Achieving Privacy-Preserving CP-ABE Access Control with Multi-Cloud

Conference Paper · December 2018

DOI: 10.1109/BDCloud.2018.00120

5 authors, including:

Chunhua Li
Huazhong University of Science and Technology
14 PUBLICATIONS **77** CITATIONS
SEE PROFILE

Ke Zhou
Huazhong University of Science and Technology
146 PUBLICATIONS **980** CITATIONS
SEE PROFILE

2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications

Achieving Privacy-Preserving CP-ABE Access Control with Multi-Cloud

Chunhua Li, Jinbiao He, Cheng Lei, Chan Guo, Ke Zhou

Wuhan National Lab for Optoelectronics
Key Laboratory of Information Storage System, Ministry of Education of China
School of Computer Science & Technology, Huazhong University of Science & Technology
Wuhan, China
li.chunhua@hust.edu.cn

***Abstract*—Cloud storage service makes it very convenient for people to access and share data. At the same time, the confidentiality and privacy of user data is also facing great challenges. Ciphertext-Policy Attribute-Based Encryption (CP-ABE) scheme is widely considered to be the most suitable security access control technology for cloud storage environment. Aiming at the problem of privacy leakage caused by single-cloud CP-ABE which is commonly adopted in the current schemes, this paper proposes a privacy-preserving CP-ABE access control scheme using multi-cloud architecture. By improving the traditional CP-ABE algorithm and introducing a proxy to cut the user's private key, it can ensure that only a part of the user attribute set can be obtained by a single cloud, which effectively protects the privacy of user attributes. Meanwhile, the intermediate logical structure of the access policy tree is stored in proxy, and only the leaf node information is stored in the ciphertext, which effectively protects the privacy of the access policy. Security analysis shows that our scheme is effective against replay and man-in-the-middle attacks, as well as user collusion attack. Experimental results also demonstrates that the multi-cloud CP-ABE does not significantly increase the overhead of storage and encryption compared to the single cloud scheme, but the access control overhead decreases as the number of clouds increases. When the access policy is expressed with a AND gate structure, the decryption overhead is obviously less than that of a single cloud environment.**

Keywords—Cloud storage, access control, privacy-preserving, CP-ABE, multi-cloud

I. INTRODUCTION

With the rapid development of cloud storage, the public's concerns about cloud security are continuously increasing. According to Crowd Research in *2017 Cloud Security Report*, 49% of users are concerned about data privacy threatened, 47% of users worry that cloud service providers (CSPs) cannot enforce strict confidentiality rules, and nearly a third of business plan to significantly increase their research budgets for cloud security technology. Therefore, using effective measures to protect the confidentiality and privacy of users' data has become an urgent security issue in the development of cloud storage [1].

Access control technology is an effective measure to protect data security, and Ciphertext-Policy ABE (CP-ABE) scheme is recognized as the most ideal access control for cloud storage service [2]. In the CP-ABE scheme, the access control policy, which consists of logical combinations of user's attributes, is specified by data owner and embedded in ciphertext in clear text. A series of attribute sets which describe user's characteristics are embedded in user's private key, as long as the attributes set satisfies the structure of access control policy, the user has access rights and can decrypt the ciphertext successfully. It can be seen that CP-ABE is a one-to-many cryptographic access control.

In recent years, researches on CP-ABE mainly focus on policy and ciphertext update [3,4], constant-size key and ciphertext [5,6], attribute revocation [7,8], etc. With the frequent leakage of personal privacy, both CSPs and users have paid more and more attention to privacy protection. In the academic, some CP-ABE studies supporting privacy preserving have been emerged successively.

To prevent CSP from speculating on user privacy through access control policies, some researches use wildcards to hide the initial attributes in policies [9,10]. Specifically, they expand the attributes or policies to the entire attribute space when generating user's private keys and accessing control policies by multiplying all the attributes iteratively. Once an attribute does not meet the policy, the result of multiplying wouldn't be consistent, hence to ensure the validity of the encryption and decryption operations. However, extending to the entire attribute space results in a high storage and computational cost of redundant attributes. In addition, access control schemes supporting privacy protection often adopts an AND-gate structure to describe the logical combination of attributes in policies [11], which limits the flexibility of policy expression.

In current CP-ABE schemes, in order to successfully perform the decryption operation, the access control policy is generally hidden in the header of ciphertext as a decryption parameter in plaintext. Similarly, the set of user attributes is put in the header of user's private key in plaintext. Due to the high complexity of CP-ABE algorithm, decryption operation is often performed at the agent or the cloud which can significantly improve the efficiency of access control [12]. If the cloud is used for decryption, not only will the user's attributes be fully exposed to the cloud, but also the privacy information hidden in the access control policy will be speculated by the malicious cloud [13]. For example, a hospital publishes an access control policy for some patient files, only medical staff whose work id number is in a scope can access it. According to the investigation, all the doctors in

978-1-7281-1141-4/18/$31.00 ©2018 IEEE
DOI 10.1109/BDCloud.2018.00120

this range belong to the cardiology department, so it can be inferred that this document is related to heart disease, leading to the disclosure of patients' privacy.

Considering that a single cloud can obtain all attributes sets and complete access control structure, resulting in privacy leakage problems, this paper proposes a privacy preserving multi-cloud CP-ABE access control scheme, which improves the traditional CP-ABE algorithm by introducing user's identity information. In order to protect the privacy of user attributes, a proxy server is introduced to cut the user's attribute private key, which can ensure that only a part of the user's attribute set can be gotten by a single cloud. In order to prevent CSP from speculating on user privacy through access control policies, we delegate the proxy to save the intermediate structure of access control policy and only leaf nodes information is kept in the cloud.

Our contributions can be summarized as follows:

(1) We utilize multiple clouds to host user attributes, each cloud can only possess part of user attributes and all attributes in different sub-clouds are disjoint. This multi-cloud access control scheme not only protects the privacy of user attributes, but also increases fault tolerance. Meanwhile, multi-cloud architecture supports parallel operations, greatly reducing the overhead of decryption and access control.

(2) We introduce the proxy server to split the access control policy, isolating the logical structure and user attributes for storage, which makes it impossible for the sub-cloud to obtain the complete access policy from the ciphertext, effectively protecting the privacy of the access policy.

(3) We improve the original CP-ABE algorithm and embed a unique global identifier (GID), only the initiator of the access request could decrypt the ciphertext, effectively resisting the attack of replay and man-in-the-middle, as well as user collusion attack.

(4) Compared with the single cloud, the multi-cloud CP-ABE scheme enables the decryption of access control policies to be performed in parallel, which greatly improves the efficiency of decryption and access control with AND-gate structure policy.

The rest of the paper is organized as follows. Section II introduces the related work. Section III describes preliminary knowledge related to our scheme. Section IV presents our multi-cloud hosting, system model and security assumption. Section V details our proposed scheme. Security properties are covered in Section VI. The performance of our proposed scheme is elaborated in Section VII. Conclusion is discussed in Section VIII.

II. Related Work

Bethencourt et al. [2] proposed the Ciphertext-Policy Attribute-Based Encryption (CP-ABE) scheme in 2007, in which the ciphertexts are associated with an access control structure and the users' keys are associated with a set of attributes. Although the CP-ABE scheme has the characteristic of one-to-multiple encryption access control, which is very suitable for file sharing scenarios in cloud, it exposes the set of users' attributes and access control policies resulting in privacy leakage.

The multi-authority can effectively prevent a single authority from acquiring the set of user attributes. Jung et al. [14] divided the entire set of attributes into N disjoint subsets, each subset is certified by a certification authority, so each certification authority knows only part of user's attributes. On the other hand, they use the OT (Oblivious Transfer) algorithm to achieve the anonymization of private key transmission, further to enhance the user privacy. However, the complex interactive protocols used in this scheme increase the communication overhead. Furthermore, the OT algorithm cannot guarantee that users can correctly select the private key corresponding to their own attributes, which easily leads to the abuse of the private key. Yang et al. [15] proposed a scheme that does not require a global authority, they introduce a certificate authority (CA) to assign a global user identifier (UID) for each user and an authority identifier (AID) for each authority. When the private key is associated with user's attribute, it will replace the random number with the global UID, so the private keys with the same UID can be integrated. It solves the problem that multiple private keys are independent of each other due to different random numbers. Xue et al. [16] proposed a multi-authority attributes management model, which adds a parameter tracking mechanism during key generation. When the system finds the illegal attribute key, the auditor can reversely obtain the authority that generate the key through the tracking algorithm, and then to detect which attribute authority incorrectly or maliciously performed the validation.

In the CP-ABE scheme, the access control policy is stored in plaintext in the ciphertext's header, which may result in the privacy leakage of the access control policy. In [11], Zhou et al. proposed a Privacy Preserving Attribute-Based Broadcast Encryption (PP-CP-ABE). Prior to the release of the access control policy, the attributes required by the policy are obfuscated. These attributes are all represented by the same wildcard when the attributes with wildcards in the access policy are retained. Thus, there are 2N possible access policies during decryption (N is the number of wildcards) that protect the access policy. However, this scheme needs a large amount of space to store the redundant information and has a high computation cost because the attributes or policies need to be extended to the entire attribute sets when generating user's private keys and accessing control policies. Ning et al. [12] modified the CP-ABE process and introduced the public and private key for CSP and user to ensure the user's private key not to be leaked. Han et al [10] proposed a Privacy-Preserving Decentralized CP-ABE (PPDCP-ABE) scheme. In this scheme, each authority can work independently and supervise its own distribution of attribute sets and private keys. For the conspiracy attack, it binds the user's private key to the GID and uses the anonymous signature when the user obtains the private key, so the authorization center cannot obtain the GID and user attribute.

All above schemes are based on a single cloud environment, for multi-cloud environment, Secure Storage in Multi-Cloud

Environment (SSME) model was proposed by Pietro et al. [17] in2018. To ensure the confidentiality and integrity of the data, they combined the symmetric and asymmetric encryption algorithms at client-side and used a worldwide distributed middleware for data splitting, dissemination and retrieval. However, the cost of key management in this scheme is quite considerable.

III. PRELIMINARIES

A. Bilinear Map

A bilinear map is a map e: $G_1 \times G_2 \rightarrow G_T$, where G_1 and G_2 are two Gap Diffie-Hellman groups of prime order p, G_T is another multiplicative cyclic groups with the same order. A bilinear map has the following three properties:

- *Bilinearity*: for all $u, v \in$G and $a, b \in Z_p$, $e(u^a, v^b) = e(v, u)^{ab}$.
- *Symmetry*: for all $u, v \in$ G, $e(u, v) = e(v, u)$.
- Non-degenerate: $e(g_1, g_2) \neq 1$, where g_1, g_2 are generators of G_1 and G_2.

The bilinear pairs can be constructed by the Weil on the elliptic curve or the Tate pair.

B. CP-ABE

In the CP-ABE scheme [2], a series of attribute sets that describing user characteristics information are embedded into the user's private key, while the access control policies formulated by the data owner are embedded into the ciphertext. The algorithm consists of the following four parts:

(1) Server Initialization

In the initialization phase, there are multiplication cycle group of G_0, G_1, prime order p, and a bilinear map $e: G_0 \times G_0 \rightarrow G_1$, where the generating element of G_0 is g. The server generates the master key MK and public parameters PK by selecting random numbers $\alpha, \beta \in Z_p$. Then the server keeps MK and distributes PK to the legitimate users in the system. The public key is published as:

$$PK = \left\{ \mathbb{G}_0, g, h = g^{\beta}, f = g^{\frac{1}{\beta}}, e(g,g)^{\alpha} \right\}$$

The master key is : $MK = \{\beta,\ g^{\alpha}\}$.

(2) Generating Private Key

The user provides a set of attributes, and the server generates a user secret key. The generation rule is *SK = GenerateUserKey(MK, S)*, where S represents the attribute set submitted by the user. Server chooses the random number r, ($r \in Z_p$), then for each attribute $j \in S$, chooses a random number $r_j \in \mathbb{Z}_p$, calculates the attribute secret keys D_j and D_j'. Last, it combines all attribute private keys to form the user secret key SK and returns SK to user. The secret key is of the form:

$$SK = \left\{ D = g^{\frac{\alpha+\beta}{\beta}}, \forall j \in S: D_j = g^r \cdot H(j)^{r_j}, D_j' = g^{r_j} \right\}$$

(3) Encrypting plaintext

Encrypting the plaintext according to the access control policy provided by the user and generate the ciphertext. The generation rule is *CT = Encrypt(PK, M, T)*, where M represents the plaintext data and T represents the access control structure, in the form of an access control tree. Starting from the root node R, choose random number $s \in Z_p$ to make $q_R(0) = s$, and then select other nodes d_R randomly to reconstruct $q_R$polynomial. For any other node y, let $q_y(0) = q_{parent(y)}(index(y))$, and then choose other nodes d_y according to Lagrange interpolation to reconstruct q_y polynomial. Ciphertext CT is obtained after encryption of plaintext data M. The ciphertext CT is constructed by the following equation:

$$CT = \left\{ \begin{matrix} T, \tilde{C} = Me(g,g)^{\alpha s}, C = h^s, \\ \forall y \in Y: C_y = g^{q_y(0)}, C_y' = H(att(y))^{q_y(0)} \end{matrix} \right\}$$

(4) Decrypting ciphertext

User receives ciphertext CT from the server and decrypts ciphertext by using its own private key SK to obtain the plaintext data M. The decryption rule is *M = Decrypt(CT, SK)*. The decryption process is described as follows:

$$\begin{aligned} &\tilde{C} / \left(\frac{e\left(g^r \cdot H(i)^{r_i}, g^{q_x(0)}\right)}{e\left(g^r, H(i)^{q_x(0)}\right)} \right) \\ &= \tilde{C} / \left(\frac{e\left(g^r \cdot g^{\delta \cdot r_i}, g^{q_x(0)}\right)}{e\left(g^r, g^{\delta \cdot q_x(0)}\right)} \right) \\ &= \tilde{C} / \left(\frac{e(g,g)^{r \cdot q_x(0) + \delta \cdot r_i \cdot q_x(0)}}{e(g,g)^{r_i \cdot \delta \cdot q_x(0)}} \right) \\ &= \tilde{C} / e(g,g)^{r \cdot q_x(0)} = M \end{aligned}$$

C. Access Control Structure

BSW CP-ABE access control tree is adopted in this paper [2]. For instance, assuming that a file can only be accessed by the teacher of Huazhong University (hust) or Wuhan University (whu), then the access control policy can be expressed as *((hust OR whu) AND teacher)*, the corresponding access control tree is shown in Fig.1. The leaf nodes in access control tree represent attribute values, and the non-leaf nodes represent the logical combination relation of its child nodes. We can achieve different combinations by setting thresholds for non-leaf nodes, i.e. for non-leaf node x, suppose its number of child nodes is num_x, its threshold value k_x satisfies $0 < k_x \leq num_x$. When $k_x = 1$, it means that x is an OR gate; when $k_x = num_x$, it means that x is an AND gate. For all child nodes of x, numbering them from 1 to *num*, function $index(y)$ returns the index of the child nodes y.

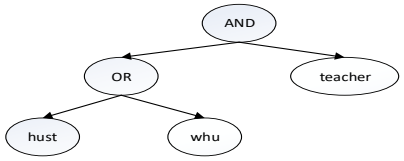


Fig.1 Access control tree for policy *(hust OR whu) AND teacher*

For any node x, $parent(x)$ represents its parent, $att(x)$ represents user's attribute when x is a leaf node, and the hash

function $H:\{0,1\}^* \to G_0$ maps any binary attribute string to the multiplication cycle group G_0.

Let T be a tree whose root node is r. T_x represents a subtree of T, and x is the root node of T_x. If the attribute set S satisfies T_x, we set $T_x(S) = 1$. If x is a non-leaf node, we judge all the leaf nodes x' of x and then get each $T_{x'}(S)$, $T_x(S)$ returns 1 if and only if at least k_x child nodes return 1, otherwise it returns $\perp$. If x is a leaf node, $T_x(S)$ returns 1 if and only if $\mathrm{att}(x) \in S$, otherwise it returns $\perp$. Recursively until the root node r returns.

IV. Definitions of Multi-Cloud CP-ABE Scheme

A. Multi-Cloud Hosting

In the CP-ABE algorithm, the access control policy is generally stored in the ciphertext header in clear text, and the user attribute set is also stored in the header of user's secret key in clear text. If we decrypt the ciphertext at the cloud, access control structure and user attribute set will be fully exposed to a single cloud, therefore, user's privacy will suffer a great risk of leakage. In this paper, we employ a multi-cloud architecture to host user attributes, each cloud can only possess part of user attributes and all attributes in different clouds are disjoint, namely, $Attrs\mathcal{C}_i \cap Attrs\mathcal{C}_j = \emptyset,\ i \neq j$.

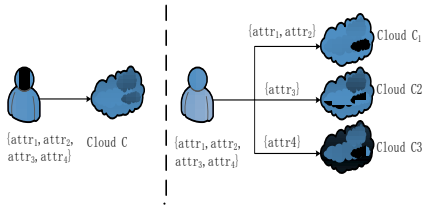


Fig.2 The schematic of multi-cloud hosting

As shown in Fig.2, assuming that there are three clouds $\mathcal{C}_1$,$\mathcal{C}_2$,$\mathcal{C}_3$, where $\mathcal{C}_1$ hosts attribute *attr*1 and *attr*2, $\mathcal{C}_2$ hosts attribute *attr*3, and $\mathcal{C}_3$ hosts attribute *attr*4, all above attributes are mutually exclusive. Considering a set of user attribute is {*attr*1, *attr*2, *attr*3, *attr*4}, then the set will be cut by the Proxy to ensure that each cloud can only obtain a portion of the set corresponding to its governed attributes. That is, $\mathcal{C}_1$ can only get attribute *attr1* and *attr*2, $\mathcal{C}_2$ can only get the attribute *attr*3, and $\mathcal{C}_3$ can only get the attribute *attr*4, which ensuring that any single cloud cannot get the complete set of user attributes. Besides, the multi-cloud architecture allows multiple copies of the same ciphertext in different clouds, thereby providing better redundancy and fault tolerance mechanisms.

B. System Architecture

Fig.3 shows our overall system architecture. It consists of four parts: Attribute Authority (AA), Client, Proxy and Cloud servers. A proxy server is introduced between the Client and Cloud. On the one hand, the private key of user attributes is cut through the Proxy to ensure that only part of user attributes can be acquired in a sub-cloud, effectively protecting the privacy of user attributes. On the other hand, the intermediate logical structure of access control tree is saved at the Proxy, only leaf nodes are pushed to the Cloud, thus effectively protecting the privacy of access control policy.

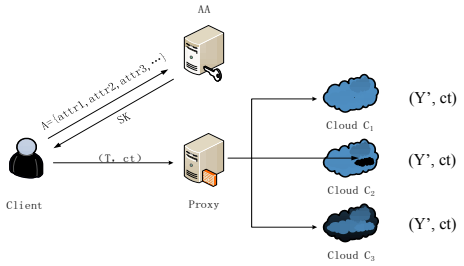


Fig.3 System architecture

Attribute authority (AA). *AA* is used to process the user's request for private key according to the user's attributes, and issues the attribute private key *SK* to the user.

Client. When a user uploads a file, the Client encrypts the file and uploads it to the Proxy. When a user downloads a file, the Client sends *SK* to the Proxy and obtains the intermediate decryption result from Proxy, then Client uses the independent private key to perform the final decryption operation.

Proxy server. When receiving a ciphertext *ct* from Client, Proxy splits the access policy *T* hidden in the ciphertext header into two parts: logical structure and leaf nodes, the former is saved at Proxy, and the latter are then sent to the Cloud with ciphertext. When receiving *SK* from Client, Proxy cuts *SK* and distributes them to different sub-clouds. After receiving the calculation result of leaf nodes returned by each sub-cloud, the Proxy integrates them and restores the secret parameters of root node to obtain the intermediate decryption result, and then returns it to the Client along with ciphertext.

Cloud. Each sub-cloud hosts part of user attribute set and stores the file uploaded by the user. When a user downloads a file, the corresponding leaf node information is calculated at Cloud according to the partial attributes and returns the result to Proxy along with ciphertext.

C. Security Assumption

Semi-trusted Server. We assume that the cloud server is honest and curious. That means, the cloud will faithfully store user data and reliably perform user-initiated operation requests. However, it still maintains enough interest in the user's data stored in the cloud and may peek on user's data information. For example, the access control structure in the ciphertext header may be snooped by the cloud, thus leading to privacy leaks.

Trusted Attribute Authority. In this paper, the attribute authority (*AA*) is introduced to generate the public key and issue the attribute private key to users. We assume that *AA* is trustworthy and communicates with users over a secure channel. Besides, *AA* will never collude with users to issue false private keys.

V. Privacy-Preserving Multi-Cloud CP-ABE Scheme

In this section, we will describe our proposed multi-cloud CAP-ABE scheme for privacy-preserving in detail.

A. Setup

Considering multiplication cycle groups G_0, G_1 with prime order p, and bilinear mapping $e: G_0 \times G_0 \to G_1$, where the

generator of G_0 is g. Randomly choosing $\alpha, \beta \in Z_p$, the public key PK and the master key MK is defined as follows respectively:

$$PK = \left(G_0, g, h = g^{\beta}, e(g,g)^{\alpha}\right)$$
$$MK = (\beta, g^{\alpha}) \quad (1)$$

B. KeyGen(MK, S, ID_{sk})

According to the attribute set S submitted by user, the system generates the corresponding attribute private key SK. Every valid user has a unique global identifier (GID), when a user is successfully registered, AA generates a GID for the user and sends the corresponding $H(GID)$ to the user as an independent private key ID_{sk}. The ID_{sk} is only owned by AA and the user. Randomly selecting $r \in Z_p$, for each attribute $j \in S$, selecting a random number $r_j \in Z_p$, the structure of attribute private key is defined as follows:

$$SK = \begin{Bmatrix} D = g^{(\alpha+r)/\beta}, \\ \forall j \in S: \ D_j = g^{r} \cdot H(j)^{r_j} \cdot H(GID)^{\beta}, D_j' = g^{r_j} \end{Bmatrix} \quad (2)$$

Compared to the original CP-ABE algorithm, the proposed algorithm introduces the GID into SK structure which can ensure that the user's attribute private key D_j differs even if the random number r_j is the same for the same attribute, further ensuring the uniqueness of the attribute private key. In addition, as can be seen from the later analysis, the introduction of GID makes it possible for the intermediate decryption result to be successfully decrypted only by the initiator of the decryption request, while others cannot recover the clear text even if they acquire the intermediate decryption result. Therefore, the security of decryption operation can still be ensured even if the proxy is illegally controlled.

C. Encrypt(PK, M, T)

Encrypting the plaintext M to generate ciphertext CT according to the access control tree T. Starting with the root node, select a random number $s \in Z_p$ to make $q_R(0) = s$, then pick the number of d_R child nodes to reconstruct the polynomial q_R. For any other child node x, let $q_x(0) = q_{parent(x)}(index(x))$, and choose other different d_x nodes to reconstruct the polynomial q_x. Let Y be the set of leaf nodes in T. The structure of CT is as follows:

$$CT = \{T, ct\}$$
$$ct = \begin{Bmatrix} \tilde{C} = Me(g,g)^{\alpha s}, C = h^{s}, \\ \forall y \in Y: C_y = g^{q_y(0)}, C_y' = H(att(y))^{q_y(0)} \end{Bmatrix} \quad (3)$$

The Client sends the ciphertext CT to the Proxy, and then Proxy deals with the access control tree T contained in CT header. As shown in Fig.4, assuming that the access tree T = *((AttrA OR AttrB) AND (AttrC OR AttrD))*, the set of leaf nodes Y = {*AttrA, AttrB, AttrC, AttrD*}. The Proxy numbers the leaf nodes in order from left to right in tree T to form a new set of leaf nodes Y' = {*AttrA-1, AttrB-2, AttrC-3, AttrD-4*}. Replacing tree T with Y' in CT to form a new ciphertext CT', ie. CT'= {Y', ct} (seen in Fig.3), and then CT' is sent to each sub-cloud, intermediate nodes and their logical combination T'=*((OR) AND (OR)* are stored at Proxy. Further, in order to prevent the Cloud from guessing the possible combination of leaf nodes according to the set Y', some redundant leaf nodes can be added into Y' to increase the difficulty of cloud cracking, and the subsequent decryption will not be affected.

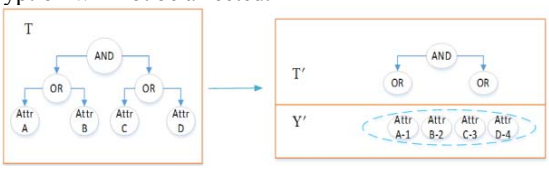


Fig.4 Cutting process of access control tree

Since the cloud can only obtain the information of leaf nodes and cannot acquire the combination of these leaf nodes, it is unable to get complete or partial access control policy. Besides, even if the Proxy is broken, the attacker can only obtain the logical combination of the intermediate nodes, without the knowledge of the underlying specific leaf information. Therefore, neither Cloud nor Proxy can obtain effective information about the access control policy, thus guaranteeing the privacy of access control policy.

D. Decrypt(CT, SK, ID_{sk})

When the user needs to decrypt the file, the Client sends the user's attributes set S and the private key SK to the Proxy, and then S is split into the subset SC_i of the $AttrsC_i$ according to the attribute set $AttrsC_i$ hosted by the Cloud C_i, and SK is also split into SKC_i accordingly. SC_i and SKC_i are then sent to the Cloud C_i together (as shown in Fig.5).

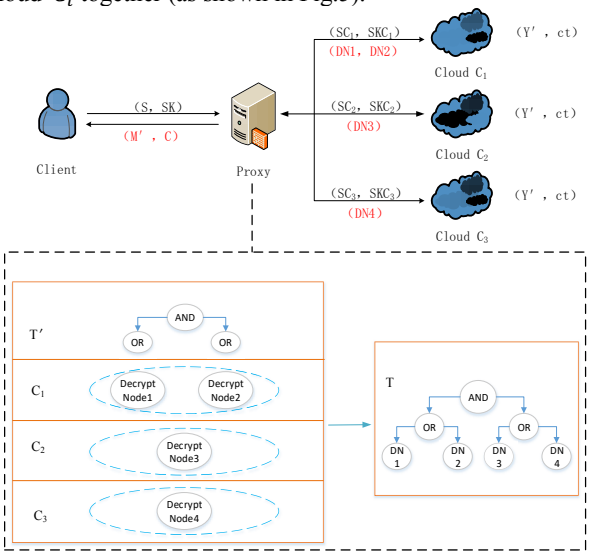


Fig.5 Decryption process of leaf nodes in the Cloud

After receiving the Proxy's request, C_i traverses the set Y'. For $\forall x \in Y'$, let $i = att(x)$, if x $\in SC_i$, then the leaf node x is decrypted with the equation (4).

$$DecryptNode(CT', SK, x) = \frac{e(D_i, C_x)}{e(D_{i'}, C_{x'})}$$
$$= \frac{e\left(g^{r} \cdot H(i)^{r_i} \cdot H(GID)^{\beta} \cdot g^{q_x(0)}\right)}{e\left(g^{r_i}, H(i)^{q_x(0)}\right)}$$
$$= e\left(g^{r} \cdot H(GID)^{\beta} \cdot g^{q_x(0)}\right) \quad (4)$$

Each sub-cloud C_i sends all calculated results $\{DN_i\}$ to the Proxy, Proxy then reconstructs the access control tree T based on the number of leaves. For the intermediate node z whose threshold value is K_z, Proxy traverse its child nodes to find the number of K_z calculated results, the corresponding leaf nodes constitute a set S_z. If there is no subset matching to the set S_z in user attribute set S, it indicates that S does not satisfy the access control tree T.

Suppose $i = index(\mathrm{x}),\ S_{z'} = \{index(x): x \in S_z\}$. The intermediate result F_z is calculated as follows:

$$F_z = \prod_{x \in S_z} DN_x^{i,S_{z'}(0)} = \prod_{x \in S_z} \left(e\left(g^r \cdot H(GID)^\beta, g\right)^{q_x(0)}\right)^{\Delta_{i,S_{z'}}(0)}$$

$$= \prod_{x \in S_z} \left(e\left(g^r \cdot H(GID)^\beta, g\right)^{q_{parent(x)}(index(x))}\right)^{\Delta_{i,S_{z'}}(0)}$$

$$= \prod_{x \in S_z} e\left(g^r \cdot H(GID)^\beta, g\right)^{q_z(i)\cdot\Delta_{i,S_{z'}}(0)}$$

$$= e\left(g^r \cdot H(GID)^\beta, g\right)^{q_z(0)} \quad (5)$$

Do recursive operation from the intermediate node to the root node and get the calculation result of root node:

$$\mathrm{Key} = \mathrm{e}\left(g^r \cdot H(GID)^\beta, g\right)^{q_R(0)}$$

$$= \mathrm{e}\left(g^r \cdot H(GID)^\beta, g\right)^s$$

Then Proxy calculates the intermediate decryption result M':

$$M' = \tilde{C}/(e(C,D)/Key)$$

$$= \frac{Me(g,g)^{\alpha s}}{(e(h^s, g^{\alpha+\gamma/\beta})/e(g^r \cdot H(GID)^\beta, g)^s)}$$

$$= M \cdot \frac{e(g,g)^{\alpha s} \cdot e\left(g^r \cdot H(GID)^\beta, g\right)^s}{e(g,g)^{(\alpha+\gamma)/s}}$$

$$= M \cdot e\left(H(GID)^\beta, g\right)^s$$

Proxy sends *M'* and the parameter *C* in *ct* to the Client, and the Client will perform the final decryption.

$$\frac{M'}{e(H(GID), C)} = \frac{M \cdot e(H(GID)^\beta, g)^s}{e(H(GID), g^{\beta s})} = M$$

Due to the introduction of independent private key *H* (*GID*), the decryption of ciphertext can only be performed by the initiator of decryption request. Since the *GID* is unknown, even if the Proxy is compromised, the attacker obtains the ciphertext and intermediate calculation results, he cannot perform the final decryption to get the plaintext, thus improving the security of ciphertext and ensuring the uniqueness of the decryption operation. Meanwhile, the user's unique identity *GID* is embedded in the attribute private key, which makes the attribute private key of different users different even for the same attribute. Since the *GID* is unique, the collusive user is unable to obtain the calculation results of intermediate nodes or even root node with Lagrange interpolation method, thus effectively resisting the user attribute collusion attack.

VI. Security Analysis

A. Privacy Protection for User Attributes

When a user sends his attribute private key to the Proxy, in order to prevent the Cloud from gaining the complete attribute set, the Proxy will split the user attribute set kept in the header of private key according to the attribute set hosted by different clouds. Therefore, each cloud only possesses a portion of user's attributes, thus preventing the full disclosure of user's personal information. Unless the Proxy is compromised or all of the cloud colludes, the privacy of user attributes will not be leaked.

B. Privacy Protection for Access Control Policy

When a user uploads the ciphertext to the Proxy for the first time, Proxy will split the access control tree kept in the header of ciphertext into two parts: leaf node and logical relationship. All leaf nodes are restored in ciphertext while logical relationships are kept in the Proxy, so each sub-cloud can only obtain the leaf nodes. First of all, the attribute sets hosted by each cloud are mutually disjoint, so a single cloud cannot identify the attributes that do not belong to its cloud in general. Secondly, suppose that there are N leaf nodes, and the access control policy is expressed with AND gate structure, so there are 2^N types of access control trees that need to be guessed, and the guessing difficulty increases exponentially with the number of leaf nodes. If the access control policy adopts the OR gate structure, the difficulty of guessing will be sharply increased, thus protecting the privacy of access control policy. Unless the Proxy is breached at the first time the user uploads the ciphertext, the access control policy will be exposed. At any time after that, neither the Proxy nor the Cloud has a chance to obtain a complete access control policy.

C. Man-in-the-middle Attacks and Replay Attacks

In this paper, *GID* is embedded into each user's attribute private key. According to the CP-ABE's algorithm flow, the decryption information of leaf node calculated by the Cloud must contain the user's *GID* information when a user initiates a decryption request. Then the Proxy restores the secret information in root node according to the leaf nodes, which still contains *GID* information. Since only the request initiator owns the *GID* information, the intermediate decryption result from Proxy only can be decrypted by the initiator. Obviously, it can effectively resist against man-in-the-middle attacks and replay attacks.

D. User *Collusion Attack*

In the BSW CP-ABE scheme [2], random numbers are introduced when generating the attribute private key for each user attribute to ensure that the same attribute of different users corresponds to different attribute private key. Our scheme introduces *GID* into the private key of user attributes, which ensures that the user's attribute private key D_j is different even if the generated random number r_j is the same for the same attribute, and further ensures the uniqueness of the private key. Therefore, when different users combine the attribute private keys, the secret information in root node still cannot be solved by using the Lagrange interpolation method since different GID numbers cannot be eliminated, thereby effectively resisting the user's collusion attack.

VII. Performance Analysis and Evaluation

We implement the prototype system of our scheme based on the Swift Object Storage System (Swift 1.11.0), and analyze the overhead of storage space, encryption/decryption operation and access control. All experiments were conducted on a Linux Server with Inter Xeon(R) E5405, run at 64-bit Ubuntu 14.04 LTS. Encryption in CP-ABE is implemented based on cryptography library PBC, GMP and OPENSSL, and 128-bit encryption is used in AES.

A. Storage Overhead

In our scheme, the storage objects mainly involve user's private key, ciphertext and intermediate structure of access control policy, so we will analyze the storage overhead around the above three aspects.

Storage Overhead for Private Key. In our scheme, each user attribute is added with a cloud identifier to indicate which cloud is hosting the attribute. Obviously, the extra overhead of a cloud identifier is almost negligible compared to the long attribute private key. Besides, the amount of space required to store the private key is only related to the number of user attribute and independent of the number of clouds. Tab.1 lists the storage space for private key when the number of attributes varies from 1 to 16.

Table I The length of private key (bytes)

attribute number	1	2	4	8	16
storage space	338	553	983	1843	3581

Storage Overhead for Ciphertext. In order to protect the privacy of access control policy, the logical structure of access policy is removed and kept in the proxy, only leaf nodes is preserved in ciphertext. Therefore, compared to a single cloud scenario, the storage cost of ciphertext per cloud is much smaller in multi-cloud, as shown in Tab.2. However, extra space is needed to store the intermediate structure in the proxy. With the increasing of leaf nodes, the logical structure of access policy becomes more complex, the required space in proxy is also greater, but the storage overhead of each cloud will decrease.

Table II Storage overhead for ciphertext (bytes)

the number of attributes	**the length of ciphertext**		**the length of intermediate structure**
	single cloud	***two clouds***	
1	530	530	5
2	746	739	13
4	1180	1157	31
6	1615	1575	50
8	2044	1993	69
16	3794	3689	151

B. Encryption Overhead

Since the encryption process ends after the proxy reprocesses the ciphertext, it has nothing to do with the number of clouds. Here, we select two clouds for comparison test with the original CP-ABE single cloud scheme, as shown in Fig.6. It can be seen that the encryption time will increase when the number of user attributes increases, because the Lagrange interpolation corresponding to each attribute is calculated one by one during encrypting. Meanwhile, we can find the encryption time in multi-cloud is a little longer than that of in one-cloud. The reason for this is that it needs additional operations to split the policy and repackage the ciphertext. As the number of attributes increases, the split takes more time but at an acceptable level of milliseconds.

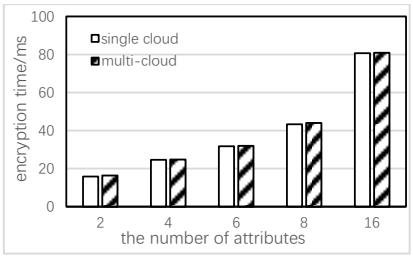
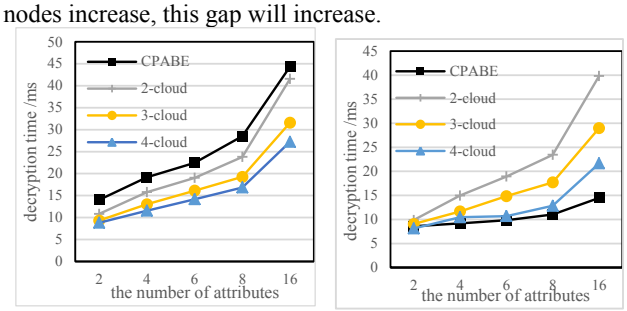


Fig.6 Encryption overhead

C. Decryption Overhead

In the process of decryption, the Lagrangian interpolation is first calculated based on the information saved by leaf nodes through bilinear pairings, and then the power exponent of each intermediate node is calculated successively, and finally the secret information stored by root node is obtained through recursion operation.

As shown in Fig.7, for the 'AND' policy structure, such as '(attr0 and attr1) and (attr2 and attr3)', the left and right child nodes are calculated in turn during decryption. So all leaf nodes need to be calculated from left to right in a single cloud environment. In the case of multi-cloud environment, each sub-cloud is responsible for calculating its hosting attribute leaves which means the decryption operation is performed in parallel. Therefore, the decryption speed is faster in multi-cloud than that of CP-ABE scheme. As the number of leaf nodes increase, this gap will increase.



(a) 'AND' policy (b) 'OR' policy

Fig.7 Decryption overhead

For the 'OR' policy structure, as seen in Fig.7(b), since there is no access control logical structure in the sub-clouds, only when the user's private key fragment reaches the cloud and contains the valid attributes, the cloud will calculate the private key of this attribute and return it to the proxy. As a result, the proxy often needs to wait for the calculation results of multiple sub-clouds, so the single-cloud CP-ABE has low decryption cost.

In conclusion, for the 'AND' policy structure, multi-cloud has better decryption performance, while for the 'OR' policy structure, single cloud maintains greater advantage. If the

access policy contains both structures, the overall performance will depend on the specific policy expression.

D. Access Control Overhead

As we can see from Fig.8, as the number of user attribute increases, the overhead of access control increases. This is because when making access control judgment, the left and right subtrees of the access control tree need to be traversed recursively. As the number of leaf nodes increases, the tree height becomes higher and the recursion depth deepens, hence the time consumption increases. In addition, we can also find that the time for access control decreases as the number of clouds increases because each sub-cloud take part in access control operations in parallel.

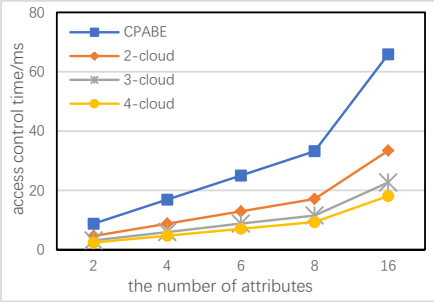


Fig.8 Access control overhead

VIII. Conclusions

A privacy-preserving multi-cloud CP-ABE scheme is proposed in this paper, which delegates multiple clouds to participate in hosting mutually disjoint sets of attributes. We improve the original CP-ABE algorithm and embed the global unique user identifier, so that only the initiator of access request could finally decrypt the ciphertext and effectively resist replay attack and man-in-the-middle attack. Meanwhile, we introduce a Proxy server to preserve the logical structure of the access control policy, which makes it impossible for the cloud to obtain the complete access policy from the ciphertext, thereby effectively protecting the privacy of the access policy. When decrypting the proxy will split the user's attribute private key, only a partial set of user attributes can be obtained by a sub-cloud, which effectively protects the privacy of the user's private key. Further, compared to the single cloud, the multi-cloud CP-ABE enables the decryption of access control policies with AND-gate structure to be performed in parallel, which greatly improves the efficiency of decryption and access control.

Acknowledgment

The authors would like to thank the anonymous referees of ISPA2018 for their reviews and suggestions to improve this paper. This work is supported by the National Key R&D Program of China(2016YFB0800402), partially supported by the National Natural Science Foundation of China under Grant No.61232004 and the Fundamental Research Funds for the Central Universities(HUST: 2016YXMS020).

References

[1] Zhang YQ, Wang XF, Liu XF, Liu L. Survey on cloud computing security. Journal of Software, 2016, 27(6):1328-1348.

[2] John Bethencourt, Amit Sahai, Brent Waters. Ciphertext-policy Attribute-based Encryption. In Proceedings of the IEEE Symposium on Security and Privacy, 2007:321-334

[3] Kan Yang, Xiaohua Jia, Kui Ren, et al. Enabling Efficient Access Control with Dynamic Policy Updating for Big Data in the Cloud. IEEE INFOCOM, 2014:2013-2021

[4] Kan Yang, Xiaohua Jia, Kui Ren. Secure and Verifiable Policy Update Outsourcing for Big Data Access Control in the Cloud. IEEE Transactions on Parallel & Distributed Systems, 2015, 12(26):3461-3470

[5] Kai Zhang, Junqing Gong, Shaohua Tang, et al. Practical and Efficient Attribute-Based Encryption with Constant-Size Ciphertexts in Outsourced Verifiable Computation. In Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, 2016:269-279

[6] Fuchun Guo, Yi Mu, Willy Susilo, et al. CP-ABE With Constant-Size Keys for Lightweight Devices. IEEE Transactions on Information Forensics and Security, 2014, 9(5):763-771

[7] Junbeom Hur, Dong Kun Noh. Attribute-Based Access Control with Efficient Revocation in Data Outsourcing Systems. IEEE Transactions on Parallel and Distributed Systems, 2011, 22(7):1214-1221

[8] Kan Yang, Xiaohua Jia. Expressive, Efficient, and Revocable Data Access Control for Multi-Authority Cloud Storage. IEEE Transactions on Parallel and Distributed Systems. 2014, 25(7):1735-1744

[9] Tran Viet Xuan Phuong, Guomin Yang, Willy Susilo. Hidden Ciphertext Policy Attribute-Based Encryption Under Standard Assumptions. IEEE Transactions on Information Forensics and Security, 2016, 11(1):35-45

[10] Jinguang Han, Willy Susilo, Yi Mu, et al. Improving Privacy and Security in Decentralized Ciphertext-Policy Attribute-Based Encryption. IEEE Transactions on Information Forensics and Security, 2015, 10(3):665-677

[11] Zhibin Zhou, Dijiang Huang, Zhijie Wang. Efficient Privacy-Preserving Ciphertext-Policy Attribute Based-Encryption and Broadcast Encryption. IEEE Transactions on Computers, 2015, 64(1):126-138

Jiguo Li, Haiping Wang, Yichen Zhang, et al. Ciphertext-Policy Attribute-Based Encryption with Hidden Access Policy and Testing. KSII Transactions on Internet and Information Systems, 2016,10(7):3339-3352

[12] Jianting Ning, Zhenfu Cao, Xiaolei Dong, el al. Auditable σ-Time Outsourced Attribute-Based Encryption for Access Control in Cloud Computing. IEEE Transactions on Information Forensics and Security, 2018,13(1):94-105

[13] Ruixuan Li, Chenglin Shen, Heng He, et al. A Lightweight Secure Data Sharing Scheme for Mobile Cloud Computing. IEEE Transactions on Cloud Computing, 2017:1-14

[14] Taeho Jung, Xiang-Yang Li, Zhiguo Wan, et al. Control Cloud Data Access Privilege and Anonymity With Fully Anonymous Attribute-Based Encryption. IEEE Transactions on Information Forensics and Security. 2015, 10(1):190-199

[15] Kan Yang, Xiaohua Jia. Attribute-Based Access Control for Multi-authority Systems in Cloud Storage. In 32nd IEEE International Conference on Distributed Computing Systems, 2012:536-545

[16] Kaiping Xue, Yingjie Xue, Jianan Hong, et al. RAAC: Robust and Auditable Access Control With Multiple Attribute Authorities for Public Cloud Storage. IEEE Transactions on Information Forensics and Security, 2017,12(4):953-967

[17] Riccardo Di Pietro, Marco Scarpa, Maurizio Giacobbe. Secure Storage as a Service in Multi-Cloud Environment. International Conference on Ad-Hoc Networks and Wireless, 2017:328-341