SECURE SHARING OF DATA IN THE CLOUD COMPUTING USING PROXY RE-ENCRYPTION AND CP-ABE

Abu Salim¹, Rajesh Kumar Tiwari², Sachin Tripathi³

¹Department of Computer Science, College of CS/IS, Jazan University, Jazan, Kingdom of Saudi Arabia (KSA)
²Department of Computer Science and Engineering, RVS College of Eng. & Tech., Jamshedpur, Jharkhand, India
³Department of Computer Science Engineering, Indian School of Mines, Dhanbad, Jharkhand, India

ABSTRACT:

Cloud computing is an internet based computing model which provides services using three models, SaaS, IaaS, and PaaS. Organizations are using cloud storage server, a service offered by IaaS, to share their data with authorized users which brings many benefits like cost-efficiency, high productivity and less hand on management. However, it also raises much concern to its privacy and security. In this paper we propose a data sharing framework using proxy re-encryption (PRE) and cipher-text attribute based encryption (CP-ABE), this scheme ensure privacy and security of data from unauthorized access by users, cloud service provider and illegal authorization by cloud service provider to the user for accessing the data.

Keywords: Secure sharing, Symmetric key, CP-ABE, PRE, Asymmetric key, CSP, Cloud Computing.

[1] INTRODUCTION

Organizational data can be outsourced to the cloud storage which provides enormous benefits due to its characteristics which includes scalability, fault tolerance, pay per use, broad network access, increase productivity. However, outsourcing data not only brings advantage but also raise much concern to its privacy and security. There many types of attack which affect services of cloud storage server and even some of them may compromise data privacy and security. XML Signature Wrapping attacks can take administrative rights of the cloud users and can create, modify and delete the data [1], cross side scripting attack can injects a piece of code into web application to bypass access control mechanism [1], In Flooding attack problem
malicious user can overload server by sending bogus request [2], in denial of service attack, malicious code is injected into web browser to open many windows, so legitimate user can’t use the service. The attackers have different motive such as, to steal valuable data so they can get monetary benefits, to cause controversy, some former employees may take revenge by hacking the cloud storage and can steal, modify or in worst case delete the entire data. Therefore, many organizations hesitate to adopt cloud services [3, 4].

Consider that CSP provides cloud storage as a service, where data owner can store an access the data. These data are also accessed by the authorized user (authorized by data owner). To share the data in cloud the following privacy and security requirements must be addressed.

[1] **Data Confidentiality:** Unauthorized user should be not allowed to access the data not even the cloud service provider. Confidentiality of data can be compromise during transition or at rest. So data should be protected both during transition and data at rest.

[2] **User Revocation:** Once the access right of user revoked he should be not able to access data, revoking access right of a user should not affect other users in the group.

[3] **Scalable and Efficient:** Cloud storage should be scalable and efficient, a new user can join and an old user access rights can be revoked with minimal effort.

[4] **Collusion between entities:** Even when certain users collude, they should not be able to access shared data without data owner permissions.

A number of schemes has been purpose [5-10] to solve above problem, but best of our knowledge no scheme solve all of the above mentioned problem. Therefore, we need data sharing scheme in which data owner should be able to impose access policy so only authorized user can access the data and CSP should not be able to access the data or authorizing unauthorized user to access the data.

In this work we will propose a scheme which can solve above problem using a combination of symmetric and asymmetric key cryptography. Data is encrypted using symmetric key we can use any symmetric key cryptography but we purpose to use Advanced Encryption Standard (AES) [11] which offer more security on the small key size [12].

The organizations of this paper are as follows. Section-II presents a literature survey on the related work. Section-III Discussed purposed work, in the Section-IV security analysis and discussion and finally paper concluded in the section-V with conclusion.

[2] **RELATED WORKS**

To address security and privacy a naïve approach is that, first data owner encrypt the data before storing on the untrusted cloud server using symmetric key and then supply the symmetric key to the authorized user, authorized user can download an decrypt the data with symmetric key. However, this solution is not suitable for cloud computing as to revoke access permission from an authorized user the data owner need to re-encrypt entire data and redistribute the new key to
each remaining legitimate user, as the number of user leave and join continuously this approach is impractical and inefficient because, this will impose huge burden on data owner as well as on the cloud server.

In the following section we have discussed some cryptographic schemes used by researcher to solve privacy and security issues like, Attribute Based Encryption, Proxy Re-Encryption and Hybrid ABE and PRE.

2.1 Attribute Based Encryption (ABE)

Attribute based encryption (ABE) [13] is an access control mechanism in which a number of attributes associated with the data or key and an access policy is defined on those attributes which also can be associated with data or key, if an access policy is satisfied then user can access data. Based on the association of access policy and attribute it can be divided into two category [4]

2.1.1 Key-Policy ABE (KP-ABE): In this access policy is associated with the private key of users and attributes are associated with the data, if access policy satisfies the attribute associated with the data then user can decrypt the data. [Figure-1] shows KP-ABE Scheme.

2.1.2 Ciphertext-Policy ABE (CP-ABE): In this access policy is associated with the data and attributes are associated with the private key of user, if access policy associated with ciphertext satisfy the attribute associated with the private key of the user, then user can decrypt the data. [Figure-2] shows CP-ABE Scheme.

![Figure 1: KP-ABE Scheme](image1.png)

![Figure 2: CP-ABE Scheme](image2.png)
2.2 Proxy Re-Encryption

This concept was introduced by Blaze at el. (14). In this cipher text encrypted under owners public key can be translated into another cipher text using re encryption key which can decrypted by the user secret key. In this approach semi trusted third party translates the cipher text using re-encryption without decrypting the cipher text, so third party (Proxy Server) can’t access the data in plain text format.

2.3 Hybrid ABE and PRE

ABE and PRE can be combined to design a scheme which will protect privacy and security of outsourced data and will be more effective and efficient.

2.4 Bilinear Map

Let $G_1, G_2, G_3$ is a cyclic group of order $q$ with $g_1$ and $g_2$ are the generator of group $G_1, G_2$ and $e: G_1 \times G_2 \rightarrow G_3$ is a bilinear map for all $a, b \in \mathbb{Z}_q$ with following characteristics

1. $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$
2. The map is non-degenerate i.e. if $g_1$ generate $G_1$, $g_2$ generate $G_2$ then $e(g_1, g_2)$ generate $G_3$.

3. $e$ must be efficiently computable.

In [15] Ateniese et al. purposed a proxy re-encryption scheme using bilinear map based on ElGamal encryption [16], which is unidirectional, Collusion resistant and do not required pre-sharing of secret key. Scheme detail purposed in [15] is as follows

Let $G_1, G_2$ is a cyclic group of order $q$, $g$ is a generator of Group $G_1$ and $e: G_1 \times G_1 \rightarrow G_2$ is a bilinear map.

Data owner private and public key pair are $sk_a = a \in Z_q^*$, $pk_a = g^a$, user private public key pair are $sk_b = b \in Z_q^*$, $pk_b = g^b$, $r \in Z_q^*$ is a random number and $e(g^r, g^b) = Z$ is a bilinear map.

Encryption for data $d$ is

$$C_a = (Z^r.d, g^{ar})$$

Decryption can be performed as

$$d = Z^r.d / e(g^a, g^{1/a}) = Z^r.d / Z^r$$

Re-encryption key for proxy is $rk_{a\rightarrow b} = (g^b)^{1/a} = g^{b/a}$.

Re-Encryption can be performed as

$$C_a = (Z^r.d, g^{ar})$$

$$C_b = (Z^r.d, e(g^r, rk_{a\rightarrow b}))$$

$$C_b = (Z^r.d, e(g^r, g^{b/a}))$$

$$C_b = (Z^r.d, Z^{rb})$$

The ciphertext $C_b$ can be decrypted using user secret key, which was initially encrypted with data owner public key.
Zhao et al. [6] purposes a scheme, Progressive elliptical curve encryption scheme, in this scheme data owner can encrypt the data using his public key before storing to the cloud server. If a user request data using his public key, the data owner will send a credential to the cloud server for re encryption of ciphertext and to the user for decryption the data. This scheme does not allow the cloud server to obtain the plain text but also have certain limitation, in this data owner has to be online for issuing credential which is in efficient for data owner point of view and also assume that cloud server share his private key with the data owner, this make this scheme impractical as cloud server never share his private key with the data owner.

Tu et al. [7] purposes an scheme based on CP-ABE, in this scheme a set of attribute associated with the private key of user has been assigned to the each users and disturbed to the corresponding users and access policy is associated with the data and defined by the data owner, if the user key satisfy access policy associated with the data he can access the data. Whenever a user access right is revoked, the data is reencrypted again rendering the attribute associated with revoked user does not satisfy new access policy. In this scheme there is no need to redistribute key after each revocation to the remaining user but updating the ciphertext policy after each user revocation placed heavy work load on the server, so this scheme is not efficient.

Tran et al. [8] purposed an scheme based on proxy re encryption to share social network data on the cloud, In this scheme each user uses same public key to encrypt the data and different secret key to decrypt the data. The data owner public key is divided into two parts one is stored on data owner machine and other on cloud server, and the user who is authorized to access the data have the secret key of which half is associated with cloud server and other half is associated with the user machine. The data owner first encrypts data with his half key and then sends to server where the server again encrypts with his part of half key. If user request a data the cloud server first decrypt the data with the half of user’s private key and then send to the user, now user can decrypt the data with his half key. The main strength of this scheme is , data remain encrypted during entire storage duration, if access rights of an user is revoked ,the cloud server simply remove his piece of key. The main drawback with this scheme is if an user and cloud server collude the user can send his half piece of private key to the cloud server and cloud server can reveals other users private key in his group, also there is one encryption and one decryption operation is involved with each access and which putted heavy loads on the cloud server.

[3] PURPOSED WORK

In our purposed system the participating entities are data owner, Cloud Service Provider (CSP) and a set of users. The data originator has the data which should be outsourced, The CSP provides cloud storage and proxy reencryption service to the user, and users are the entities to which data are shared.
Our data sharing scheme is based on symmetric key and asymmetric key encryption, symmetric key encryption is used to encrypt the data and asymmetric key encryption is used to encrypt keys. Select a key $k$ to encrypt data using symmetric key, Select another key $k_1$ and encrypt this key using attribute based encryption. compute $k_2 = k_1 \oplus k$ and encrypt $k_2$ using proxy re-encryption. When a user request for data access, CSP first check the authorized list, if valid CSP re-encrypt $k_2$ into another ciphertext using re-encryption key $rk_a \rightarrow b^*$, which can be decrypted by the user using his private key and he can obtain $k_2$, using his ABE key he can get $k_1$ if his attribute associated with key is satisfy access policy associated with ciphertext of $k_1$. Using $k_1$ and $k_2$ user can compute $k = k_1 \oplus k_2$, key $k$ can be used to decrypt and retrieve the data $d$.

3.1 Preliminaries

Let $d$ is the data to be share between users. $g$ be generator of group $G_1$ of prime order $q$. $sk_a = a \in Z_q^*$, $pk_a = g^a$ are the data owner private and public key pair, $sk_b = b \in Z_q^*$, $pk_b = g^b$ are user private and public key pair, $r \in Z_q^*$ is a random number and $e(g, g^r) = Z$ is a bilinear mapping $e: G_1 \times G_1 \rightarrow G_z$.

$PK, SK_m$ are public and master secret key pair used in ABE. $PK$ is used to encrypt data with associated access policy and $SK_m$ is used to create secret key $A.sk_b$ with a set of attribute associated with it.

3.2 ABE consist following function

$A.setup$: This function take a security parameter $1k$ as input and output a public key ($PK$) and secret key ($SK_m$).
**A.keygen:** This function takes master secret key \((SK_m)\), set of attribute corresponding to the user access privileged \((I)\) as input and output user secret key \((sk_b)\).

**A.enc:** This function takes public key \((PK)\), data to be encrypted \((d)\), access policy associated with the data \((A_{pol})\) as input and output \(c = A.enc_{pk}(A_{pol},d)\), where \(c\) is ciphertext.

**A.dec:** This function takes user secret key \((sk_a)\), ciphertext \((c)\) as input and output \((d)\), if \(sk_b\) attribute match with access policy associated with the data, False, if attribute associated with key does not match with access policy associated with the data.

### 3.3 Proxy Re-Encryption consist following function

**P.setup:** This function takes a security parameter \((1^k)\) as input and output description of plain text space \(pram\).

**P.keygan:** This function takes \((pram)\), user identification \((ID)\) as input and output public and private key pair \((pk_a,sk_a)\).

**P.reykey:** This function takes private key \((sk_a)\) of data owner (first party), public key \((pk_b)\) of user (second party), ciphertext \((C_a)\) under public key of \((pk_a)\) and output, re-encryption key \((rk_{a\rightarrow b})\) which transform ciphertext under public key of data owner \((pk_a)\) to the public key under user \((pk_b)\).

**P.enc:** This function takes public key \((pk_a)\) of data owner, plain text \(d\) as input and output \(C_b = P.enc_{pk}(d)\)

**P.reenc:** This function takes \(rk_{a\rightarrow b},C_a\) as input and output \(C_b = P.reenc_{pk}(C_a,rk_{a\rightarrow b})\)

**P.dec:** This function takes secret key of user \(sk_b\), re-encrypted ciphertext \((C_b)\) and output \(d = P.dec(C_b,sk_b)\).

### 3.4 Processing at client side

1. Choose a random key \(K_1\) and encrypt data \(d\) using symmetric key encryption.

\[C_a = E_k(d)\]
2. Let \( A_{pol} \) represent access policy associated with data \( d \), choose a random value \( K_1 \) and encrypt using access policy.

\[
C_b = A.enc_{PK}(A_{pol}, K_1)
\]

3. Compute \( k_2 = k \oplus k_1 \) and encrypt as

\[
C_c = P.enc_{pk}(k_2)
\]

\[
C_c = (Z', k_2, g'^m) \rightarrow \text{using [15] and also discussed in related part.}
\]

### 3.5 User Authorization

1. \( A.sk_b = A.keygen(SK_m, I) \), \( I \) is set of attribute associated with user.

2. Data owner send \( A.sk_b \) securely to the user.

3. use \( P.reykey \) to generate re-encryption key using private key of data owner \( (sk_d = a) \) and public key of user \( pk_b = g^b \).

\[
rk_{A \rightarrow B} = g^{b/a}
\]

For every authorized user a re-encryption key and user ID will be generated and stored on the secure CSP database.

\((pk_B, g^{b/a})\)

### 3.6 Data Request by an authorized user

1. To access data every user send a request to CSP in the form.

\[
pk_B, \text{Sign}(pk_B)
\]

Upon receiving request first CSP verify the identity of user in the authorization list if not exist abort. If found CSP will retrieve the corresponding re-encryption key and re-encrypt \( k_2 \) with re-encryption key as follows [15].

\[
C_c = (Z', k_2, g'^m)
\]

\[
C_c = (Z', k_2, e(g'^m, rk_{A \rightarrow B}))
\]
\[ C_c = (Z', k_2, e(g^a, g^{b^a})) \]
\[ C_c' = (Z', k_2, Z^{''b}) \]

and send \( \{C_a, C_b, C_c'\} \) to the user.

At the User Side, user perform following action to retrieve data

2. User decrypts \( C_a \) using \( A.sk_b \) and retrieves \( k_1 \).

3. Decrypt \( C_c' \) using \( sk_b \) and retrieve \( k_2 \).

4. \( k = k_1 \oplus k_2 \)

5. Decrypt \( C_a \) using \( k \) and retrieve the data \( d \).

3.7 User Revocation

To revoke a user from accessing data, data owner inform to CSP, to remove entry corresponding to the user from the authorization List.

3.8 Data Deletion

To remove data record from sharing, the data owners ask to remove the entire user corresponding to that data.

4] SECURITY ANALYSIS AND DISCUSSION

In this scheme, user revocation only required to delete the corresponding re-encryption key from the CSP, there is no need to re-encrypt the data or redistribute the key to other legitimate user.

Pre sharing of user secret key is not required as the data owner compute the re-encryption key \( (g^b)^{1/a} \), which require only public key \( g^b \) of user, the re-encryption key will be stored on the CSP which is used in re-encryption of data which was initially encrypted by data owner public key but after re-encryption can be decrypted by user’s secret key.

There is no way to access the data by CSP as the data remain encrypted using symmetric key during entire duration of storage and transmission.

This scheme is based on a PRE scheme [15] which is unidirectional under the inverse exponent assumption (equivalent to the Diffie Hellman assumption), the proxy can’t calculate \( g^{a/b} \) from
\( g^{b/a} \). So it can’t be used to re-encrypt user data in such a way that it can be decrypted by data owner.

CSP can’t collude with the data owner to extract the user’s private key as private key of user b can’t be extracted from \( g^{b/a} \).

This scheme, support stateless cloud as there is no need to retain any information for revoked user by the CSP.

[5] CONCLUSION

Outsourcing data to the cloud brings many benefits but also raise much concern to the Privacy and security of data. The data are stored on the cloud server, a malicious cloud service provider may access the data without data owner concern, one way to solve this problem to encrypt the data before storage. However, encryption of data prior to storage solves privacy and security concern but it raises other concern. If data owner willing to revoke access permission due to user may be employee and he left the organization, he has to re-encrypt entire data again and has to redistribute the new key to remaining legitimate user, which requires lots of resources for computation and lot of bandwidth for the redistribution of key. In our scheme we have solve these problem using a combination of asymmetric key and symmetric key.
REFERENCES:


