ATLANTIS PRESS International Journal of Computational Intelligence Systems, Vol. 11 (2018) 483–495

A new access control method based on multi-authority in cloud storage service

Sheng Luo[1,*], Qiang Liu[2]

[1]*School of Economics and Commerce, South China University of Technology, Guangzhou, 510006, China*
315412323@qq.com

[2] *school of computing, Southwestern University, Chongqing, 400715, China*
378180157@qq.com

Received 16 June 2017

Accepted 12 January 2018

Abstract

With the arrival of the era of big data, data has become a kind of important assets. In order to get a better utilization of big data, paid or unpaid data sharing will be a trend. And as one of key techniques to maintain security of data sharing, access control will play an important role in cloud storage services. This paper proposes an access control method for revocation of user rights in cloud storage services. Revoking user rights includes two aspects: revoking users and revoking attributes. The model presented in this paper is composed of attribute authority (AA), data owner (DO), user and cloud server. The key components of each part are generated by AA and DO, thus avoiding the joint attack between the user and AA. Then, the security of the scheme is analyzed by using Decisional Bilinear Diffie-Hellman (DBDH) theory. Experiments show that the scheme can effectively revoke user rights. Compared with other schemes, the proposed scheme has higher efficiency in terms of computation cost and communication cost. The research results have certain theoretical and practical significance.

Keywords：Revoke rights, Cloud storage server, Encryption algorithm, Access control

1. Introduction

With the arrival of the era of big data, data has become a kind of important assets. In order to get a better utilization of big data, paid or unpaid data sharing will be a trend. Many people have gradually realized the value of big data and have begun to apply big data analysis technology in the fields of public health, commerce and scientific research[1,2,3]. For example, in order to better predict the "Double 11 shopping Carnival" volume of transactions, many e-commerce companies (e.g., Taobao, Tmall and Jd) use big data analysis technology to study consumer spending over a period of years, mainly user browsing habits, loyalty, collection of goods and other analysis, to effectively carry out stocking. In addition, IBM has worked with Tongren Hospital to set up a large data analysis system for clinical, operational, and scientific research and assessment to better serve patients. We predict that with the application and promotion of big data analysis technology, data will become more and more important in the current era, eventually becoming a huge economic asset similar to minerals and oil[1].

With the explosive growth of information and the rapid development of cloud storage as a service technology, major e-commerce enterprises have launched their own cloud storage services[5]. Access control is one of the core security concerns of cloud storage service, and it is an indispensable means of ensuring data security.

* Corresponding author. E-mail: 315412323@qq.com

Traditional access control is based on the trusted server[18]. In order to adapt to the requirements of cloud storage services, it is necessary to introduce an encryption mechanism on the basis of traditional access control. Sahai and Waters[6] proposes a fuzzy identity-based encryption algorithm; the algorithm uses a set of attributes to describe user identity, which is the prototype of attribute-based encryption (ABE). Subsequently, the key policy based on attribute encryption (KP-ABE) was proposed by scholars[7]. In the KP-ABE framework, ciphertext is associated with a set of descriptive attributes, and the access control structure is contained in the user key. This allows the user to select the matched ciphertext, and the ciphertext cannot select a particular user, which is not suitable for the access control of cloud storage. Ciphertext policy based on attribute encryption (CP-ABE) overcomes this shortcoming of KP-ABE[8]. In the CP-ABE framework, the ciphertext contains an access control structure, and the user key is associated with a set of descriptive attributes, which is similar to the traditional access control mechanism. CP-ABE has flexibility as well as anti-conspiracy attack characteristics, which is considered the most suitable model for cloud storage access control. At present, the improved model based on CP-ABE has become the focus of research, and various models have been proposed based on CP-ABE. Waters[9] proposed a new method for implementing CP-ABE, which is efficient and expressive, and has provable security under specific assumptions. The Linear Secret-Sharing Schemes (LSSS) is introduced in the encryption algorithm. It reduces the time complexity of the encryption algorithm and the decryption algorithm and realizes encryption and decryption in polynomial time, but the model does not consider the problem of revocation of user privileges.

Zhang and Chen[10] proposed a cloud storage access control model based on CP-ABE. Because the model introduces a dual encryption scheme, the user can log in and then re generate the key, and the user does not have to stay online all the time. However, the model does not have attribute authentication authority, and attribute authentication is performed by the data owner. This is not conducive to large-scale users and frequent access control. Pervez et al.[11] proposes an autonomous access scheme that revokes user rights by reconstructing access structures. The innovation of this method is that user privileges can be revoked only if a new attribute is added to the original access tree. The method is ingenious and simplifies the revocation process of user rights. Yang et al.[12] improved policy for revoking user rights, leading to the key update algorithm and version number concept. The model can flexibly and effectively revoke user rights and reduce the burden of data owners, but it does not reduce the amount of calculation of the key update process of the authentication authority. Xu et al.[13] proposes a model for dynamically revoking keys and updating keys based on CP- ABE. The model can update the system key or relieve the user's access rights, but it cannot resist collusion attacks between users and cryptographic service providers (CSP). Hur et al.[14,15] proposed a new scheme based on CP-ABE; the scheme led to the Key encryption key (KEK) tree, where the system randomly generated keys and distributed them to each leaf node and internal node. The scheme can efficiently revoke users and attributes, but it is easy to disclose too much information to semi-trusted service providers.

In cloud storage access control, once there is revocation of user rights, the data owner encrypts the ciphertext again, regenerates the user's private key and distributes it to each user. Thus, revoking user rights requires extensive computation, but revoking user rights is an essential part of cloud storage access control systems. This paper proposes an access control method to revoke user rights based on cloud storage service. There is no central authority (CA) in this method, thus reducing the security risk introduced by CA. The private key is generated by the data owner and the authorization organization and can resist the collusion attacks of the multiple authorized organizations. Access data and private key generation are not required by global identifier (GID), which can protect the user's identity information and support anonymous communication of users. Moreover, the method can revoke the attributes of the user. When the user is cancelled, the access structure tree does not need to be modified. When revoking attributes, there is no need to update the ciphertext components associated with all attributes, which can be efficiently revoked.

2. Research model

The research model is composed of the following four parts, as shown in Figure 1.

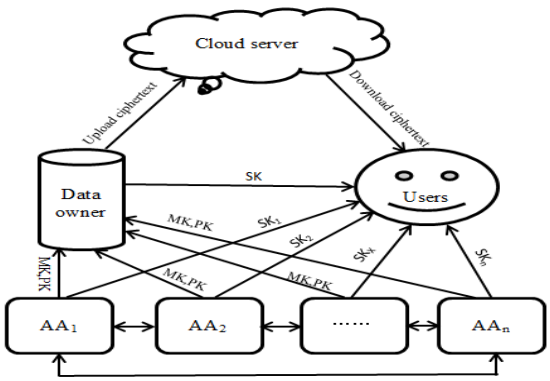

Fig.1. Research model

(1) Cloud server. The cloud server is a semi-trusted storage medium. It has very strong computing power and storage capacity. It does not perform other operations in addition to storing and re encrypting ciphertext files, but in order to gain benefits, it will acquire as much information as possible.
(2) Users. Any user can access ciphertext files on the cloud server, and only when the user's attributes satisfy the access policy defined by the data owner can the ciphertext be decrypted. The user's attributes are distributed by a plurality of authorized agents, and the user can request the key to the appropriate authority according to these attributes. To prevent joint attacks among users, the data owner also needs to embed the random number into the property private key.
(3) Data owner (DO). The DO provides valid data and expects only certain users to access the data, and other users (including cloud storage providers) are unable to access plaintext data. The DO is responsible for determining the set of legal users' attributes and making access policies.
(4) Attribute authorization (AA). AA is responsible for distributing the relevant attributes and attribute keys to users, and all AA can exchange some parameters with each other. If the user has only the property key and cannot decrypt the ciphertext, it must also have the private key component generated by the DO to decrypt it, which can effectively prevent a joint attack between servers.

3. Research design

3.1. Definition of access structure tree

In this paper, the access control adopts the tree structure(see, Figure 2), the access structure tree is defined by the DO, and the data are encrypted with the structure tree. The tree structure has the following characteristics:
(1) The access structure tree is an N fork tree, and the root node must be the AND node.
(2) Each subtree under the root node represents the access structure tree of the authorized agency (encrypt each message), and the child node of the root node must be the AND node.
(3) The leaf nodes represent the attributes assigned by each authorized agency, and the first byte of each attribute is used to mark the authorized agency that allocates the property.

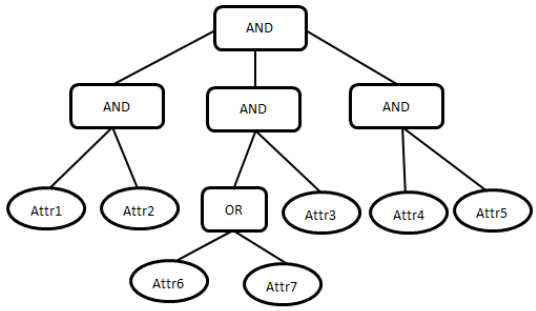

Fig.2. Access structure tree

Detailed definitions are as follows:

Let T be a tree representing the access control structure, and each non leaf node of the tree represents a threshold gate, which is composed of child nodes and a threshold.

Num_x represents the number of child nodes of node x. K_x represents the threshold among $0 \leq K_x \leq Num_x$. Each leaf node x of the tree represents the attribute, and $K_x = 1$. parent(x) represents the parent node of node x. The number of each child node is :1~Num. When threshold gates represent "OR", then $K_x = 1$. When threshold gates represent "AND", then $k_x = Num_x$. The function index(x) returns a number associated with the node x, and the index value of each node is unique. Select the polynomial q_x for each node x in the tree T. Starting from the root node R, the q_x is defined from top to

bottom. For each node, d_x is the number of polynomial q_x ; set the threshold of d_x minus 1, that is, $d_x = k_x - 1$.

Random selection of $s \in \mathbb{Z}_p$ makes $q_R(0) = s$. Then, d_R is used to define the polynomial q_R. For each node x, $q_x(0) = q_{parent(x)}(index(x))$. Then, d_x is used to define the polynomial q_x.

The relevant data are decrypted only when the user's attributes satisfy the definition of the access structure tree. The condition that satisfies the access structure tree is:

T is the access structure tree of the root node t. T_x is the subtree of node x in the access policy T. If the attribute set r satisfies the access structure tree T_x, $T_x(r) = 1$. We can define recursive operations as follows:

① If x is not a leaf node, the $T_{x'}(r)$ of all child nodes of the node x is calculated.

② If there is k_x child node, $T_{x'}(r)$ returns 1, and then $T_x(r) = 1$.

③ If x is a leaf node, then $T_x(r) = 1$.

3.2. Encryption scheme

The data security access control scheme in this article is as follows:

(1) Setup

Select a multiplication group G_0 of order p and expose it to the public. g is the generating element of G_0. Suppose there are N authorized bodies, $A_k (k \in (1,2,\cdots,N))$. Select random number $s_k \in \mathbb{Z}_p (k \in 0,1,\cdots,N)$. Then, calculate x_k:

$$\begin{cases} x_k = g^{S_k - S_{k-1}} (k \in 0,1,\cdots,N-1) \\ x_N = g^{S_N - S_0} (k = N) \end{cases} \quad (1)$$

DO sends x_k to authorized bodies A_k. All X_k are multiplied to obtain:

$$\prod_{k \in 0,1,\cdots,N} x_k = 1 \bmod p \quad (2)$$

The authority body selects the random number $v_k \in \mathbb{Z}_p$ and send $Y_k = e(g,g)^{v_k}$ to all other authority bodies. The calculation of any authority body is:

$$Y = \prod_{k \in A} Y_k = e(g,g)^{\sum_{k \in A} v_k} \quad (3)$$

For each authority body A_k, its key is: $MK_k = \{v_k, x_k\}$, Public key:

$$PK = \{G_0, g, Y = e(g,g)^{\sum v_k}\} \quad (4)$$.

(2) $KeyGen(PK, MK_k, A^u)$

The private key is generated by the DO and AA together.

① The DO randomly selects $\gamma \in \mathbb{Z}_p$; the resulting private key component is as follows:

$$A = (A_1 = x_0 g^{\gamma}, A_2 = g^{\frac{\alpha+\gamma}{\beta}})$$

and send the A to the user through the secure channel.

② For any attribute $i \in A_k^u$, A_k random selection $r_i \in \mathbb{Z}_p$, the private key component that generates the corresponding property is as follows: .

$$SK_k^u = (\forall i \in A_k^u, L_i = g^{t_i r_i}, D_i^{'} = g^{r_i})$$, among $k = 1,2,\cdots,n$.

AA are randomly selected $d_k \in \mathbb{Z}_p$; calculate $g^{v_k} \cdot g^{d_k}$, and secretly send it to all other AA. At the same time, send $x_k \cdot g^{d_k}$ to the user through the secure channel. Any AA can be calculated by the following formula, and send the results to the DO.

$$D = \prod g^{v_k} \cdot g^{d_k} = g^{\sum v_k + \sum d_k} \quad (5)$$

(3) $Encrypt(PK, M, T)$

In order to encrypt data files, a symmetric key k is selected randomly for symmetric encryption algorithms. Based on T, the DO encrypts the message M (that is, symmetric key k). DO random selection $s, \alpha, \beta \in \mathbb{Z}_p$ makes the root node of T be $q_r(0) = s$. Let Y be the leaf node in tree Γ; then, build the ciphertext as follows:

$$CT = \begin{cases} (T, \tilde{C} = Me(g,g)^{(\sum v_k + \alpha)s} \\ C = g^{\beta s} \\ \hat{C} = D^{\beta - 1} \\ \forall_y \in Y : C_{1,y} = g^{q_y(0)} \\ C_{2,y} = g^{t_y q_y(0)} \end{cases} \quad (6)$$

Then, the DO sends the ciphertext to the cloud server.

(4) $KeyAggregation(SK_k^u, A)$

After the user receives the property private key component sent by AA, the following is obtained by calculation:

$$D_i = L_i \cdot A_1 \cdot \prod x_k \cdot g^{dk} = g^{t_i r_i} \cdot g^{\sum d_k + r} \quad (7)$$

The private key component of the user u is as follows:

$$SK_u = \{\forall i \in A^u : D_{1,i} = g^{t_i r_i} \cdot g^{\sum d_k + r}, D_{2,i} = g^{r_i}\} \quad (8)$$

(5) $Decrypt(PK, SK_u, CT)$

Every user can download the ciphertext from the cloud server, but only the user's attributes that satisfy the access structure tree can decrypt. If x is a leaf node, order $i = att(x)$; if $i \in A^u$, then the formula is as follows:

$$DecryptNode(CT, SK_u, x) = \frac{e(D_{1,i}, C_{1,x})}{e(D_{2,i}, C_{2,x})}$$

$$= \frac{e(g^{t_i r_i} \cdot g^{\sum d_k + r}, g^{q_x(0)})}{e(g^{r_i}, g^{t_i q_x(0)})}$$

$$= e(g,g)^{(\sum d_k + r) q_x(0)}$$

Otherwise, define:

$$DecryptNode(CT, SK_u, x) = \perp$$

If x is not a leaf node, then for all child nodes z of x, call function $DecryptNode(CT, SK_u, x)$, and save the result as F_z. Make S_x express a collection of child nodes z; therefore, $F_z \neq \perp$. If such a collection does not exist, then the node does not satisfy the decryption condition, and the function returns $\perp$. Otherwise, order:

$$i = index(z), S_x' = \{index(z) : z \in S_x\},$$

calculated as follows:

$$F_x = \prod_{z \in s_x} F_z^{\vee_{i,s_x'(0)}}$$

$$= \prod_{z \in s_x} (e(g,g)^{(\sum d_k + r) q_z(0)})^{\vee_{i,s_x'(0)}}$$

$$= \prod_{z \in s_x} (e(g,g)^{(\sum d_k + r) q_{pareny(z)}(index(z))})^{\vee_{i,s_x'(0)}}$$

$$= \prod_{z \in s_x} (e(g,g)^{(\sum d_k + r) q_x(i))})^{\vee_{i,s_x'(0)}}$$

$$= e(g,g)^{(\sum d_k + r) q_x(0))}$$

After defining the function:

$$DecryptNode(CT, SK_u, x)$$

we can define the decryption algorithm. The algorithm starts with the root node of T and calls the function. If the user's property set satisfies T, then E is:

$$E = DecryptNode(CT, SK, R)$$

$$= e(g,g)^{(\sum d_k + r) R(0)}$$

$$= e(g,g)^{(\sum d_k + r) s}$$

Then, H is calculated as follows:

$$H = \frac{e(C, \hat{C} \cdot A_2)}{E}$$

$$= \frac{e(g^{\beta s}, g^{\frac{\sum v_k + \sum d_k}{\beta}} g^{\frac{\alpha+\gamma}{\beta}})}{e(g,g)^{(\sum d_k + r) s}}$$

$$= e(g,g)^{(\sum v_k + \alpha) s} \quad (9)$$

So, finally get:

$$M = \frac{\hat{C}}{H} \quad (10)$$

3.3. Revocation scheme

(1) Revoke user

In the cloud storage service of electronic commerce, if the user has purchased the service expired or the user exhibits malicious behavior, the DO can revoke the user's access to public resources. The cancelled user will no longer access the public resources. Whenever the user is revoked, the DO generates a new symmetric encryption key M', and encrypts the file with the new key. Then, the DO randomly selects $\alpha' \in \mathbb{Z}_p$, obtaining by calculation:

$$rk = g^{\frac{\alpha' - \alpha}{\beta}}, \quad \hat{C} = M' e(g,g)^{(\sum v_k + \alpha') s};$$

the DO sends the results to the Irrevocable user and the cloud server. The specific actions for revoking users

are as follows:

DO:

$$\alpha \to \alpha'$$

DO generates new key components.

$$\begin{cases} rk = g^{\frac{\alpha'-\alpha}{\beta}} \\ rk' = e(g,g)^{(\alpha'-\alpha)s} \end{cases} \tag{11}$$

DO sends rk and rk' to the legal user and the cloud server.

Cloud servers and users:

The cloud server encrypts the ciphertext using rk', and the user updates the key using the rk.

///Update ciphertext///

$$C' = rk' \cdot C = Me(g,g)^{(\sum v_k + \alpha')s}$$

///Update key///

$$A_2' = rk \cdot A_2 = g^{\frac{\alpha'+r}{\beta}}$$

Because revoking the user's scheme does not require redefining the access policy, there is no need to modify the T, so users can be efficiently revoked.

(2) Revoke attributes

Since the user's properties are revoked, a large number of attribute keys and public keys will be updated, so it is difficult to implement effective Undo properties. In this paper, we propose a scheme that can revoke user attributes. If an attribute of the user is revoked, then the user's remaining attributes still satisfy access to the structure tree, and the user can still decrypt the ciphertext. When the attribute j is revoked, the AA tells the DO that attribute j has been revoked. This is the same as revoking users. First, the DO generates the new symmetric key M' and uses it to encrypt data files again. Second, the DO randomly generates $s' \in \mathbb{Z}_p$, sets the root node of T to s', updates T, and computes $q_x'(0)$ for each leaf node. Last, the proxy re-encrypts the keys PRK_0, PRK_1 and PRK_2, and sends them to the cloud server. After the cloud server is processed, new ciphertext files C, $C_{1,y}$, and C_{2,y^n} are generated.

The specific actions for revoking attributes are as follows:

DO:

DO randomly selects $s' \in \mathbb{Z}_p$ and updates T.

$$s \to s', q_x(0) \to q_x'(0)$$

DO generates a proxy re-encryption key PRK and sends it to the cloud server.

$$PRK = \begin{cases} PRK_0 = e(g,g)^{(\sum v_k + \alpha)(s'-s)} \\ PRK_1 = s'/s \\ PRK_2 = \dfrac{q_i'(0)}{q_i(0)} \text{(except the revoked attributes)} \end{cases} \tag{12}$$

CSP (Cryptographic Service Provider):

The CSP receives the data sent by DO and recalculates and announces the new ciphertext.

$$CT' = \begin{cases} T', C' = PRK_0 \cdot C, C' = C^{PRK_1} \\ \forall y \in Y \setminus \{j\}, C_{1,y}' = C_{1,y}^{PRK_2} \\ C_{2,y}' = C_{2,y}^{PRK_2} \\ \forall j, C_{1,j}^2 = C_{1,j}, C_{2,j}' = C_{2,j} \end{cases} \tag{13}$$

The undo property scheme just modifies the value of the leaf node (revoked property node.) in T. In order to resist collusion attacks between users and cloud servers, the DO only sends the proxy re encryption key PRK to the CSP, and then the CSP is responsible for updating the encrypted component, thereby reducing the cost of revocation.

3.4. Reliability analysis

This paper will analyze the reliability of the scheme from the following four aspects: conspiracy attack, data secrecy, data separation, and security proof.

3.4.1. Conspiracy attack

This scheme is capable of resisting joint attacks by N authorized institutions, each of which is only responsible for generating part of the private key components. The private key component D_i is generated by the co computing of the data owner and the authorized institution. The data owner only sends its generated parameters to the legitimate user. Therefore, the authority institution can't get the D_i, and cannot further execute the decryption operation to obtain the

$e(g,g)^{(\sum d_k+r)s}$,and therefore cannot decrypt the ciphertext. Therefore, despite the fact that there are N authorized agencies collusion and cannot get all the keys, the proposed scheme can resist the collusion attack of N authorized agencies.

In addition, the proposed scheme can also resist the conspiracy attack between any number of users. In order to decrypt the ciphertext, the user must first get the $e(g,g)^{(\sum v_k+\alpha)s}$ .Before getting $e(g,g)^{(\sum v_k+\alpha)s}$, the user has to get $e(g,g)^{(\sum d_k+r)s}$. Because the data owner randomly generates a parameter r for each user . When two different key components are combined, there are different random numbers in each key. Therefore, the combination key cannot be computed by the polynomial calculation of the decryption algorithm to get the $e(g,g)^{(\sum d_k+r)s}$.

3.4.2. Data secrecy

Assuming that the symmetric algorithm used to encrypt data files is secure, the data secrecy depends only on the security of the key file. If the user's attribute set cannot meet the access structure tree embedded in the ciphertext, the user will not be able to obtain the decryption factor $e(g,g)^{(\sum d_k+r)s}$ during the decryption phase, and the user cannot access the key file. At the same time, if one of the attributes of the user is revoked, the attribute that the user has not been revoked satisfies the access structure tree, otherwise, the user cannot decrypt the ciphertext, thus ensuring the secrecy of the key file.

3.4.3. Data separation

In the proposed scheme, the cloud server reencrypts the ciphertext when the user is revoked. At the same time, in order to enable unrevoked users to obtain a new decryption factor $e(g,g)^{(\alpha'+\sum v_k)s}$, and update the key generated by the data owner. Data owners only send rk to legitimate users, the legitimate user can calculate a new decryption factor through rk . The revoked user cannot compute the new decryption factor with its previous key, so it cannot decrypt the encrypted ciphertext.

When a new user joins the system, the key generated by the data owner has been replaced by the new key α'. Therefore, although users retain the previous ciphertext before obtaining the attribute key, and its attributes satisfy the access control policy, it is still not possible to decrypt encrypted ciphertext encrypted by α .

3.4.4 Safety detection

(1)Scenario hypothesis

This paper tests the security of a model by playing a game between a challenger and an attacker. The game is divided into the following stages:

Step 1. Initialization stage. The attacker selects an access structure tree T that needs to be challenged and sends it to the challenger.

Step 2. Establishment stage. The Challenger runs the "Setup" algorithm and sends the generated public key to the attacker.

Step 3. Query stage 1. In order to obtain the private key component, the attacker submits its attribute set to the challenger, but these attributes do not satisfy the access structure tree T . Then, the Challenger runs the generation algorithm of the attribute key and sends the corresponding private key to the attacker.

Step 4. Challenge stage. The attacker submits two copies of the same size text (M_0 and M_1) to the challenger. The Challenger randomly tosses a coin, obtaining $b \in \{0,1\}$. The access structure tree T is used to encrypt the message M_b, and then the ciphertext is sent to the attacker.

Step 5. Query stage 2. Consistent with query stage 1.

Step 6. Guess stage. The attacker's guess regarding b is b'; when $b' = b$, the attacker wins the game.

In the above game, an attacker is called a chosen plaintext attack. The attacker's advantage is: $P_r[b' = b] - \frac{1}{2}$.

(2) Game process

Definition 1: This scheme uses selective plaintext security assumptions. That is, in any polynomial time, if the attacker's advantage in the security game is negligible, then the scheme is secure.

Definition 2: Decisional Bilinear Diffie-Hellman (DBDH) hypothesis.

Set a multiplication group (G , G_T) of order p ; g is the generating element of G . Select random number $a,b,c \in \phi_{bp}$. Then, the elements $g, g^a, g^b, g^c \in G$ and $Z \in G_T$ are sent to the attacker. The attacker determines whether Z is equal to $e(g,g)^{abc}$ .In polynomial time, if the attacker cannot solve the DBDH hypothesis with an ignorable advantage, the DBDH assumption is valid on the group (G , G_T).

Theorem: In group (G , G_T), if the DBDH assumption is established, the scheme is secure under the standard model.

Prove: Suppose there exists a probabilistic polynomial time; the attackers can win games by relying on the advantages ε of the non-negligible, and can prove that DBDH games can be resolved by the advantages $\varepsilon / 2$ of the non-negligible.

Order $e: G \times G \to G_T$ is a bilinear map. G is a multiplicative cyclic group that takes prime number P as an order, and g is its generating element. First, the Challenger randomly throws a coin to get a random value u . Then, there is random selection $a,b,c,z \in \phi_p$. If $u = 1$, the calculation is as follows:

Order:

$$(g, A, B, C, Z) = (g, g^a, g^b, g^c, e(g,g)^{abc})$$

Otherwise:

$$(g, A, B, C, Z) = (g, g^a, g^b, g^c, e(g,g)^z)$$

The Challenger sends

$$(g, A, B, C, Z) = (g, g^a, g^b, g^c, Z)$$

to the mimic, and the mimic plays the Challenger role in the DBDH game.

Step 1. Initialization stage.

The attacker creates an access structure tree T^* that he wants to challenge.

Step 2. Establishment stage.

The mimic randomly selects

$$d_n (n \in 1, 2, \cdots, N),$$

$$v_n (n \in 1, 2, \cdots, N),$$

$s, \alpha, r \in \phi_p$.

Then, $a = \sum d_k + r$,

$$b = \frac{\sum v_k}{\sum d_k + r},$$

$$c = \frac{(\sum v_k + \alpha)s}{\sum v_k}.$$

At the same time, the common parameters are generated and sent to the attacker,

$$Y = e(A, B) = e(g,g)^{ab}.$$

Step 3. Query stage 1.

The attacker repeatedly submits property $S_1, S_2, \cdots, S_q$ to query for the private key, and these attribute sets are managed by different AA, but the attribute sets does not satisfy T^*. After the mimic receives the private key query request, the corresponding key component is calculated. For attributes $i \in A^u$, the mimic randomly selects $r_i, t_i \in \mathbb{Z}_p$ and calculates $D_i = Ag^{t_i r_i}$, $D_i^{'} = g^{r_i}$; then, it sends the private key component to the attacker.

Step 4. Challenge stage.

The attacker submits M_0 and M_1 to the mimic.The mimic randomly tosses a coin to get the $b \in \{0,1\}$; then, the following ciphertext is generated:

$$CT^* = (T^*, C^{'} = M_b g^{Z}, C = g^{\beta s},$$
$$\hat{C} = D^{\beta-1}, C_i = g^{q_i(0)}, C^{'}_i = g^{t_i q_i(0)}) \quad .$$

① If $u = 1$,then

$$Z = e(g,g)^{abc} = e(g,g)^{(\sum v_k + \alpha)s}$$

this also means that the ciphertext CT^* is valid.

② If $u = 0$, then $Z = e(g,g)^z$, that is, $C^{'} = M_b e(g,g)^z$.

Since $z \in \mathbb{Z}_p$ is a random element, $C^{'} \in G_T$ is also a random element. Therefore, CT^* is invalid ciphertext.

Step 5. Query stage 2.

Consistent with query stage 1.

Step 6. Guess stage.

The attacker's guess in relation to b is $b^{'}$; if $b^{'} = b$, the mimic outputs $u^{'} = 1$ and points out that the tuples given to it are valid DBDH tuples $(g, A, B, C, e(g,g)^{abc})$. Otherwise, output $u = 0$, and points out that the tuple belongs to random tuples.

① When $u = 0$, the attacker receives no information about b, and there is no advantage to guess the correct $b^{'}$. Therefore, it is possible to get:

$$P_r[b \neq b^{'} \mid u = 0] = \frac{1}{2}$$

② When $b = b^{'}$, the challenger guess $u^{'} = 0$ can get:

$$P_r[u^{'} = u \mid u = 0] = \frac{1}{2}.$$

③ When $u = 1$, the attacker gets the valid ciphertext M_b. It has been defined in this case that the attacker's advantage is ε. Therefore, it is possible to get:

$$P_r[b = b^{'} \mid u = 1] = \frac{1}{2} + \varepsilon.$$

From the above, the probability of success between the mimic and the challenger is P_{Success}:

$$P_{\text{Success}} = \frac{1}{2} P_r[b = b^{'} \mid u = 1]$$

$$+ \frac{1}{2} P_r[b \neq b^{'} \mid u = 0] - \frac{1}{2}$$

$$= \frac{1}{2}(\frac{1}{2} + \varepsilon) + \frac{1}{2} \times \frac{1}{2} - \frac{1}{2}$$

$$= \frac{1}{2}\varepsilon$$

Thus, if there is probabilistic polynomial time, the attacker's advantage of winning the game is ε. Then, there is a non negligible advantage $\frac{1}{2}\varepsilon$ in polynomial time of solving the DBDH problem. That is, when ε is an advantage that cannot be ignored, the attackers can break the scheme presented in this article. It also means that the advantages $\frac{1}{2}\varepsilon$ of solving DBDH problems in polynomial time are not to be ignored. That is to say, the DBDH problem can be solved, which is contrary to the theory. Therefore, it can be concluded that in this security model, an attacker cannot break the scheme of this article.

4. Experiment and result

In order to test the effectiveness of the proposed scheme, we compare the proposed scheme with the reference[16] and [17] schemes. The two aspects computational cost and communication cost are compared. The configuration of the experimental environment is shown in Table 1.

Table 1. The configuration of the experimental environment

Name		Parameter	Remarks
Hardware	CPU	Intel(R) CoreTM i5-2400 @ 3.10 GHZ	8M cache, I/O Reading speed 2M.
	Memory	DDR3 1333/1600 MHz,4G	
	Hard disk	256 GB PCIe × 4 NVMe SSD	
	Network	Guangzhou Telecom	
Software	Operating system	Ubuntu 12.04	Nothing
	PBC Library	PBC-0.5.12	
	Encryption mode	OpenSSL-1.0.0c,128Bit,AES	

4.1. Computational cost

The computational cost includes two aspects: On the one hand, the relation between the encryption time and the number of AA is compared. On the other hand, the relation between the encryption time and the number of attributes of each AA is compared. The experimental results are shown in Figure 3, Figure 4, Figure 5 and Figure 6.

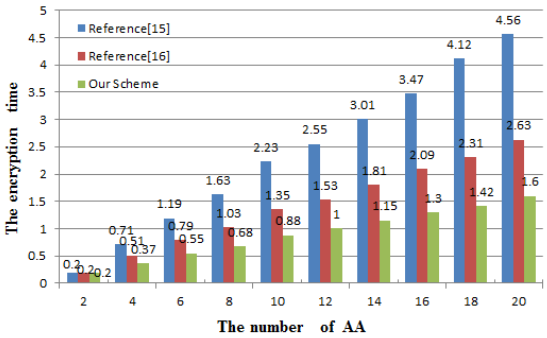

Fig.3. The relationship between the number of AA and the encryption time.

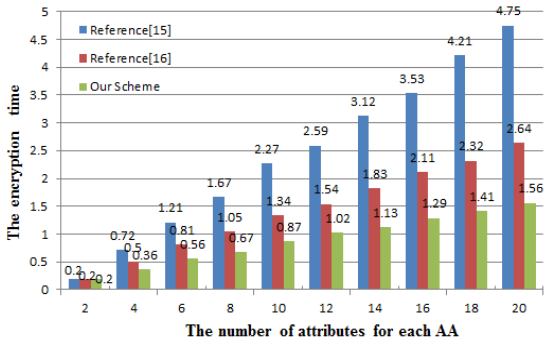

Fig.4.The relationship between the number of attributes for each AA and the encryption time

Figure 3 describes the relationship between the number of AA and the encryption time, and the number of attributes for each AA is set to 10. Figure 4 describes the relationship between the number of attributes for each AA and the encryption time, and the number of AA is set to 10. From Figure 3 and Figure 4, we can see that with an increase in the number of AA or the number of attributes of each AA, their attributes and corresponding attribute keys are continuously increased, resulting in a linear increase in data encryption time. However, in the encryption process, compared with the reference[16] and [17] algorithm, the algorithm used in this paper requires the least amount of computation time and exhibits the highest computational efficiency. In the encryption phase, the logarithmic and exponential operations of G_1 require a lot of time. Suppose the total number of attributes is l.

In the encryption phase, the cipher component generated by this scheme is less than the other two; this scheme only needs 2 times the logarithmic operation and $2l+3$ times the exponent operation. Reference[16] needs $1+2l$ times the logarithmic operation and $3l$ times the exponent operation. Reference [17] needs 1 times the logarithmic operation and $5l+3$ times the exponent operation. Since logarithmic computation requires more time than exponential operations, the computation cost of M is the highest with the increase in AA number. Thus, the computation cost of our scheme is the least.

Figure 5 describes the relationship between the number of AA and the decryption time, and the number of attributes for each AA is set to 10. Figure 6 describes the relationship between the number of attributes for each AA and the decryption time, and the number of AA is set to 10. From Figure 5 and Figure 6, we can see that if the number of AA is increased or the number of attributes of each AA is increased, their attributes and corresponding attribute keys are continuously increased, resulting in a linear increase in data decryption time. Similarly, in the decryption phase, our scheme requires less ciphertext components than the other two schemes, and our scheme only needs a $2l+1$ sub logarithmic operation. Reference[16] requires $4l$ times the logarithmic operations and l times the G_T elements multiple. Reference[17] requires $4l$ times the logarithmic operations. From the above, we can see that the computational cost of document M is still the highest, and the computational cost of this research scheme is the lowest.

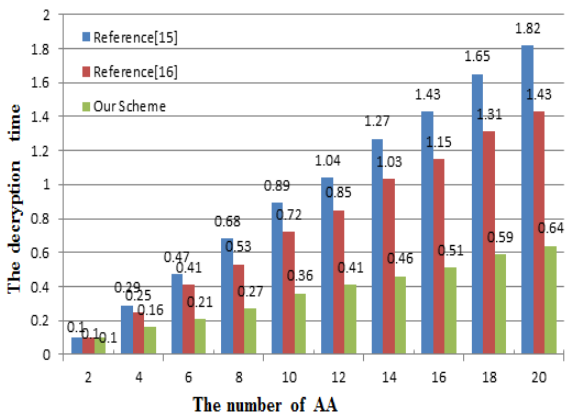

Fig.5.The relationship between the number of AA and the decryption time

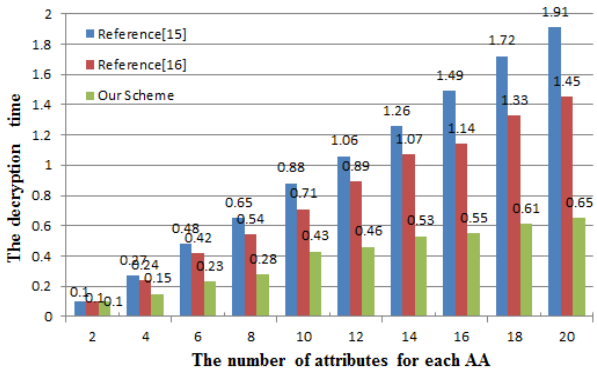

Fig.6. The relationship between the number of attributes for each AA and the decryption time

4.2. Communication cost

In Table 2, $|G|$ and $|G_T|$ represent the size of G and G_T elements, respectively. I_c represents all the number of attributes used for the encryption phase. I_k represents the attribute of the authorized organization AA_K. The number of AA is K. $u, n_{k,u}$ represents the number of attributes assigned by AA_K to the user.

In this system, the communication costs between AA and users mainly comes from the user's key components. Because the number of key components needed by our scheme is more than the other two schemes, the communication cost is greater than that of the other two schemes. Since the other two schemes require more common parameters than our scheme, in the communication between AA and DO, the communication cost of our scheme is greater than that of the other two schemes. The communication cost between the cloud server and DO is mainly caused by ciphertext. The ciphertext component of our scheme is less than that of the other two schemes, so the communication cost is small. Similarly, since the communication cost between the cloud server and the user is also derived from the ciphertext component, our scheme is less costly than the other two schemes. In our scheme, the DO is responsible for generating partial keys in order to resist the joint attack of AA. Therefore, there is some communication cost between the DO and users, but its cost is very small. The communication cost is shown in Table 2.

Table 2. Communication costs for various schemes

Scheme	Our scheme	Reference [16]	Reference[17]
AA and User	$n_k \mid G \mid +2\sum n_{k,u} \mid G \mid$	$\sum n_{k,u} \mid G \mid$	$\sum (n_{k,u} \mid G \mid +2 \mid G \mid)$
AA and Owner	$(k+2) \mid G \mid + \mid G_T \mid$	$k\mathbf{g}I_k(\mid G \mid + \mid G_T \mid)$	$2k(I_k \mid G \mid + \mid G \mid + \frac{1}{2} \mid G_T \mid)$
Server and User	$2 \mid G \mid (I_c+1)+ \mid G_T \mid$	$2I_c \mid G \mid + \mid G_T \mid (I_c+1)$	$2 \mid G \mid (2I_c+1)+ \mid G_T \mid$
Server and Owner	$2 \mid G \mid (I_c+1)+ \mid G_T \mid$	$2I_c \mid G \mid + \mid G_T \mid (I_c+1)$	$2 \mid G \mid (2I_c+1)+ \mid G_T \mid$
User and Owner	$2 \mid G \mid$	0	0

5. Conclusions and Future Work

This paper proposes an access control method that can revoke user rights in the e-commerce cloud storage service. Through constructing access structure trees to manage attributes distributed by different authorized organizations, users can realize secure and flexible cross-domain data storage and access control. When the user obtains the property private key, the GID is no longer required to be submitted to the AA, so that the user's identity information is protected.

The proposed scheme can implement fine-grained access control and can revoke the user's attributes and revoke users. The user's private key is generated by the DO and AA, which can resist any joint attack by any user and a joint attack by all authorized organizations. Theoretical analysis and experimental results show that the proposed scheme has obvious advantages over other schemes, and it can implement encryption and decryption computation in a highly efficient way. However, in the calculation process, each data owner needs to generate new parameters, which increases the computational cost of the system. In future research, under the premise of ensuring system security, we will continue to study the computational cost and expect to find a method that can effectively reduce the computational cost.

Acknowledgments

The authors would like to thank the anonymous referees and the editor for their valuable opinions.The subject is sponsored by the Social Science Planning Office of Guangzhou City, Guangdong Province, China (no.2015WHJD03) and the development research center of the people's Government of Guangdong Province, China (no. 201402).

References

1. J. Bethencourt, A. Sahai and B. Waters, Ciphertext-policy attribute-based encryption, IEEE Symposium on Security and Privacy, 2007, p.^pp. 321-334.
2. G. Li, Research status and scientific thinking of big data, Bulletin of Chinese Academy of Sciences (2012).
3. M. J. Bietz, A. Wiggins, M. Handel and C. Aragon, Data-intensive collaboration in science and engineering, 2012, p.^pp. 3-4.
4. P. Elavarasi., Key updation for the dynamic attributes in cloud computing for competent user retraction, International Journal of Engineering Science & Technology 5 (2013), no. 6S.
5. J. U. Jiehui, W. U. Jiyi, F. U. Jianqing and Z. Lin, A survey on cloud storage, Journal of Computers 6 (2011), no. 8, 1764-1771.
6. A. Sahai and B. Waters, Fuzzy identity-based encryption, International Conference on Theory and Applications of Cryptographic Techniques, 2005, p.^pp. 457-473.
7. V. Goyal, O. Pandey, A. Sahai and B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, ACM Conference on Computer and Communications Security, 2006, p.^pp. 89-98.
8. V. Goyal, A. Jain, O. Pandey and A. Sahai, Bounded ciphertext policy attribute based encryption, DBLP, 2008.
9. B. Waters, Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization, Lecture Notes in Computer Science 2008 (2011), 321--334.
10. R. Zhang and P. S. Chen, A dynamic cryptographic access control scheme in cloud storage services, Journal of Information Processing & Management 4 (2012), no. 1, 50-55.
11. Z. Pervez, A. M. Khattak, S. Lee and Y. K. Lee, Sapds: Self-healing attribute-based privacy aware data sharing in cloud, Journal of Supercomputing 62 (2012), no. 1, 431-460.
12. K. Yang, X. Jia and K. Ren, Attribute-based fine-grained access control with efficient revocation in cloud storage systems, ACM Sigsac Symposium on Information, Computer and Communications Security, 2013, p.^pp. 523-528.

13. Z. Xu and K. M. Martin, Dynamic user revocation and key refreshing for attribute-based encryption in cloud storage, IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 2012, p.^pp. 844-849.
14. J. Hur and K. N. Dong, Attribute-based access control with efficient revocation in data outsourcing systems, IEEE Transactions on Parallel & Distributed Systems 22 (2010), no. 7, 1214-1221.
15. J. Luo, H. Wang, X. Gong and T. Li, A novel role-based access control model in cloud environments, International Journal of Computational Intelligence Systems 9 (2016), no. 1, 1-9.
16. S. Ruj, A. Nayak and I. Stojmenovic, Dacc: Distributed access control in clouds, IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 2012, p.^pp. 91-98.
17. K. Yang and X. Jia, Expressive, efficient, and revocable data access control for multi-authority cloud storage, IEEE Transactions on Parallel & Distributed Systems 25 (2014), no. 7, 1735-1744.
18. L. X. Xie, F. K. Bo, and B. B. Zhao, virtual group revocation policy-based cloud storage access control model. Computer Science, 43(2016),122-126.