An efficient access control scheme based on CP-ABE with supporting attribute change in cloud storage systems

Tao Ye

Faculty of Information Technology,
Beijing University of Technology,
Beijing, China
and
College of Computer,
Qinghai Nationalities University,
Xining, China
Email: fly_940218@163.com

Yongquan Cai

Faculty of Information Technology,
Beijing University of Technology,
Beijing Key Laboratory of Trusted Computing,
National Engineering Laboratory for
Critical Technologies of Information,
Security Classified Protection,
Beijing 100124, China
Email: cyq940218@163.com

Xu Zhao*

College of Applied Sciences,
Beijing University of Technology,
Beijing 100124, China
Email: yyyyll_jiaren@163.com
*Corresponding author

Yongli Yang, Wei Wang and Yi Zhu

Faculty of Information Technology,
Beijing University of Technology,
Beijing Key Laboratory of Trusted Computing,
National Engineering Laboratory for
Critical Technologies of Information,
Security Classified Protection,
Beijing 100124, China
Email: yyyyll1218@163.com
Email: yyl_1218@126.com
Email: flyfly_0218@163.com

Abstract: The CP-ABE-based access control scheme, which can better realise the access control of many-to-multi-ciphertext shared in the cloud storage architecture, is still facing the problems that the system cost is too large, and the policy attribute revocation or restore is not flexible. This paper proposes an efficient access control scheme based on CP-ABE with supporting attribute change in cloud storage system. The fine-grained access control can be achieved by re-encryption mechanism which takes the minimum shared re-encryption key for policy attribute set. And then the access structure tree is expanded by creating a corresponding virtual attribute for each leaf node attribute. The analysis results of the scheme indicate that the efficient and flexibility of the attribute change is not only improved, but also the system cost is reduced.

Keywords: access control; policies attribute change; cloud storage; ciphertext-policy ABE.

Reference to this paper should be made as follows: Ye, T., Cai, Y., Zhao, X., Yang, Y., Wang, W. and Zhu, Y. (2019) 'An efficient access control scheme based on CP-ABE with supporting attribute change in cloud storage systems', *Int. J. Wireless and Mobile Computing*, Vol. 16, No. 1, pp.41–49.

Biographical notes: Tao Ye is currently PhD Candidate of the Faculty of Information Technology, Beijing University of Technology, China. His main research interests include information security and computer networks.

Yongquan Cai is currently a Professor at the Faculty of Information Technology, Beijing University of Technology, China. His main research interests include information security, computer network and cryptographic protocols analysis.

Xu Zhao is a Lecturer at the College of Applied Sciences, Beijing University of Technology, China. Her main research interests are statistical analysis and statistics inference.

Yongli Yang is a PhD Student at the Faculty of Information Technology, Beijing University of Technology, China. Her research interests are recommendation system and swarm intelligence.

Wei Wang is a Lecturer at the Faculty of Information Technology, Beijing University of Technology, China. Her research interests are information security and so on.

Yi Zhu is a Master at the Faculty of Information Technology, Beijing University of Technology, China. Her research interests are information security and so on.

1 Introduction

With the rapid development and extensive application of cloud storage service in open network environment for a large number of cross-domain users, the approaches of traditional access control such as mentioned in the literature (Zhou et al., 2015; Ueda and Ruggiero, 2012; Zhu et al., 2015), where a trusted server is in charge of defining and enforcing access control policies, is still facing new problems. Especially, when an access user no longer is allowed to access some data in the cloud storage system, the data owner needs to revoke the user's access authority to the data in a timely manner. Although the user whose access authority has been revoked has reserved the previously assigned key, it can't use the key to access the data in the cloud. Thus, the cloud storage service systems require flexible, fine-gained and efficient access control approaches.

The mechanism of the ciphertext-policy ABE (CP-ABE) (Waters, 2011; Li et al., 2016; Ma et al., 2015) is a promising approach that fulfils these requirements in the cloud storage architecture. Because it enables data owners to choose an access structure on attributes, and to encrypt data to be outsourced under the access structure via encrypting with the corresponding public attributes. Different users are allowed to decrypt different pieces of data per the security policy in the cloud storage system. Most of CP-ABE schemes are based on the Decisional Bilinear Diffie-Hellman (DBDH) problem (Veugen, 2015) and Key Management Mode (Tan et al., 2016)

At present, most of the research is focused on two aspects. First, this is how to reduce the encrypting data cost of the data owner with to support variable policy access control, i.e., the problem of revoking attribute rights; the second is how to achieve fine-grained, flexible access control by using the powerful computing and storage capabilities of the cloud computing environment, including flexible changes that support user attributes. Thus, in this study, we will attempt to solve these problems in attribute-based data access control by using CP-ABE for cloud storage system.

2 Related work

Bethencourt et al.(2007) proposed the CP-ABE scheme for the first time, and realised the numerical comparison method by using the "AND" and "OR" doors over the inspiration of attribute revocation mechanism on the IBE numerical system to carry out the system attribute revocation. Pirretti et al. (2010) proposed that the user private key is regularly updated by a trusted third party through a maintaining list of revoked attributes. Liang et al. (2009) combines the access policy with the user's private key, to re-encrypt the ciphertext under one access policy into a ciphertext under another different access policy based on the proxy re-encryption technique, thus realising the flexibility of the access policy. Hur and Noh (2011) proposed an attribute-based access control with efficient revocation (ABAC-ER) that supports a fine-grained attribute revocation mechanisms, But the data owner needs to re-encrypt all the data when the policy changes. Its cost is too large.

In this paper, an efficient access control scheme based on CP-ABE is proposed to support the change of virtual policy attributes over the existing fine-grained access control scheme (Hurand Noh, 2011; Naor et al., 2001). The scheme can not only reduce the cost of data owner, but also realise the flexible of access control.

3 Preliminaries and definition

3.1 Bilinear pairings

Let $\mathbf{G}_1$, $\mathbf{G}_2$ be two cyclic groups of prime order q. A bilinear map e is a map $e:\mathbf{G}_1\times\mathbf{G}_1\to\mathbf{G}_2$ with the following properties.

1 *Bilinearity*: For all $g_1,g_2\in\mathbf{G}_1$, and $a,b\in\mathbb{Z}_q^*, e\left(g_1^a,g_2^b\right)=e\left(g_1,g_2\right)^{ab}$

2 *Non-degeneracy*: $e\left(g_1,g_2\right)\neq 1$. The mapping e does not map all the elements of $\mathbf{G}_1\times\mathbf{G}_1$ to the unit of $\mathbf{G}_2$. If g is the generator of $\mathbf{G}_1$, then $e(g, g)$ is the generator of $\mathbf{G}_2$.

3 *Computability*: There is an efficient algorithm to compute $e(g_1, g_2)$ for any $g_1,g_2\in\mathbf{G}_1$. Then, e is a bilinear mapping, $\mathbf{G}_1$ is used as the additive group, and $\mathbf{G}_2$ is taken as the multiplicative group.

3.2 Bilinear Diffie-Hellman (BDH) assumption

The Bilinear Diffie-Hellman problem is to compute $r=\left(g,g\right)^{abc}\in\mathbf{G}_2$ given a generator g of cycle group $\mathbf{G}_1$ and elements g^a,g^b,g^c for $a,b,c\in\mathbb{Z}_q^*$.

To determine the bilinear Diffie-Hellman problem is to determine whether $r = e(g, g)^{abc}$ is true or not while given a five-tuple $\left(g,g^a,g^b,g^c,r\right)$. Where g be a generator of the cyclic group $\mathbf{G}_1, r\in\mathbf{G}_2$, $a,b,c\in\mathbb{Z}_q^*$ and unknown.

3.3 Shamir threshold secret sharing mechanism

Shamir proposed a classical threshold secret sharing scheme, which was designed by using the Lagrangian polynomial interpolation method (Bethencourt et al., 2007).

Let q be a prime number and $k\in\mathbb{Z}_q^*$ be the shared key. Assuming that the subkey is assigned to the $n(n<q)$ participants $q_i\left(1\le i\le n\right)$ by a trusted third party, the procedure of Shamir key allocation scheme is as follows.

1 Randomly generate a $s-1$ polynomial $g(x)=a_{s-1}x^{s-1}+a_{s-1}x^{s-2}+L+a_1x+a_0\in?_q^*$, the coefficient $a_0=k$, and then there is $g(0) = k$.

2 To arbitrarily select n nonzero and different elements x_1,x_2,L,x_n in the finite field $\mathbb{Z}_q^*$, and calculate $y_i=g\left(x_i\right),1\le i\le n$.

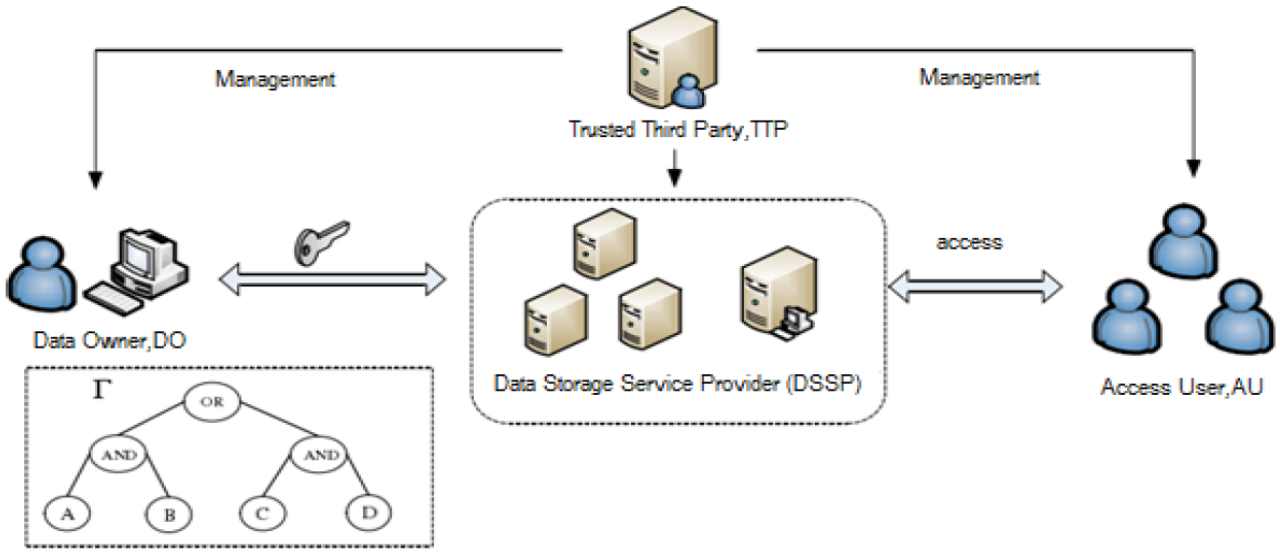
3 Assign $\left(x_i,y_i\right),\left(1\le i\le n\right)$ to the participants q_i, where x_i is public and y_i is the subkey for each q_i .Given the S subkey $y_r\left(1\le r\le s\right)$, the reconstructed polynomial by using the Lagrangian interpolation method is g (x) as follow.

$$g(x)=\sum_{r=1}^{s}y_r\prod_{j-1,j\neq r}^{s}\frac{x-x_j}{x_r-x_j}$$

3.4 Cloud data storage architecture

The scheme consists of a Data Storage Service Provider (DSSP), a Data Owner (DO), a Trusted Third Party (TTP), and the ordinary Access Users (AU), as shown in Figure 1.

Figure 1 Cloud data storage architecture



1 *Data owner* (*DO*): It is a client who owns the original plaintext data. DO is responsible to define attribute-based access policy, and to encrypts it and the original plaintext data, finally stores the ciphertext to the storage server provided by the DSSP.

2 *Data storage service provider* (*DSSP*): It is an entity that provides data storage services for users. It is assumed that the DSSP can't be completely trusted, that is, he will perform the algorithm tasks honestly, but would also try to obtain confidential information including access strategy and plaintext.

3 *Trusted third party* (*TTP*): It is mainly used to generate the initial public parameters and MK of the system. TTP distributes public parameters and PK of the system for each data owner, and distributes, revokes and updates its private key SK for each AU. The AU will send a request to the TTP. When the AU or AU's attribute changes, TTP will notify DSSP. In this scheme, TTP is assumed that it is a fully trusted participant in the data storage system model.

4 *Access user* (*AU*): It is an entity who can read the ciphertext on the data storage server. A user is able to decrypt the plaintext if and only if he has a set of attributes to satisfy the ciphertext access policy.

3.5 Definitions

Definition 1: Let $u=\{u_1,u_2,L,u_n\}$ be the universe of users. n is the total number of users in the system. Let $A=\{\lambda_1,\lambda_2,\mathrm{L},\lambda_q\}$ be the universe of descriptive attributes in the system. Among them, q is the order of group **G**. Let $G=\{G_1,G_2,L\ G_q\}$ be the universe of such attribute groups. Let $G_i\subset u$ be a set of users that hold the attribute λ_i, which is referred to as an attribute group. Let G_i be used as an access user list to λ_i. Let K_{λ_i} be the attribute group key that is shared among the non-revoked users in $G_i\in G$.

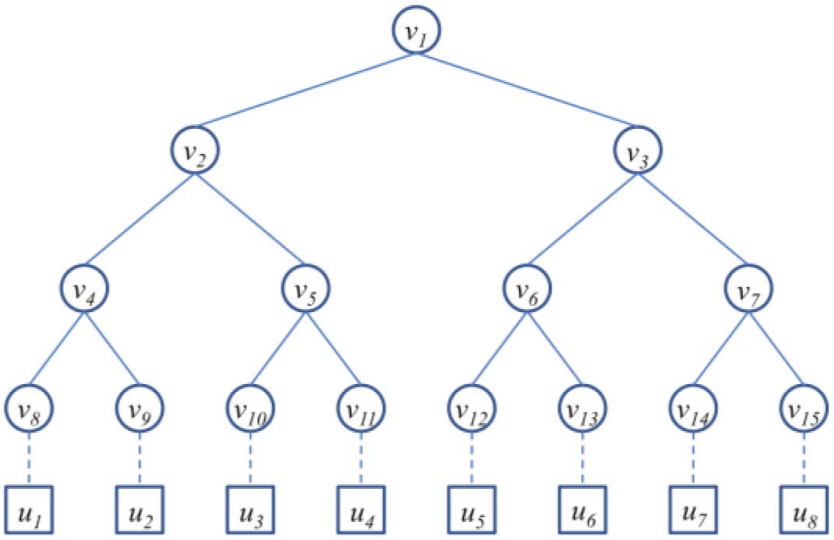
Definition 2 (*Access structure*): Let $P=\{P_1,P_2,L,P_n\}$ be a set of parties. A collection $\Gamma\subseteq 2^{\{P_1,P_2,L,P_n\}}$ is monotone if $\forall B,C$: if $B\in\Gamma$ and $B\subseteq C$ then $C\in\Gamma$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) Γ of nonempty subsets of $P=\{P_1,P_2,L,P_n\}$, i.e., $\Gamma\subseteq 2^{\{P_1,P_2,L,P_n\}}\setminus\{\phi\}$. Γ is an access structure on the participant P. The sets in Γ are called the authorised sets, and the sets not in Γ are called the unauthorised sets. The role of the parties is taken by the attributes. Thus, the access structure AA will contain the authorised sets of attributes.

Definition 3 (*KEK and KEK tree*): The Key Encrypting Key (KEK) is the distribution key for the re-encryption key. It is used to encrypt the different group attribute keys in the re-encryption phase, which is distributed to the user through the KEK tree as in Figure 2. The KEK tree is a binary tree managed and saved by the DSSP. The KEK tree is constructed by the DSSP as follows:

1 Every member in u is assigned to the leaf nodes of the tree. Random keys are generated and assigned to each leaf node and internal node.

2 Each member $u_t\in AU$ receives the path keys PK_t from its leaf node to the root node of the tree securely. For instance, u_2 stores $PK_2=\{PK_9,PK_4,PK_2,PK_1\}$ as its path keys as shown in Figure 2.

Figure 2 KEK tree for attribute group key distribution (Naor et al., 2001)



Then, the path keys will be used as KEKs to encrypt the attribute group keys by the data service manager in the data re-encryption phase. The key assignment in this method is information theoretic, which is keys are assigned randomly and independently from each other.

Definition 4 (*User Path Key, UPK*): There is a KEK set from each leaf node *j*, to the root in the KEK tree, it is denoted by $\{\exists v_j \in V: KEK_j\}$, and *V* is all nodes in the tree. The collection $\{\exists v_j \in V: KEK_j\}$, is called the user path key (*UPK*)

3.6 Attribute-dased access control with efficient revocation (ABAC-ER)

In this paper, we adopt the same definition with ABAC-ER scheme (Naor et al. 2001), so the related definitions and algorithm steps of ABAC-ER scheme are given here. A Scheme of Attribute-Based Access Control with Efficient Revocation consists of six probabilistic polynomial-time algorithms.

1 *setup*():It is a system initialisation algorithm that takes no input other than the implicit security parameter. It outputs the public key *PK* and a master key *MK*.

2 *AttrKeyGen* (*MK, Λ, AU*):. It is an attribute key generation algorithm that takes as input the master key *MK,* a set of attributes $\Lambda \subseteq A$, and a set of user indices $AU \subseteq u$. It outputs a set of private attribute keys *SK* for each user in *AU* that identifies with the attributes set.

3 *KEKGen* (*AU*): It is the *KEK* generation algorithm that takes as input a set of user indices *AU*, and outputs KEKs for each user in *AU*, which will be used to encrypt attribute group keys K_{λ_i} for each $G_i \in G$.

4 *Encrypt (PK, M, Γ').* The encryption algorithm is a randomised algorithm that takes as input the public parameter *PK*, a message M, and an access structure Γ over the universe of attributes. It outputs a ciphertext *CT* such that only a user who possesses a set of attributes that satisfies the access structure will be able to decrypt the message.

5 *ReEncrypt* (*CT, G*)*:* The re-encryption algorithm is a randomised algorithm that takes as input the ciphretext *CT* including an access structure Γ, and a set of attribute groups *G*. If the attribute groups appear in Γ, it re-encrypts *CT* for the attributes; else, returns ⊥. Specifically, it outputs a re-encrypted ciphertext *CT*' such that only a user who possesses a set of attributes that satisfies the access structure and has a valid membership for each of them at the same time will be able to decrypt the message.

6 *Decrypt* (*CT',SK,KΛ*)*:* The decryption algorithm takes as input the ciphertext *CT*' which contains an access structure Γ, a private key *SK*, and a set of attribute group keys K_Λ for a set of attributes Λ. The decryption can be done if Λ satisfies Γ and K_Λ is not revoked for any $\lambda \in \Lambda$.

4 Our schemes construction

Attribute-Based Access Control with Node Virtual Extension (ABAC-NVE) is proposed based on ABAC-ER in the paper (Naor et al. 2001). A leaf node to represent the attribute in the access structure tree adds positive, negative attribute nodes, thus creating an extended access tree. The data owner DO generate encrypted ciphertext based on the extended access tree in the encryption phase, and uploaded the ciphertext to the DSSP, where a full strategy access control mechanism can be established based on all the attributes of existing strategy tree.

When the DO needs to revoke or restore the policy attributes of a policy tree, it can be quickly completed by DSSP without the participation of DO. This will make full the resource advantages of the cloud computing environment to realise flexible access policy changes. Of course, because the scheme extends the access policy tree, the computational cost of the encryption phase will increases. Such designs will undoubtedly lead to significant advantages for systems where policy attribute revocation and recovery are frequent.

Let a security parameters κ determine the size of the groups. We will also make Lagrange coefficients $\Delta_i; \Lambda$ for any $i \in \mathbb{Z}_q^*$ and a set Γ of elements in $\mathbb{Z}_q^*$ define $\Delta_i \Lambda(x) = \prod_{j \in \Lambda, j \neq i} \frac{x-j}{i-j}$. We additionally employ a hash function $H: \{0,1\}^* \to \mathbf{G}_1$ to associate each attribute with a random group element in $\mathbf{G}_1$, which we will model as a random oracle.

4.1 System algorithm

1 *Setup* (1^k): The algorithm is run by the TTP. It is a randomised algorithm that takes as no input other than the implicit security parameter. It outputs the public key *PK* and a master key *MK*.

$$PK = \{g, h = g^\beta, e(g,g)^\alpha\},$$
$$MK = \{\beta, g^\alpha\}$$

where α, β is random selection and $\alpha, \beta \in \mathrm{Z}_q^*, \mathbf{G}_1$ is an additive group of order *q*, and *g* is a generator of group $\mathbf{G}_1$. $\mathbf{G}_2$ is a multiplicative group of order *q*, and *e* is a bilinear mapping of $\mathbf{G}_1 \times \mathbf{G}_1 \to \mathbf{G}_2$.

2 *AttrKeyGen* (*MK, Λ, AU*): The algorithm is run by the TTP. It takes as input the master key MK, a set of attributes $AU \subset u.$ and a set of user indices $AU \subset u.$ It outputs a set of private attribute keys *SK* for each user $u_t \subset AU$ that identifies with the attributes set.

$$SK_{u_t} = \begin{pmatrix} D = g^{(\alpha+r)/\beta}, \\ \forall \lambda_i \in \Lambda: D_i = g^r \cdot H(\lambda_i)^{r_i}, \\ D_i' = g^{r_i} \end{pmatrix}$$

where $\gamma \in Z_q^*$ is randomly selected for each user, $y_i \in Z_q^*$ is randomly selected for each attribute $\lambda_i \in \Lambda$.

3 *KEKGen* (*AU*): The KEK generation algorithm is runs by the DSSP. It takes as input a set of user indices *AU*, and outputs KEKs for each user $u_t \subset AU$, which will be used to encrypt attribute group keys K_{λ_i} for each $G_i \in G$. It will Generate a header message

$$Hdr = \left(\forall y \in \left\{Y : E_{KKK}\left(K_{\lambda_y}\right)\right\}\right).$$

4 *Encrypt* (*PK*, *M*, *Γ′*): The encryption algorithm is executed by the DO. It is a randomised algorithm that takes as input the public parameter *PK*, a message M, and an access structure Γ′ that is defined by DO over the universe of attributes *A*. It outputs a ciphertext *CT* such that only a user who possesses a set of attributes that satisfies the access structure will be able to decrypt the message.

$$CT' = \begin{pmatrix} \Gamma', C' = M \cdot e(g,g)^{\alpha s}, C = g^{\beta s}, \\ \forall y \in Y : C_{y^+} = g^{q_{y^+}(0)}, \\ C'_{y^+} = \left(H\left(\lambda_{y+}\right)^{q_{y^+}(0)}\right)^{k_{\lambda_y}}, C_{y^-} = g^{q_{y^-}(0)}, \\ C'_{y^-} = \left(H\left(\lambda_{y-}\right)^{q_{y^-}(0)}\right)^{k_{\lambda_y}} \end{pmatrix}$$

where Γ′ is extended through adding the positive and negative attribute on the leaf nodes. The original leaf node becomes internal node 'AND'. M is the plaintext to be encrypted, $\alpha, \beta, s \in Z_q^*$, y^+ is the positive attribute of a leaf node, and y^- is the negative attribute of a leaf node. $H\left(\lambda_{y^+}\right)$ and $H\left(\lambda_{y^-}\right)$ are the public mapping function defined in the traditional algorithm CP-ABE, λ_{y^+} and λ_{y^-} are a random element mapped on group $\mathbf{G}_2$ respectively.

5 *ReEncrypt* (*CT, G*): The algorithm is executed by DSSP. It is a randomised algorithm that takes as input the ciphretext *CT* including an access structure Γ′, and a set of common attribute groups *G*. If the attribute groups appear in Γ′, It select a re-encryption key $k_{\lambda_y} \in Z_q^*$, to complete the ciphertext *CT* re-encryption for attribute group $G_y \in G$; returns ⊥ else,. Specifically, it outputs a re-encrypted ciphertext *CT′* such that only a user who possesses a set of attributes that satisfies the access structure and has a valid membership for each of them at the same time will be able to decrypt the message.

$$CT' = \begin{pmatrix} \Gamma', C' = M \cdot e(g,g)^{\alpha s}, C = g^{\beta s}, \\ \forall y \in Y : C_{y^+} = g^{q_{y^+}(0)}, \\ C'_{y^+} = \left(H\left(\lambda_{y+}\right)^{q_{y^+}(0)}\right)^{k_{\lambda_y}}, C_{y^-} = g^{q_{y^-}(0)}, \\ C'_{y^-} = \left(H\left(\lambda_{y-}\right)^{q_{y^-}(0)}\right)^{k_{\lambda_y}} \end{pmatrix}$$

6 *Decrypt* (*CT′, SK′, K*$_{\Lambda}$): The algorithm is executed by the user u_t who needs to access the encrypted file. It takes as input the ciphertext *CT′* which contains an access structure Γ′, a private key *SK*, and a set of *attribute* group keys K_{Λ} for a set of attributes Λ. The user u_t first obtains *Hdr* from the DSSP and the re-encryption key K_{λ_y}, and updates *SK* as *SK′*.

$$SK'_u = \left(D = g^{(\alpha+r)/\beta}, \forall \lambda_i \in \Lambda : D = g^r \cdot H\left(\lambda_i\right)^{r_i}, D' = \left(g^{r_i}\right)^{1/\mathrm{k}_{\lambda_i}}\right)$$

Define the recursive algorithm *DecryptNode* (CT', SK', x), where *x* represents all leaf nodes.

$$Decrypt\ Node(CT', SK', x) = \begin{cases} e\left(D_i, C_{y+}\right) / e\left(D'_i, C'_{y^+}\right) = e(g,g)^{rq_{y^+}(0)}, \\ e\left(D_i, C_{y-}\right) / e\left(D'_i, C'_{y^-}\right) = e(g,g)^{rq_{y^-}(0)}, \\ \perp \end{cases}$$

We will get each virtual leaf node $e(g,g)^{rq_{y\pm}(0)}$ using threshold-based shared secret algorithm. We ultimately can still get $e(g,g)^{rs}$, through $M = C^{\%} / e\left(C, D\right) / \left(e\left(g,g\right)\right)$ to restore the plaintext.

4.2 Changes of access policy attribute

DO needn't re-encrypt the M after the access policy attribute are revoked or recovered to introduce the extended access tree Γ′, this is completed by DSSP with the calculation of resource-rich.

When some system attributes are to be revoked, DO encrypts the extracted attribute set Γ′ with the shared key KEK and sends it to the DSSP. Then the DSSP executes the algorithm Re *MoveEncrypt* $\left(CT', T\right)$ for $\forall y \in T$. DSSP re-encrypts the ciphertext with randomly selected $S_{\lambda_y} = Z_p^*$ for each revoked system attribute λ_y, which is constructed as follows:

$$CT' = \begin{pmatrix} \Gamma', C' = M \cdot e(g,g)^{\alpha s}, C = g^{\beta s}, \\ \forall y \in Y : C_{y^+} = g^{q_{y^+}(0)}, \\ C' = \left(H\left(\lambda_{y+}\right)^{q_{y^+}(0)}\right)^{k_\lambda}, C_{y^-} = g^{q_{y^-}(0)}, \\ \forall y \in T : C'_{y^-} = \left(H\left(\lambda_{y-}\right)^{q_{y^-}(0)+s_{\lambda_y}}\right)^{k_{\lambda_y}} \end{pmatrix}$$

When DO need s restore some of the system properties, DO encrypts the extracted attribute set Γ′ with the shared key KEK and sends it to the DSSP. Then the DSSP executes the algorithm Re *CoverEncrypt* $\left(CT', \Gamma'\right)$. DSSP re-encrypts the ciphertext for $\forall y \in T'$, the algorithm is constructed as follows:

$$CT' = \begin{pmatrix} \Gamma', C' = M \cdot e(g,g)^{\alpha s}, C = g^{\beta s}, \\ \forall y \in Y : C_{y^+} = g^{q_{y^+}(0)}, \\ C' = \left(H\left(\lambda_{y+}\right)^{q_{y^+}(0)}\right)^{k_{\lambda_y}}, C_{y^-} = g^{q_{y^-}(0)}, \\ \forall y \in T' : C'_{y^-} = \\ \left(H\left(\lambda_{y-}\right)^{q_{y^-}(0)+s_{\lambda_y}}\right)^{k_{\lambda_y}} / \left(H\left(\lambda_{y^-}\right)^{s_{\lambda_y}}\right)^{k_{\lambda_y}} \end{pmatrix}$$

The construction of the above scheme makes the change of the access policy of the data owner more flexible, and does not need the data to do the re-encryption work.

5 Security analysis

5.1 Algorithm security analysis

ABAC-NVE is constructed based on the ABAC-ER scheme. The main difference is that the access policy tree is extended in the encryption algorithm through adding the positive and negative attribute nodes for each leaf node. In addition, it needs to extend the access policy tree in the encryption algorithm, and requires for user attributes to satisfy the access policy tree in the decryption algorithm. The security of ABAC-NVE scheme is equivalent to the security of the ABAC-ER scheme algorithm. As a result of the security of the cryptography generation and decryption algorithm was proved in the ABAC-ER algorithm.

5.2 Confidentiality of access strategy

The re-encryption in this paper is still only for the leaf nodes of the access tree. When the DO changes the access policy, the being encrypted set of attribute that need to update will be sent to the DSSP following the KEK key tree mechanism adopted in the ABAC-ER scheme. The Re *MoveEncrypt* (CT', T) or Re*CoverEncrypt* (CT', Γ') algorithm is used to re-encrypt the ciphertext. This does not involve the intermediate nodes of the access policy tree in the algorithm construction, so the access control policy is still secure.

5.3 Data confidentiality

When the attribute is deleted by DO in the re-encryption algorithm, the DSSP modifies the secret of the virtual node (negative attribute node) that will delete the attribute. According to Shamir's secret sharing scheme, the secret partner is reduced, that is, the threshold is less than $\tau - 1$, where τ represents the threshold of the secret shared partner, this is $e\left(D_i, C_{y^-}\right) / e\left(D'_i, C'_{y^-}\right) \neq e(g,g)^{rq_{y^-}(0)}$, so the access user will not be able to decrypt the plaintext $e(g, g)^{rs}$, thus the algorithm ensures the confidentiality of the data.

6 Performance analysis

6.1 Performance analysis

In this scenario, because the weak client computing power is low, the memory space is small and so on, we transfer the cumbersome bilinear operation from the client to the DSSP as far as possible. So the client only needs to complete a smaller amount of calculation in the attribute password generation and attribute revocation process.

The efficiency of the right revocation and recovery in ABAC-NVE and ABAC-ER schemes is shown in Table l.

Table l Analysis of the calculate amount

Contrast item	*ABAC-NVE*	*Our scheme*
DO's cost of generating the ciphertext for the first time	$E = 2n \cdot t_e + t_\varepsilon$	$E' = 4n \cdot t_e + t_\varepsilon$
DSSP's cost of generating the ciphertext for the first time	$R = 4n \cdot t_e + t_\varepsilon$	$R' = 2n \cdot t_e$
DO's cost of generating the ciphertext for revoking the policy attribute	$E = 2(n+\lvert T \rvert) \cdot t_e + t_\varepsilon$	–
DSSP's cost of re-encrypted the ciphertext for revoking the policy attribute	$R = (n-\lvert T \rvert) \cdot t_e$	$R' = \lvert T \rvert \cdot t_e$
DO's cost of generating the ciphertext for restoring the policy attribute	$E = 2(n-\lvert T \rvert) \cdot t_e + t_\varepsilon$	–
DSSP's cost of re-encrypted the ciphertext for restoring the policy attribute	$R = (n+\lvert T \rvert) \cdot t_e$	$R' = \lvert T \rvert \cdot t_e$

where t_ε is the time required for a bilinear pair of operations, t_e is the primary exponentiation operation time, n is the number of attributes of the access tree Γ′, $\lvert T \rvert$ is the number of attributes of the attribute change attribute set, and $\lvert T \rvert \leq n$.

As shown in Table l. the cost of modular exponential computing on DO and DSSP in the encryption phase is about twice that of the ABAC-ER scheme while first generating the ciphertext in ABAC-NVE, its computing cost in the first encryption is more,

When the policy attributes change, the computing cost of both DO and DSSP in our scheme has greatly improved, the DSSP computing cost also has greatly improved.

6.2 Simulation results analysis

The environments configuration of experimental verification is as follow Windows7 64-bit, Intel (R) Core i7-4790 CPU, 8 GB memory. The related operations use the MIRACL library and the bilinear development kit PBC library file, and version number PBC-0.4.7-vc.

When our scheme is compared with ABAC-ER, the number of attributes in the two schemes is 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 respectively, regardless of the generality, assuming $| T |= 3$ compares the computational cost of the DSSP in the attribute revocation and recovery. The result analysis of the calculation cost of the DO and the DSSP for the first encryption is shown as Figures 3 and 4.

Figure 3 Computes time comparison of first encryption on DO

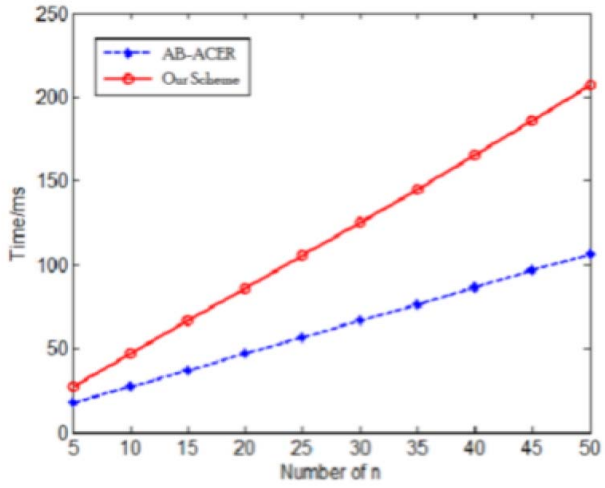
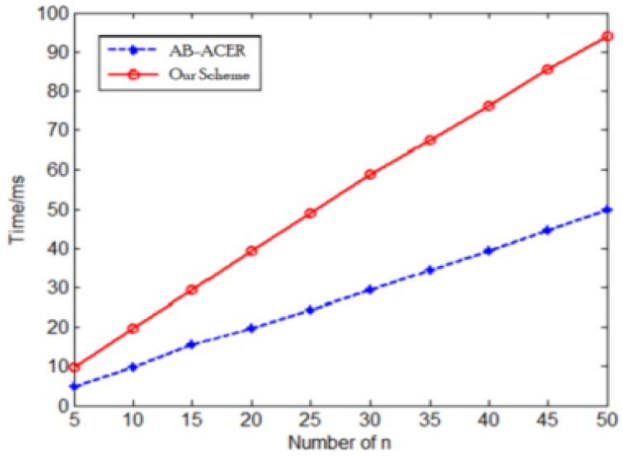


Figure 4 Compute time comparison of first re-encrypted on DSSP



Figures 3 and 4 reflect the computational cost of the data being the primary DO and the DSSP at the first encryption. Obviously, ABAC-NVE scheme in the first encryption process, because the access control policy tree has been expanded, the number of processing attributes doubled, so the first encryption calculation cost increased.

The DO in the attribute policy changes does not need re-encryption of the plaintext, so there is no calculation of cost. Here, we only compare the cost of our scheme with ABAC-ER's on DSSP in the strategy attribute revocation.

Figures 5 and 6 show that the calculation cost on DSSP in the original ABAC-ER scheme is related to the number of attributes in the policy attribute revocation and recovery process. The more attributes the greater the cost. The revocation of the policy attribute in the ABAC-NVE scheme only requires less bilinear computation. It maintains almost a constant cost.

Figure 5 Calculation time comparison of DSSP re-encrypts as the revoked attribute

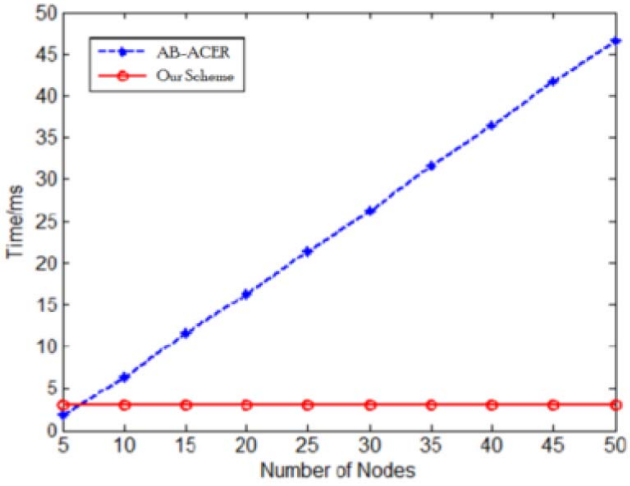
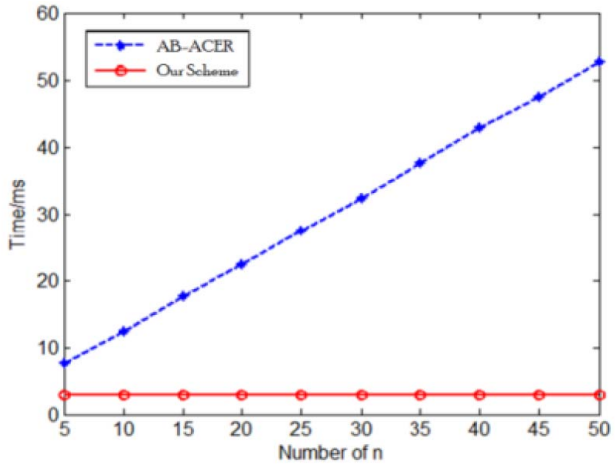


Figure 6 Calculation time comparison of DSSP re-encrypts as the restored attribute



The above analysis show that the ABAC-ER scheme needs completely to re-encrypt the plaintext at the revoking authority, and this ABAC-NVE scheme in the first encryption needs to pay more computing cost, but in exchange the flexible changes of policy attributes are supported. The revocation and recovery of attributes are realised only by DSSP without more computing cost on DO. Therefore, in the cloud storage environment, the ABAC-NVE scheme is more suitable in the situations which need frequently to change the policies on DO, and it is convenient for the data master to implement the flexible and efficient policy changes.

7 Conclusion

The problem, which the policy attribute revocation on CP-ABE is expensive and not flexible in the open network environment, is studied. We propose a minimum shared re-encryption key attribute set access control scheme that supports the revocation of policy attributes on the basis of ABAC-NVE. We demonstrate that the proposed scheme not only keeps the original security and fine-grained access control, but also has better flexibility and efficiency.

References

Bethencourt, J., Sahai, A. and Waters, B. (2007) 'Ciphertext-policy attribute-based encryption[C]', *IEEE Symposium on Security and Privacy*, IEEE Computer Society, pp.321–334.

Hur, J. and Noh, D.K. (2011) 'Attribute – based access control with efficient revocation in data outsourcing systems [J]', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 22, No. 7, pp.1214–1221.

Li, J., Wang, Y., Zhang, Y., et al. (2016) 'Full verifiability for outsourced decryption in attribute based encryption[J]', *IEEE Transactions on Services Computing*, Vol. 99, pp.1–1.

Liang, X., Cao, Z., Lin, H., et al. (2009) 'Attribute based proxy re-encryption with delegating capabilities[C]' *International Symposium on Information, Computer, and Communications Security*, ACM, pp.276–286.

Ma, H., Zhang, R. Wan, Z., et al. (2015) 'Verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing[J]', *IEEE Transactions on Dependable and Secure Computing*, pp.1–1.

Naor, D., Naor, M. and Lotspiech, J. (2001) 'Revocation and tracing schemes for stateless receivers[C]', *Proceedings of the International Cryptology Conference Advances in Cryptology (CRYPTO'01)*, pp.41–62.

Pirretti, M., Traynor, P., Mcdaniel, P., et al. (2010) 'Secure attribute-based systems[J]', *Journal of Computer Security*, Vol. 18, No. 5, pp.799–837.

Tan, H., Ma, M., Labiod, H., et al. (2016) 'A secure and authenticated key management protocol (SA-KMP) for vehicular networks[J]', *IEEE Transactions on Vehicular Technology*, Vol. 65, No. 12, pp.9570–9584.

Ueda, E.T. and Ruggiero, W.V. (2012) 'A systematic mapping on the role-permission relationship in role based access control models[J]', *IEEE Latin America Transactions*, Vol. 10, No. 1, pp.1243–1250.

Veugen, T. (2015) 'Linear round bit-decomposition of secret-shared values[J]', *IEEE Transactions on Information Forensics and Security*, Vol. 10, No. 3, pp.498–506.

Waters, B. (2011) 'Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization[J]', *Lecture Notes in Computer Science*, pp.321–334.

Zhou, L., Varadharajan, V. and Hitchens, M. (2015) 'Cryptographic role-based access control for secure cloud data storage systems[J]', *IEEE Transactions on Information Forensics & Security*, Vol. 10, No. 11, pp.2381–2395.

Zhu, Y., Huang, D., Hu, C.J. et al. (2015) 'From RBAC to ABAC: constructing flexible data access control for cloud storage services[J]', *IEEE Transactions on Services Computing*, Vol. 8, No. 4, pp.601–616.