsible for performing user legitimacy verification and generating intermediate keys for legitimacy verified users. Unlike most of the existing multi-authority schemes where each *AA* manages a disjoint attribute set respectively; our proposed scheme involves multiple authorities to share

the responsibility of user legitimacy verification and each *AA* can perform this process for any user independently.

When an *AA* is selected, it will verify the users' legitimate attributes by manual labour or authentication protocols,

and generate an intermediate key associated with the attributes that it has legitimacy-verified. Intermediate key is a new concept to assist *CA* to generate keys.

• **The data owner (Owner***)* defines the access policy about who can get access to each file, and encrypts the file under the defined policy. First of all, each owner encrypts his/her data with a symmetric encryption algorithm. Then,the owner formulates access policy over an attribute set and encrypts the symmetric key under the policy according to public keys obtained from *CA*. After that, the owner sends the whole encrypted data and the encrypted symmetric key (denoted as ciphertext *CT*) to the cloud server to be stored in the cloud.

• **The data consumer (User)** is assigned a global user identity *Uid* by *CA*. The user possesses a set of attributes and is equipped with a secret key associated with his/her attribute set. The user can freely get any interested encrypted data from the cloud server. However, the user can decrypt the encrypted data if and only if his/her attribute set satisfies the access policy embedded in the encrypted data.

• **The cloud server** provides a public platform for owners to store and share their encrypted data. The cloud server doesn't conduct data access control for owners. The encrypted data stored in the cloud server can be downloaded freely by any user.

IV. OUR PROPOSED ACCESS CONTROL SCHEME

Our scheme consists of five phases, namely ***System Initialization, Encryption, Key Generation, Decryption,* and *Auditing & Tracing*.**

To achieve a robust and efficient access control for public cloud storage, we propose a hierarchical framework with single *CA* and multiple *AA*s to remove the problem of single-point performance bottleneck and enhance the system efficiency. In our proposed RAAC scheme, the procedure of key generation is divided into two sub-procedures:

1) the procedure of user legitimacy verification

2) the procedure of secret key generation and distribution.

The user legitimacy verification is assigned to multiple *AA*s, each of which takes responsibility for the universal attribute set and is able to verify all of the user's attributes independently. After the successful verification, this *AA* will generate an intermediate key and send it to *CA*. The procedure of secret key generation and distribution is executed by the *CA* that generates the secret key associated with user's attribute set without any more

verification. The secret key is generated using the intermediate key securely transmitted from an *AA* and the master secret key.

1) System Initialization*:* Firstly, *CA* chooses two multiplicative cyclic groups G(the parameter *g* is a generator of

G) and G*T* with the same prime order *p*, and defines a binary map $e : G \times G \rightarrow GT$ on G. *CA* randomly chooses α, β, a and $b \in Zp$ as the master secret key. *CA* also randomly generates public keys for each attribute *Atti*, ($i = 1, 2, \ldots, U$):$h1, h2, \ldots, hU \in G$. Besides, let $H : (0, 1)* \rightarrow Zp$ be a hash function. The published public key is:$PK = GT, G, H, g, ga, e(g, g)\alpha, h1, ..., hU$ and the master secret key is:$MSK = \alpha, \beta, a, b$

which implicitly exists in the system, and doesn't need to be obtained by any other entity.Another task for *CA* in this operation is handling *AA*s' and users' registration. Here, *CA* generates a pair of keys (*skCA*, *vkCA*) to sign and verify, in which, *vkCA* is publicly known by each entity in the system. Each *AA* sends a registration request to *CA* during the *System Initialization*. For each legal *AA*, *CA* assigns a unique identity $Aid \in Zp$,randomly chooses a private key $kAid \in Zp$, and computes its corresponding public key $PKAid = gkAid$. Furthermore, *CA* generates a certificate *CertAid* which includes the public key *PKAid*, and sends it with the corresponding private key *kAid*

to the *AA* with the identity *Aid*. Meanwhile, each user gets his/her *Uid*, private key *kUid* and *CertUid* from *CA*.

2) Encryption*:* The procedure of *Encryption* is performed by the data owner himself/herself. To improve the system's performance, the owner first chooses a random number $\kappa \in GT$ as the symmetric key and encrypts the plaintext message *M* using κ with the symmetric encryption algorithm. The encrypted data can be denoted as $E\kappa(M)$. Then the owner encrypts the symmetric key κ using CP-ABE under the access policy A defined by himself/herself. The owner defines an easy expressed monotonic boolean formula. Following the method defined in [42], the owner can turn it to an LSSS access structure, which can be denoted as (M, ρ). Here,*M* is an $l \times n$ matrix, where *l* is the scale of a specific attribute set associated with a specific access policy and *n*

is a variable that is dependent on the monotonic Boolean formula definition and the LSSS turning method. The function ρ maps each row of *M* to a specific attribute, marked as $\rho(i) \in \{Att1, Att2, \ldots, AttU\}$. A random secret parameter *s* is chosen to encrypt the symmetric key κ. To hide the parameter *s*, a random vector $\vec{v} = (s, y2, y3, \ldots, yn) \in Znp$ is selected, where $y2, y3, \ldots, yn$ are randomly chosen and used to share the parameter *s*. Each $\lambda i = Mi \vec{v} T$ is computed for $i = 1, 2, \ldots, l$, where *Mi* denotes the *i*-th row of the matrix *M*. Owner randomly selects $r1, r2, \ldots, rl \in Zp$ and uses the public key generated by *CA* to compute.

$(C = \kappa e(g, g)\alpha s, C = gs, \forall i = 1 \text{ to } l, Ci = (ga)\lambda i \cdot h\rho(i) - ri , Di = gri)$.

V.CONCLUSION

By effectively reformulating CPABE cryptographic technique into our novel framework, our proposed scheme provides a fine-grained, robust and efficient access control with one-*CA*/multi-*AA*s for public cloud storage.

Our scheme employs multiple *AA*s to share the load of the time-consuming legitimacy verification and standby for serving new arrivals of users' requests. We also proposed an auditing method to trace an attribute Authority's potential misbehaviour. We conducted detailed security and performance analysis to verify that our scheme is secure and efficient. The security analysis shows that our scheme could effectively resist to individual and colluded malicious users, as well as the honest-but-curious cloud servers.

REFERENCES

[1] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Transactions on Parallel & Distributed Systems*, vol. 27, no. 9, pp. 2546– 2559,2016.

[2] J. Hong, K. Xue, W. Li, and Y. Xue, "TAFC: Time and attribute factors combined access control on Time sensitive data in public cloud," in *Proceedings of 2015 IEEE Global Communications Conference (GLOBECOM 2015)*. IEEE, 2015, pp. 1–6.

[3] Y. Xue, J. Hong, W. Li, K. Xue, and P. Hong, "LABAC:A location-aware attribute-based access control scheme for cloud storage," in *Proceedings of 2016 IEEE Global Communications Conference (GLOBECOM 2016)*.IEEE, 2016, pp. 1–6.

[4] J. Chen and H. Ma, "Efficient decentralized attribute based access control for cloud storage with user revocation,"in *Proceedings of 2014 IEEE International Conference on Communications (ICC 2014)*. IEEE, 2014, pp. 3782–3787.

[5] S. Hohenberger and B. Waters, "Online/offline attribute based encryption," in *Public-Key Cryptography–PKC 2014*. Springer, 2014, pp. 293–310.

[6] J. Shao, R. Lu, and X. Lin, "Fine-grained data sharing in cloud computing for mobile devices," in *Proceedings of 2015 IEEE Conference on Computer Communications (INFOCOM 2015)*. IEEE, 2015, pp. 2677–2685.