J. Cent. South Univ. (2017) 24: 807−818
DOI: 10.1007/s11771-017-3483-z

Springer

NC-MACPABE: Non-centered multi-authority proxy re-encryption based on CP-ABE for cloud storage systems

XU Xiao-long(徐小龙)[1, 2], ZHANG Qi-tong(张栖桐)[1], ZHOU Jing-lan(周静岚)[2]

1. College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China;
2. State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210046, China

Abstract: The cloud storage service cannot be completely trusted because of the separation of data management and ownership, leading to the difficulty of data privacy protection. In order to protect the privacy of data on untrusted servers of cloud storage, a novel multi-authority access control scheme without a trustworthy central authority has been proposed based on CP-ABE for cloud storage systems, called non-centered multi-authority proxy re-encryption based on the cipher-text policy attribute-based encryption (NC-MACPABE). NC-MACPABE optimizes the weighted access structure (WAS) allowing different levels of operation on the same file in cloud storage system. The concept of identity dyeing is introduced to improve the users' information privacy further. The re-encryption algorithm is improved in the scheme so that the data owner can revoke user's access right in a more flexible way. The scheme is proved to be secure. And the experimental results also show that removing the central authority can resolve the existing performance bottleneck in the multi-authority architecture with a central authority, which significantly improves user experience when a large number of users apply for accesses to the cloud storage system at the same time.

Key words: cloud storage; data privacy; proxy re-encryption; multi-authority

1 Introduction

Cloud computing technology is proposed to aggregate all the hardware and software resources such as computing, storage and other information in WAN and LAN to solve the requirements of large-scale computing and mass data processing [1]. Cloud storage focuses on integrating different types of storage devices together to provide a large-scale storage resource pool, which has become a cost-effective solution for many users with the demand of data storage [2].

At present, data security protection mechanisms provided by cloud storage systems are not strong enough to protect users' data privacy or security [3−7]. Salesforce's cloud service system was attacked in 2007, which made a lot of users' private data leaked; in 2009, Microsoft, Amazon and some other companies' cloud service systems all encountered major failures, resulting in tens of thousands of users' stored data lost and information services affected; in 2010, two Google employees invaded into Gtalk, Google Voice and other Google's systems, causing users' data leaked; AWS's user agreement in 2010 announced clearly that AWS could not guarantee the safety of users' data; in 2011, the Chinese software develop net (CSDN) had more than 600 million users' data stolen and disclosed by hackers; in 2014, users' private digital photos were leaked out from Apple's iCloud.

The separation of data management and ownership is the root of the security problem of cloud storage [5]. When a user uploads his data to a cloud storage system, he will lose the substantial control ability to his data, and be unable to make sure that the storage, processing and transmission of his data are protected effectively. Instead, the cloud storage service provider obtains the data's substantial control power. Therefore, if there are internal staff derelictions, hacker attacks or system faults, the cloud storage service provider cannot provide sufficient evidences to assure that the user's data are stored and processed correctly without being extracted and analyzed, or sold to others, and can be completely destroyed in accordance with user requirement, etc.

According to the white paper released by the Cloud Security Alliance [8], the protection ability of data security in cloud storage system can be improved in two ways, the efficient encryption algorithms and the suitable access control mechanisms. Traditional encryption

Foundation item: Projects(61472192, 61202004) supported by the National Natural Science Foundation of China; Project(14KJB520014) supported by the Natural Science Fund of Higher Education of Jiangsu Province, China
Received date: 2015−09−07; **Accepted date:** 2016−03−22
Corresponding author: XU Xiao-long, PhD; Tel: +86−13813885172; E-mail: xuxl@njupt.edu.cn

methods have three defects [9], which makes them unsuitable for protecting data in cloud storage system directly. First, when the cloud storage system encrypts a user's data, it must obtain the user's public key. Second, the cloud storage system takes up extra system and network resources in the process of receiving each user's public key to encrypt information and sending ciphertext back to users. Third, the cloud storage service provider needs to get the list of users before providing services, while the identity information of users is difficult to obtain in one time under the distributed, large-scale cloud environment and user's privacy can be violated easily by illegally applications. The traditional access control models are not suitable for cloud environment, either. The trusted third party is introduced to protect the safety and integrity of data and realize the fine-grained access control to data and the data security independent on the security of the underlying file system, especially for untrusted network environments [10]. Depot, a cloud storage system that minimizes trust assumptions is designed and implemented, which ensures that the updates observed by correct nodes are consistently ordered under Fork-Join-Causal (FJC) consistency, and implements protocols that use this consistent ordering of updates to provide other desirable consistency, staleness, durability, and recovery properties [11]. Venus, a service for securing user interaction with untrusted cloud storage is presented, which guarantees integrity and consistency for applications accessing a key-based object store service, without requiring trusted components or changes to the storage provider [12]. Venus completes all operations optimistically, guaranteeing data integrity. It then verifies operation consistency and notifies the application. Airavat, a MapReduce-based system is presented, which provides strong security and privacy guarantees for distributed computations on sensitive data, and is a novel integration of mandatory access control and differential privacy [13]. Data providers control the security policy for their sensitive data, including a mathematical bound on potential privacy violations. The federated identity management is combined together with the hierarchical identity-based cryptography (HIBC), not only the key distribution but also the mutual authentication can be simplified in the cloud [14]. The trusted third party audit institutions to the basic cloud computing model are introduced, identifiing the difficulties and potential security problems of direct extensions with fully dynamic data updates from prior works, an elegant verification scheme for seamless integration of these two salient features is constructed, and the proof of retrievability model by manipulating the classic Merkle hash tree (MHT) construction for block tag authentication to achieve efficient data dynamics was improved [15]. The privacy, an approach by which confidential data is stored in a highly distributed database, partly located on the center of cloud and partly on the clients is presented [16]. Data sharing is based on the simple grant-and-revoke permission of shared data and the row-level data encryption for fine-grained data access control. Existing schemes mostly focus on user data protection rather than personally identifiable information protection. An implicit data partitioning scheme based on confusion for user data protection is presented [17]. For identifiable information protection, a cloud storage architecture based on the trust server concept is also presented, which isolates the data storage and the personal information management. The cloud server decides whether customers have the right to store personal data according to the storage authentication code provided by the trust server. Storage of the customers' identifiable information onto trust servers effectively protects the private information. A computable encryption scheme based on vector and matrix calculations (CESVMC) is proposed [18], which realizes the encryption of data, and supports fuzzy string search and basic arithmetic calculations (addition, subtraction, multiplication and division on encrypted data.

In order to realize low-cost data privacy protection mechanism for cloud storage, we proposed NC-MACPABE, the non-centered multi-authority proxy re-encryption based on CP-ABE for cloud storage systems. NC-MACPABE optimizes the weighted access structure, makes different users have different permissions to access the same files, and introduces the concept of identity dyeing to improve the protection ability of data privacy. The proxy re-encryption algorithm is improved in the scheme so that the data owner can revoke user's access right in a more flexible way.

2 Related works

SHAMIR [19] proposed an identity-based encryption (IBE), which enables any pair of users to communicate securely and to verify each other's signatures without exchanging private or public keys, without keeping key directories, and without using the services of a third party. Soon after, BONEH et al [20] realized IBE for the first time. Fuzzy IBE was presented which can be applied to enable encryption using biometric inputs as identities [21]. The error-tolerance property of a fuzzy IBE scheme is precisely what allows for the use of biometric identities, which inherently will have some noise each time when they are sampled. On the basis of Fuzzy IBE [22]. A new cryptosystem for fine-grained sharing of encrypted data called key-policy attribute- based encryption (KP-ABE) is developed [23].

In KP-ABE, ciphertexts are labeled with sets of attributes and private keys are associated with access structures that control which ciphertext a user is able to decrypt. A system for realizing complex access control on encrypted data called ciphertext-policy attribute-based encryption (CP- ABE) is presented, the encrypted data can be kept confidential even if the storage server is untrusted. Moreover, CP-ABE is secure against collusion attacks. Previous ABE systems use attributes to describe the encrypted data and build policies into user's keys, while CP-ABE attributes are used to describe a user's credentials, and a party encrypting data determines a policy for who can decrypt. Thus, CP-ABE is conceptually closer to traditional access control methods such as role-based access control (RBAC).

Both KP-ABE and CP-ABE mechanisms can be applied to express more flexible access control policy. However, ABE-based mechanisms themselves have some important issues to be solved. First, the authority generates and distributes private keys for users, and user's private key is only related to property, not related to user's private information (such as user's ID). If a private key is leaked out, it would be impossible to determine whether the authorized institution or the users should take the responsibility. Second, for users' private keys are associated with their attributes, the dynamically revocations of users' permissions will bring the huge overhead of system. In addition, since the encryption policy in CP-ABE is set by the data owner, complex strategies may make the public key system quite complex, limiting the flexibility of the access strategy.

However, CP-ABE does not require the fully credibility of authority, which is more suitable for distributed, multi-tenants, not fully trustworthy cloud storage systems. But if the management of attributes is done by one single authority, the authority will be overloaded, which is easy to be the performance bottleneck of the whole system, and to face the risk of security. In multi-authority-based ABE, when a user applies for accessing a file, he/she needs to request the decryption key from each authority, which decides whether or not to grant the user the decryption key by verifying whether the user has the attributes to access the file. Multi-authority-based ABE divides the entire set of attributes into different attribute domains so that the security risks caused by the invasion into single authority can be shared by multiple authorities.

The architecture of multi-authority-based ABE can be centralized or decentralized:

1) Centralized architecture

The global identifier (GID) and the pseudorandom function (PRF) are utilized, and a multi-authority-based ABE with the central trustworthy authority is presented [24]. Each authority generates and provides decryption key independently for users only by interacting with the central trustworthy authority, preventing the conspiracy of users if the number of users does not exceed m. The central trustworthy authority only participates in the grant of users' private keys, rather than the management of system or the validation of users' attributes.

The centralized architecture for ABE requires that the central authority has to be completely trusted, for it can get the users' private keys to decrypt cipher text. A scheme that allows the central authority to be "honest but curious" is proposed, which means that the credible institutions can honestly execute tasks according to the agreement, but maybe try to decrypt stored ciphertext curiously at the same time [25].

2) Decentralized architecture

In order to avoid the potential safety risks from the central authority, a threshold multi-authority fuzzy identity based encryption scheme (MA-FIBE) without a central authority is presented [26]. An encrypter can encrypt a message such that a user could only decrypt if he/she has at least d_k of the given attributes about the message for at least $t+1$, $t \leq n/2$ honest authorities of all the n attribute authorities in this proposed scheme. The security proof is based on the secrecy of the underlying joint random secret sharing protocol and joint zero secret sharing protocol and the standard decisional bilinear Diffie-Hellman (DBDH) assumption. The proposed MA-FIBE could be extended to the threshold multi-authority attribute based encryption scheme (MA-ABE) and further extended to a proactive MA-ABE scheme.

However, the decentralized multi-authority-based ABE still suffers some shortcomings. For example, since the trusted central authority in the system has been removed, each authority needs to interact one another to ensure that users get complete and correct decryption private keys; authorities might collude with each other to get the complete set of a user's attributes and then do harm to the user's private information.

Proxy re-encryption was presented by BLAZE et al [27] at the 1998 European cryptography annual meeting. The half-credible proxy, which obtains certain additional information, such as re-encryption keys, uses one user's public key to encrypt the plaintext, and then gets the ciphertext re-encrypted with another user's public key. In this process, except the re-encryption key, the proxy does not get any information about the plaintext. Ateniese presents the proxy re-encryption specifications at the 2005 Network and Distributed System Security Symposium and the 2007 ACM Conference on Computer and Communications Security, solving the problem that the previous schemes can only resist the chosen plaintext attacks (CPA), which is able to resist the chosen ciphertext attacks (CCA) [28]. ABE associates attributes

with private keys, which means that the change of user's permission will change the conditions of access control, and the re-encryption will bring the significant overhead.

3 System model

3.1 Components

NC-MACPABE proposed in this work gets rid of the central authority, which includes three kinds of cloud storage entity, as shown in Fig. 1.

1) Attribute authority. The whole attribute set is divided into k disjoint sets managed k attribute authorities respectively. Each attribute authority needs to calculate its own master key independently during the initialization phase, and generates the attribute private key components for all users during the following data sharing phase, then sends the re-encrypted results to users.

2) Cloud-side data processing module. The cloud-side data processing module is based on the powerful cloud-side clusters with powerful processing and storing capacities, responsible for data storage, proxy re-encryption, authentication of signature, etc. Meanwhile, the cloud-side data processing module saves the operation validation set for each file in order to realize the fine-grained management of user's access right.

3) User. Users of a cloud storage system include data owners and data consumers. Any user can be a data owner and data consumer at the same time. A data owners can obtain the public key from any attribute authority and use it to encrypt the data ready to be uploaded to the cloud-side system. When a data consumer wants to access a file, he/she has to obtain the private key components from all attribute authorities to generate the private key at the client side.

3.2 Workflow

As mentioned above, NC-MACPABE is based on the decentralized architecture to solve the problems of security and performance bottleneck because the central authority, which is more suitable for the public cloud storage systems serving a large-scale users.

NC-MACPABE utilizes the identity dyeing technology to realize the interaction between users and attribute authorities with dyed names instead of with GIDs, which can prevent multi-attribute authorities conspire to steal users' identity information. The workflow of NC-MACPABE is composed of seven phases, including initialization, data segmentation, identity dyeing, key generation, data encryption, data decryption and revocation of access right.

1) Setup: Setup $\rightarrow$ PK, MK_k

All attribute authorities have to execute the setup algorithm to calculate the public key Y of the whole system together, release the public parameter (PK), and

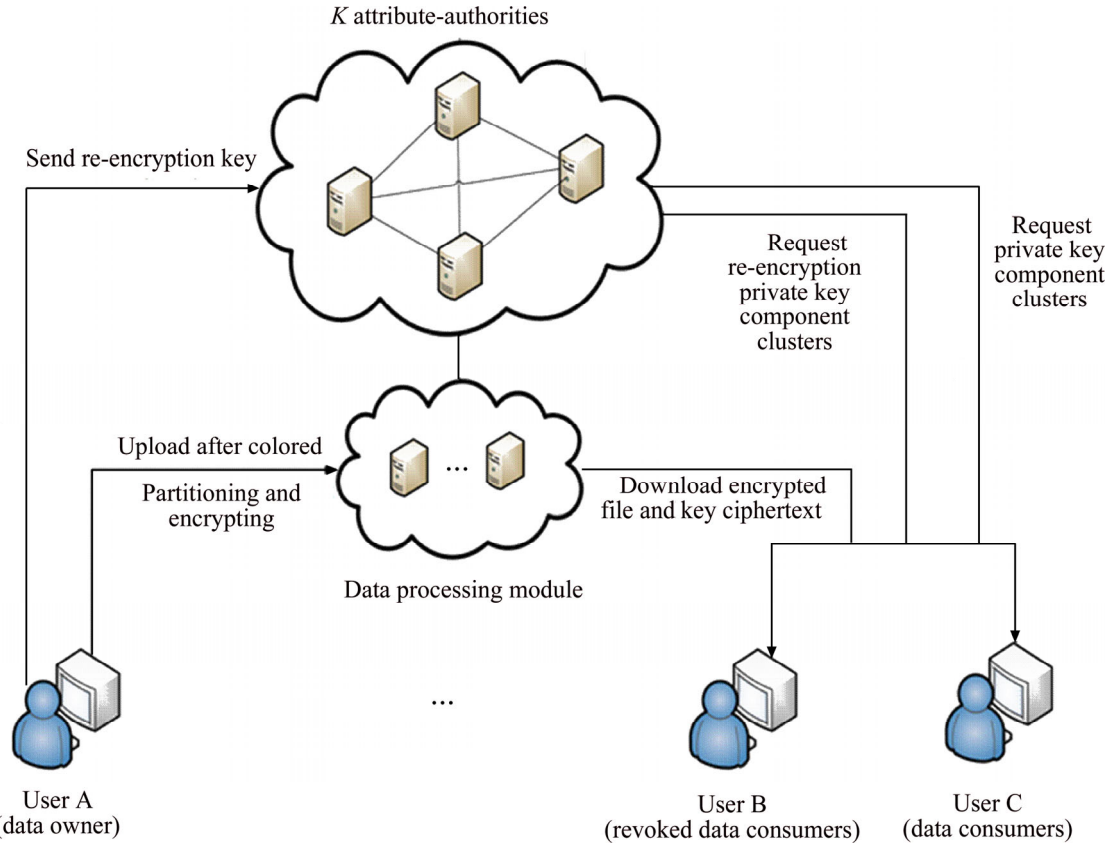

Fig. 1 System architecture

then calculate their own master key (MK_k) respectively.

2) Data Segmentation: $\text{Part}(F) \rightarrow F_s, F_l$

A user executes the segmentation algorithm to partition file F into a small block (F_s) and a large block (F_l).

3) Identity dyeing: $\text{Dye}(\text{GID}_u) \rightarrow \text{color}_{(u,k)}$

User u inputs his/her own GID_u, executes the dyeing algorithm for many times, and then outputs the different dyed name $\text{color}_{(u,k)}$.

4) Key generation: $\text{KeyGen}(\text{PK}, \text{MK}_k, A^u, \text{color}_{(u,k)}) \rightarrow \{D_i^{(1)}\}_{i \in A^k \cap A^u}, D_i^{(2)}$

Each attribute authority executes the key generation algorithm and outputs u's attribute private key component cluster $\{D_i^{(1)}\}_{i \in A^k \cap A^u}$ and the authority component $D_i^{(2)}$ according to $\text{color}_{(u,k)}$, the user attribute set A^u, the system public parameter PK and MK_k, and A^k which is the part of attribute set according to F managed by authority k and related to u's private key SK_u and GID_u.

5) Data encryption: $\text{Encrypt}(\text{PK}, F_s, \text{WASL}) \rightarrow (C_k, \text{OV})$

Before the user uploads F, F_s is used as a symmetric key to encrypt F_l. PK, F_s and the weighted access structure list (WASL) are able to differentiate operation rights, and then the ciphertext C_k and the operational validation set (OV) are outputted.

6) Data decryption: $\text{Decrypt}(\text{PK}, (D_i^{(1)}), D_i^{(2)}, C_k) \rightarrow (F_s, p_{\text{OV}})$

For any encrypted files stored in the cloud storage system, all users can download it. However, only the users who reach the specific access right defined by the weighted access tree (WAT) can decrypt the symmetric key and then use it to decrypt the ciphertext. The data decryption algorithm can be divided into two steps.

Step 1: Generate user private key. After u obtains all $D_i^{(1)}$ and $D_i^{(2)}$, SK_u can be generated.

Step 2: Decrypt and verify the access righter. u inputs PK, C_k and SK_u, and then outputs F_s and the root node secret parameter p_{OV}. When a user wants to access F, he/she needs to get the threshold number corresponding to operating level OV from the cloud-side data processing module.

7) Revocation of access right

There are two modes of access right revocation, the direct access right revocation mode and the indirect access right revocation mode, which can be applied in one system.

Direct access right revocation mode. If a data owner wants to revoke access rights of specific users, the direct access right revocation mode is suitable. The GID_{ru} of a user whose access right will be revoked needs to be identity-dyed, and then sent to each attribute authority.

Indirect access right revocation mode. If a data owner wants to change certain attributes related users' access rights, the indirect r access right revocation mode is suitable. $\text{ReEncrypt}()\text{PK}, C_k, \text{WAT}, \text{SK}_u \rightarrow (C_k', OV')$, the indirect access right revocation is implemented with the proxy re-encryption. First, the re-encryption key cluster $\{rk_{i \rightarrow i'}\}$ is generated, then the ciphertext of key is re-encrypted, and finally, users' (whose access rights haven't been revoked) private key component cluster $\left\{D_{k,i}' = \left(D_{k,i}\right)^{rk_{i^{(n)} \rightarrow i^{(l)}}}\right\}$ is updated.

4 Algorithm

4.1 Assumptions about security

NC-MACPABE is based on the following three assumptions about security:

1) Trustworthy users.

Data owners partition and encrypt files locally before uploading them, while data consumers generate their private key after receiving all necessary attribute key components. So users need to be trustworthy, whose private information will be kept safe.

2) Honest but curious attribute-authorities.

The attribute-authorities are allowed to be "honest but curious", which means they are semi-trustworthy. We assume that they generate users' private attribute key components according to the rules set by the system, but maybe curiously try to get as much users' information as possible at the same time.

3) Honest but curious cloud-side data processing module.

Users' data uploaded to cloud storage system may be segmented and then stored in different cloud servers. Storage servers are managed by system administrators instead of users. So these storage servers cannot be trusted completely. However, completely untrustworthy servers cannot be used to execute the task of re-encryption. The data processing module is assumed to execute operations as expected, and the low possibility of ciphertext being disclosed in cloud can also be tolerated. That is why the data processing module is also described as semi-trustworthy, i.e. honest but curious.

4.2 Initialization

All attribute authorities in the system should complete the phase of initialization.

Step 1) An attribute authority (e.g. A_m) chooses one p-order double-linear group G_0. g_1 and g_2 are generators of G_0.

Step 2) A_m selects $v_m \in Z_p$, and sent $Y_m = e(g_1, g_2)^{v_m}$ to other authorities. Each authority calculates $Y = \prod Y_m = e(g_1, g_2) \sum_{m \in \{1, \cdots, k\}} v_m$ independently.

Step 3) A_m and A_n, other authorities, share the seed pair $(s_{mn} \in Z_p, s_{nm} \in Z_p)$ $(s_{mn} \neq s_{nm})$ between A_m and A_n only belonging to them. After A_m receives the seed pairs $(s_{mn}, s_{nm})_{n\in\{1,\cdots,k\}\backslash\{m\}}$ shared with other $k-1$ authorities respectively, A_m calculates the authority's parameter $x_m = g^{\sum(s_{mn}-s_{nm})_{n\in\{1,2,\cdots,k\}\backslash\{m\}}}$.

Step 4) For each attribute $i \in \{1,\cdots,n_m\}$, A_m selects $t_{m,i} \in Z_p$ randomly, computes $T_{m,i} = g_2^{t_{m,j}}$, and stores $\left\langle x_m, \{s_{mn}, s_{nm}\}_{n\in\{1,\cdots,k\}\backslash\{m\}}, \{t_{m,i}\}_{i\in\{1,\cdots,n_m\}}\right\rangle$.

Step 5) A_m's master key is $\mathrm{MK}_m = \{v_m, x_m\}$. The system releases the public key $\mathrm{PK}=\left\{G_0, Y=\hat{e}(g_1,g_2)^{\sum v_m}, \left\{T_{m,i}=g_2^{t_{m,i}}\right\}_{i\in\{1,\cdots,n_m\},m\in\{1,\cdots,k\}}\right\}$

4.3 Data segmentation

The data segmentation module will partition a file into two data blocks, a large one and a small one. There are two plans of data segmentation, one is the fixed-size segmentation and the other is the non-fixed-size segmentation. The flowchart of the fixed-size segmentation is shown in Fig. 2.

Step 1) Before the data owner upload file F, the client-side system generates a random sequence of 0−N (N is the size of file with bit number). The length of the sequence is equal to the size of the small block, e.g. 168 bits.

Step 2) The client-side system sorts the sequence ascending to get the positions of the bytes to be extracted from F.

Step 3) The client-side system extracts bytes from F sequentially with order.

Step 4) The client-side composes the extracted bytes to be the small block F_s, and takes the rest of the file as the large block F_l.

After the segmentation of the file, the client-side system will use F_s as the symmetrical encryption key to encrypt F_l to get the ciphertext C_F. The advantage of the fixed-size segmentation is that the size of the small block is fixed, which is easy for the client-side management. However, the random number generated may be unevenly distributed, which leads to a large section of continuous data in the large block.

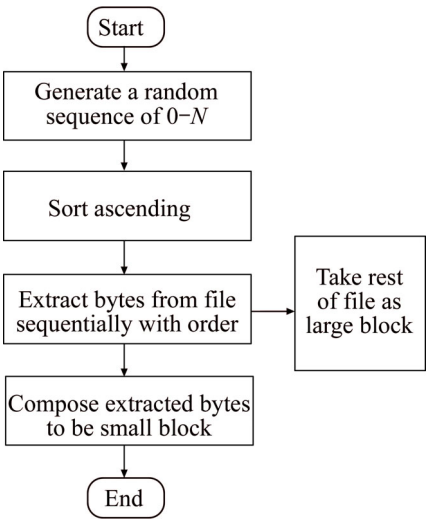

Fig. 2 Flowchart of fixed-size segmentation

4.4 Identity dyeing

When a user (e.g. u) joins the system, the system will generate GID_u, the global identity of u, and then execute the dyeing algorithm for several times to output the different colored names, which is used to interact with different attribute authorities in order to prevent conspiracies by multi-authorities.

The process of identity dyeing is shown in Fig. 3, including the following steps:

Step 1) A one-way anti-collision hash function $H_1: \{0,1\}^* \to Z_q^*$ is selected to generate GID_u for u.

Step 2) A random number sequence {c_1, c_2, ⋯, c_y, ⋯, c_k} is generated, where k is the number of the attribute authorities, and c_y $(y \in [1,k])$ is the colour factor corresponding to u.

Step 3) c_1, c_2, ⋯, c_y, ⋯, c_k are attached to GID_u respectively to get $\left\{\mathrm{GID}_u \| c_y\right\}_{y\in\{1,2,\cdots,k\}}$, where $\|$ is the operation of attachment.

Step 4) $H_2: \{0,1\}^* \to \{0,1\}^n$ acts on $\left\{\mathrm{GID}_u \| c_y\right\}_{y\in\{1,2,\cdots,k\}}$ respectively to output k colored names $\left\{\mathrm{color}_{(u,y)}\right\}_{y\in\{1,2,\cdots,k\}}$, where n is the length of colored name.

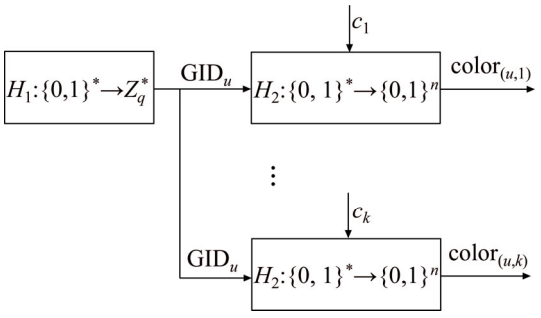

Fig. 3 Process of identity dyeing

4.5 Key generation

Suppose that u's attribute set is A^u. When u first joins the system, he/she will request the private key components from k attributes authorities with k colored names respectively to get the private key that matches A^u. The key generation process of each attribute authority composes of the following steps:

Step 1) An attribute authority (e.g. A_m) initiates $k-1$ independent anonymous key distribution protocols with

other attributes authorities. A_m randomly selects $R_m \in Z_p$, and exports the attribute authority's private key component $D_m^{(1)} = g_1^{R_m} x_m$.

Step 2) A_m calculates $p_m = v_m - \sum_{n\in\{1,\cdots,k\}\setminus\{m\}} R_m$.

For the users' each attribute $j \in A^u$, select the attribute $j \in A^u \cap A_m$, then query the storage list to get $t_{m,j}$, and finally generates the private key component cluster $\left\{D_{m,j}^{(2)} = g_1^{p_m/t_{m,j}}\right\}_{j\in A^u\cap A_m}$. The colored names and the corresponding private key components are stored in the file's attribute re-encryption key list, and the initial value of the attribute version is set as 1.

Step 3) A_m sends the attribute private key component cluster $\mathrm{SK}_{u,m} = \left(D_m^{(1)}, D_{m,j}^{(2)}\right)_{j\in A^u\cap A_m}$ generated by A_m to u.

4.6 Data encryption

The inputs of the data encryption algorithm include the public key PK, F_s and WAT. The outputs of the data encryption algorithm include F_s's ciphertext C_k and the operation validation (OV) set.

The process of data encryption composes of the following steps:

Step 1) F_s is used as the symmetric key to encrypt F_l to get the ciphertext C_F.

Step 2) The system generates WAT that can distinguish different levels of operations for different users with different attribute structures, as shown in Fig. 4.

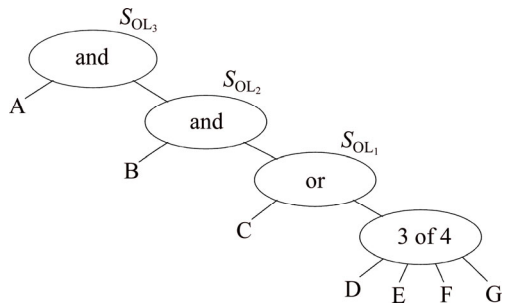

Fig. 4 WAT for distinguishing different operating levels

During the construction of WAT, all nodes are numbered, and the No. of root is 0. For each leaf node, $s_j \in Z_p$ is randomly selected as its secret value. That is to say, for each leaf node j, $q(j) = s_j$. For non-leaf nodes y, $q(y) = \sum_{\mathrm{chriden}(y)} q(x)\cdot V_a$, where chriden($y$) are all child nodes of y, and V_a is the threshold. The secret value of root node is s=q(0).

Step 3) OV is generated, which is only stored in the cloud-side data processing module, and cannot be downloaded by users, who can only request for the corresponding secret parameters of verification for obtaining the access right. $\mathrm{OV} = \left(Y^{s_{OL_1}}, Y^{s_{OL_2}}, Y^{s_{OL_3}}\right)$, where OL_0 means not allowed to access the file, OL_1 means having the right to reading the file, OL_2 means having the right to modify the file, OL_3 means means having the right to delete the file, and s_{OL_1}, s_{OL_2}, and s_{OL_3} are the secret parameters of the root node of WAT.

Step 4) The key ciphertext $C_k = \left\{E_0 = F_s \cdot Y^s, E_1 = g_2^{q(0)}, \left\{E_{k,i} = T_{k,i}^{q(0)}\right\}_{y\in u_k, k=1,\cdots,k}, \mathrm{WASL}\right\}$ is generated.

Step 5) The data owner specifies the unique ID for the ciphertext and uploads C_F, C_k and to the cloud-side system.

4.7 Data decryption

Any user is allowed to query and download the file. However, only those users with OL_1 can decrypt the corresponding C_k and then use it to decrypt C_F. The process of data decryption composes of the following steps:

Step 1) After u receives all the attribute private key component cluster $\mathrm{SK}_{u,m} = \left(D_m^{(1)}, D_{m,j}^{(2)}\right)_{j\in A^u\cap A_m}$ sent from attribute authorities, calculate $D_u = \prod_{m\in\{1,\cdots,k\}} D_m = g_1^{\sum_{m\in\{1,\cdots,k\}} R_m} \times g_1^{\sum_{(m,n)\in\{1,\cdots,k\}\times(\{1,\cdots,k\}\setminus\{m\})} (s_{mn}-s_{nm})} = g_1^{\sum_{m\in\{1,\cdots,k\}} R_m}$ according to $D_m^{(1)} = g_1^{R_m} x_m$. $\sum_{(m,n)\in\{1,\cdots,k\}\times(\{1,\cdots,k\}\setminus\{m\})} (s_{mn} - s_{nm}) = 0$, and u's private key has nothing to do with u's colored names. So, there is no need to execute the identity discoloration to recover the users' GID_u before decryption, ensuring that the decryption ability only depends on the user attributes. Let $R_u = \sum_{m\in\{1,\cdots,k\}} R_m$, and get $D_u = g_1^{R_u}$.

Step 2) Calculate the flowing formula:

$$e\left(D_{m,j}^{(2)}, E_{m,j}\right)\cdot e\left(D_u, E_1\right) = e\left(g_1, g_2\right)^{q(0)(v_m - \sum R_m)}.$$

$$e\left(g_1^{R_u}, g_2^{q(0)}\right) = Y^{q(0)}$$

Step 3) Each leaf node's value is used to calculate recursively to get its parent node's threshold value, $q = \sum_{\mathrm{chriden}(x)} q_x \cdot V_a$. When the recursive procedure reaches the root node with OL_1, if the threshold value is 1, $s' = q(0)$. The secret value of the root node, $e(g,g)^{s'(\sum v_m)} = Y^{s'}$, is sent to the cloud-side data processing module, and compared with the

corresponding value in OV. If they are equal, the user has the right to reading the file, and can decrypt C_k with $F_s=C_k/Y^s$ and use F_s to decrypt C_F to get F_l.

4.8 Revocation of access right

A data owner can revoke another user's access right to his data with the user's name directly, or indirectly revoke all those users' access right associated with particular attributes.

1) In order to revoke the access right of a certain user (e.g. u), the system needs to search the user identity list to get his/her GID_u. Then the identity is dyed to get the colored names $\{clour_{(u,k)}\}$ corresponding to each attribute authority. Suppose that the revoked user list is $R_k=\{clour_{r(u,k)}\}$, where $r(u, k)$ is the colored name corresponding to a certain attribute authority. R_k is sent respectively to each attribute authority.

2) In order to revoke all those users' access right associated with particular attributes, the version of attribute set RA should be updated. For each $i \notin RA$, $x_i' \in Z_p$ is selected randomly, and then calculate $rk_{i\leftrightarrow i'}=x'/x$; For $i \notin RA$, let $rk_{i\leftrightarrow i'}=1$, put $rk_{i\leftrightarrow i'}$ into the re-encryption key set, output $\left\{rk_{i\to i'}\right\}$, and finally add the version number with 1.

3) If the data owner wants to update the users' operation level of the file, WAT needs to be reconstructed, which means that s_{OL_1}, s_{OL_2} and s_{OL_3} need to be calculated again to generate the new operation validation set OV′.

4) C_k, $\left\{rk_{i\to i'}\right\}$ and u_F in the weighted access control structure corresponding to C_k are inputted. First, check whether the version number of $\{rk_{i\to i'}\}$ is the same with C_k. If so, for each $i \in RA$, calculate $E'_{k,i}=\left(E_{k,i}\right)^{rk_{i\to i'}}$ to get the new key cipher C'_k; if not, do nothing.

5) "Lazy updating" principle is adopted in this work, which means not immediately updating the private key component clusters of users whose access rights have not been revoked. When u applies to access the file, each attribute authority checks whether $clour_{(u,k)}$ provided by u exists in the revoked user list $R_k=\{clour_{r(u,k)}\}$. If not, this access application will be rejected directly.

6) If $clour_{(u,k)} \notin \{clour_{r(u,k)}\}$, the system will check whether the version of each attribute contained in current C_k is the same with the version of attribute private key component. If so, it means that C_k has not been re-encrypted or the attribute private key components distributed to u has been updated to the latest version, there will be no output. If $n<l$, the system will calculate $rk_{i^{(n)}\to i^{(l)}}=rk_{i^{(n)}}\times rk_{i^{(n+1)}}\cdots\times rk_{i^{(l)}}$, where n is the version number of private key component, l is the version of each attribute contained in current C_k. Finally, each authority outputs the updated attribute private key component cluster $\left\{D'_{k,i}=\left(D_{k,i}\right)^{rk_{i^{(n)}\to i^{(l)}}}\right\}$.

5 Security of scheme

5.1 Security model

Theorem1: NC-MACPABE is semantically secure enough, if none probabilistic polynomial-time adversaries have disregarded advantages in the game of random toss.

Proof: NC-MACPABE removes the central authority. It is convenient to expand WAT with OL_0 to the multi-level WAT. Assume that the access level of WAT corresponding to C_k is only OL_1, which does not affect the security of the system. NC-MACPABE is secure enough against the chosen plaintext attacks (CPA) or not if all probabilistic polynomial-time adversaries have negligible advantages in this game.

Setup: An adversary presents the access control structure T, corresponding to the attribute set u_k managed by each attribute-authority and the weighted access structure list (WASL), when he/she is waiting for challenge. The adversary also declares the attacked attribute-authorities. The challenger runs the setup algorithm to generate the public key and other parameters, and then sends them to the adversary.

Phase 1: The adversary asks each attribute authority separately for all attribute private key components corresponding to T. The challenger calculates the private key components firstly. To any GID, if at least one attribute of an attribute authority cannot be satisfied by T, the private key components cannot be used directly to decrypt the ciphertext.

Challenge: The adversary randomly outputs two plaintext messages M_0 and M_1 with the same length and GID. The challenger randomly tosses b $(b \in \{0,1\})$, get the corresponding ciphertext C according to the attribute set and T, and then sends C to the adversary.

Phase 2: The procedure of Phase 1 is executed repeatedly. The adversary continues to ask other related attribute private key components. The demanded ciphertext and the corresponding GID^* are required not to be exactly the same with C. In other words, GID^* can not be equal to GID of any private key that has been asked, which means the adversary cannot query the same user twice or more.

Guess: The adversary guesses which message has been encrypted by the challenger. The adversary outputs the guess b'. $b'=b$ indicates that the guess of the adversary is successful. The adversary's advantage is $\left|\Pr(b'=b)-\frac{1}{2}\right|$.

Theorem 2: In the above game, if all the

polynomial enemies have non-negligible advantage, the scheme will be able to withstand CPA.

Theorem 3: If in a game, an adversary can damage the scheme based on the above security model, there will be at least one probabilistic polynomial time algorithm that can break the DBDH problem with considerable advantages.

Proof: Suppose there is a probabilistic polynomial time adversary who can break the scheme at the advantage of ε. We can prove that the following DBDH game can be solved by the advantage of $\varepsilon/2$.

In the bilinear mapping $e: G_0 \times G_0 \to G_T$, G_0 is a p order prime cyclic group, and g is its generator. Suppose that $A=g^a$, $B=B^b$ and $C=C^c$. In the DBDH game, the challenger tosses a coin $\mu: \mu \in \{0,1\}$. If μ=0, $Z=e(g,g)^{abc}$, otherwise, $Z=e(g,g)^{\theta}$, where $a,b,c,\theta \in Z_p$ is selected randomly. In the following DBDH game, simulator β plays the role of challenger.

Initialization: The adversary generates the access control tree T_0^*. The nodes of T_0^* are all determined by the adversary. And then T_0^* is changed into the weighted access structure list WASL*.

Setup: β generates the version 1 public key parameter for the adversary, sets $Y=e(A,B)=e(g,g)^{ab}$ (supposing $\sum_{m\in\{1,\cdots,k\}} v_m = ab$), and then sends it to the adversary.

Phase 1: The adversary asks for private keys as many as possible. These private keys correspond to the attribute set $A^u=\{A_1,\cdots,A_q\}$. However, none of them can satisfy T_0^*. After β receives the request for key querying, it calculates these attribute private key components to reply the adversary. β randomly selects $R_m \in Z_p$. Each attribute authority (e.g. A_m) starts k−1 independent anonymous key distribution protocols to interact with other attribute authorities to output institutional private key components $D_m^{(1)}=g_1^{R_m}x_m$, saves its attribute version number as the initial value 1, and then for the property $t_{m,j}$ managed by itself calculates the attribute private key cluster $\left\{D_{m,j}^{(2)}=g_1^{p_m/t_{m,j}}\right\}_{j\in A^u\cap A_m}$. β sends the generated private key component cluster to the adversary.

Challenge: The adversary submits m_0 and m_1 to the challenger. The challenger randomly tosses $\gamma(\gamma\in\{0,1\})$ and then sends the ciphertext C_K^* to the adversary, $C_k^*=\left\{E_0=F_s\cdot Y^s, E_1=g_2^{q(0)}, \left\{E_{k,i}=T_{k,i}^{q(0)}\right\}_{y\in u_k,k=1,\cdots,k}, \text{WASL}^*\right\}$. If μ=0, $Z=e(g,g)^{abc}$. Suppose $ab=y_0$, and $c=s_0$, $Z=\left(e(g,g)^{ab}\right)^c=(Y)^c=Y^{s_0}$, and C_K^* is the determined ciphertext of message m_γ. If μ=1, $Z=e(g,g)^{\theta}$, and $\tilde{E}=m_\gamma\cdot e(g,g)^{\theta}$, where $\theta\in Z_p$ is a random value. So, E_0 is the random value of G_T to the adversary, and C_K^* does not contain any information about m_γ.

Phase 2: The adversary submits the request for the version l private key to β. Suppose that after the l round re-encryption, the adversary is not included in the list of revoked users. β executes in accordance with the procedure of phase 1. For the attribute $i\in$ RA, where RA is the attribute set to be updated, according to the adversary's attribute set's version n(n<l), β generates the attribute re-encryption key $rk_{i\to i'}$ for it , and replies the requests for generating l−n proxy re-encryption keys. β randomly selects $x_i'\in Z_p$, calculates $\left\{rk_{i^{(y)}}=x'/x\right\}_{y\in\{n,n+1,\cdots,y,\cdots,l\}}$ and then returns the version l re-encryption key $rk_{i^{(n)}\to i^{(l)}}=rk_{i^{(n)}}\times rk_{i^{(n+1)}}\cdots\times rk_{i^{(l)}}$. Finally, β executes the attribute private key re-encryption algorithm and calculates $t_{k,i^{(l)}}=\left(t_{k,i^{(n)}}\right)^{rk_{i^{(n)}\to i^{(l)}}}$.

Guess: The adversary submits the guess $\gamma'\in\{0,1\}$ about γ. If $\gamma'=\gamma$, β will output μ'=0, and $Z=e(g,g)^{abc}$, which means that a definite DBDH tuple is presented, and the adversary will get the confirmed ciphertext m_γ with the considerable advantage ε. Otherwise, β will output μ'=1, and $Z=e(g,g)^{\theta}$, which means that a random DBDH tuple is provided, so that the possibility that the adversary guesses correct is not higher than 1/2. In this DBDH game, the advantage of distinguishing Z successfully is $\frac{1}{2}\cdot\left(\frac{1}{2}+\varepsilon\right)+\frac{1}{2}\cdot\frac{1}{2}-\frac{1}{2}=\frac{\varepsilon}{2}$. To sum up, when a polynomial time adversary breaks this scheme with the advantages of ε, the adversary' advantages of $\varepsilon/2$ for solving the DBDH problem cannot be ignored, which proves that the scheme can achieve the semantic security under DBDH.

5.2 Security analysis

With NC-MACPABE, each attribute authority manages a part of the global attributes set, which protects the user's identity effectively. Besides, there is no central authority, and there is no list to record the information of users and their GIDs. Each user uses different dyed names to interact with different attribute authorities in order to prevent the conspiracy of multiple attribute authorities.

An attribute authority (e.g. A_m) randomly generates the secret parameter set $\{s_{mj}\}$ to share with other attribute authorities. Even if there are k−2 attribute authorities colluding, there will be at least 1 parameter unable to be

gotten. As long as at least two of the k attribute authorities are honest, the security of multi-attribute authority without the central authority with NC-MCPABE can be guaranteed.

6 Experiments and performance analysis

6.1 Experimental environment

We constructed the experimental platform base on a computer with 2.5 GHz Intel Core CPU, 4 GB memory, VMware Workstation and Ubuntu10.10, the development language C++, and the Cpabe-0.11 library, symmetric encryption uses a 168 bit 3DES encryption algorithm. The original code adopts the cpabe-0.11 library.

We used the 168 bit 3DES as the symmetric encryption algorithm, and used Cloudsim to create a simulated cloud data center. We created different number of virtual machines to simulate different number of attribute authorities, created different number of cloud tasks and bind them to the virtual machines to simulate the different number of attributes in the system. The proxy re-encryption algorithm is based on the JHU-MIT's Proxy Re-cryptography Library (PRL).

6.2 Experiments

We compared NC-MACPABE proposed in this work with the centralized multi-authority proxy reencryption mechanism based on ciphertext-policy attribute-based encryption (MPRE-CPABE) and the typical CP-ABE. For the initialization module does not involve attribute management operations, the performance of system has nothing to do with the number of the attribute, but only associated with the number of attribute authorities during the phase of initialization.

Figure 5 shows the time costs of NC-MACPABE and MPRE-CPABE with different numbers of attribute authority. For both NC-MACPABE and MPRE-CPABE, The larger number of attribute authority, the higher time costs. The time complexity of MPRE-CPABE is $O(k)$, for each attribute authority needs to interact with the central authority. The time complexity of NC-MACPABE is $O(k^2)$, for each attribute authority needs to interact with each other. Therefore, the time cost of NC-MACPABE is higher than the time cost of MPRE-CPABE during the phase of initialization. However, as the initialization algorithm is implemented only once when the system starts, the extra time cost of NC-MACPABE can be tolerated.

Table 1 shows the time costs of data segmentation. We select the data with different sizes, including 1 MB, 5 MB, 50 MB, 200 MB and 1 GB. It is easy to find that the time costs of data segmentation are acceptable, since most data uploaded to the cloud system are usually smaller than 1 GB.

Table 1 Time costs of data segmentation

Size of data/MB	Time/ms
1	32
5	47
50	109
500	702
1024	3051

Figure 6 shows the key generation times of NC-MACPABE, MPRE-CPABE and CP-ABE with different numbers of attribute and different numbers of attribute authority.The time costs of NC-MACPABE and MPRE-CPABE with 5 attribute authorities are lower than the time costs of NC-MACPABE and MPRE-CPABE with 2 attribute authorities, and the time costs of NC-MACPABE and MPRE-CPABE are lower than the time costs of CP-ABE. To sum up, NC-MACPABE has

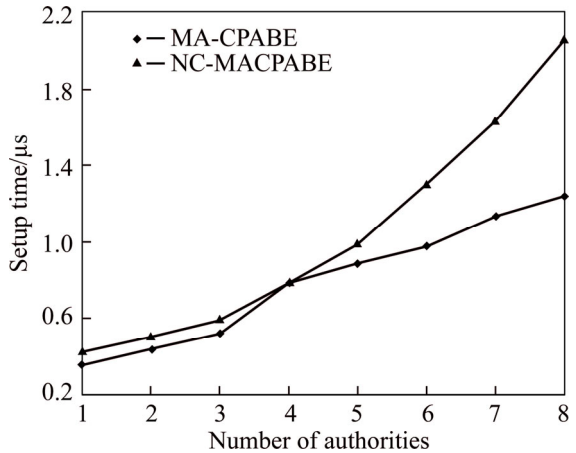

Fig. 5 Time costs of NC-MACPABE and MPRE-CPABE with different numbers of attribute authority

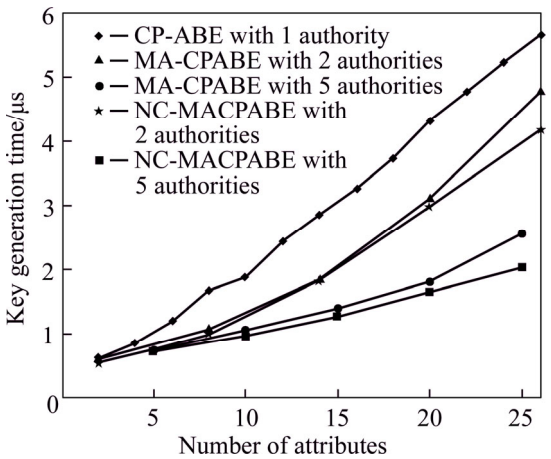

Fig. 6 Key generation time of NC-MACPABE, MPRE- CPABE and CP-ABE

the best performance of the key generation, for there is no central authority as the performance bottleneck in NC-MACPABE.

The performance of the symmetric key encryption has nothing to do with the number of attribute authority. Figure 7 shows the symmetric key encryption times of NC-MACPABE, MPRE-CPABE and CP-ABE with different numbers of attribute. The symmetric key encryption times of all NC-MACPABE, MPRE- CPABE and CP-ABE increase with the number of attribute increasing. The time costs of NC-MACPABE and MPRE-CPABE are similar and a little bit lower than the time cost of CP-ABE.

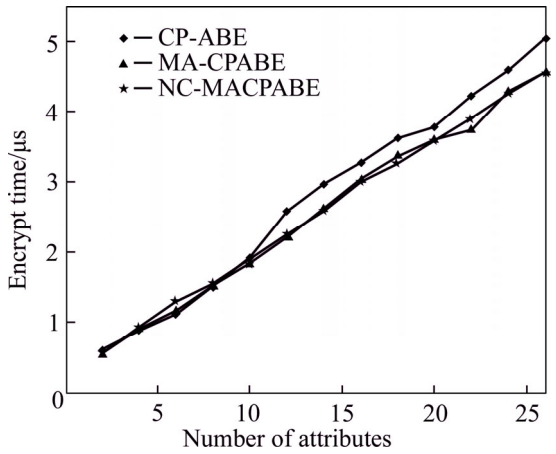

Fig. 7 Symmetric key encryption time of NC-MACPABE, MPRE-CPABE and CP-ABE

The performance of the symmetric key decryption also has nothing to do with the number of attribute authority. Figure 8 shows the symmetric key decryption times of NC-MACPABE, MPRE-CPABE and CP-ABE with different numbers of attribute. The symmetric key decryption times of all NC-MACPABE, MPRE- CPABE and CP-ABE increase with the number of attribute increasing. The time cost of NC-MACPABE is a little bit higher than the time costs of MPRE-CPABE and CP-ABE. The reason is that without the central authority, NC-MACPABE needs to combine the attribute private key components received from attribute authorities to get the private key, which is relatively complicated.

Finally, we compared the access right revocation times of NC-MACPABE and MPRE-CPABE with and without proxy re-encryption. The number of attribute authority is set 5. Figure 9 shows the experimental results. As shown in Fig. 9, the proxy reencryption can greatly reduce the time costs of the access right revocation. Furthermore, the time cost of NC-MACPABE is a little higher than the time cost of MPRE-CPABE. However, the time cost variability of NC-MACPABE is more stable than MPRE-CPABE.

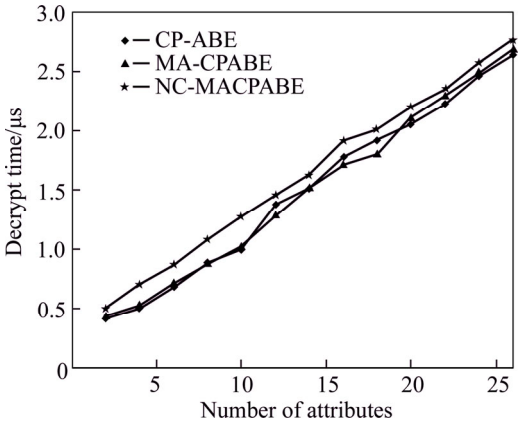

Fig. 8 Symmetric key decryption time of NC-MACPABE, MPRE-CPABE and CP-ABE

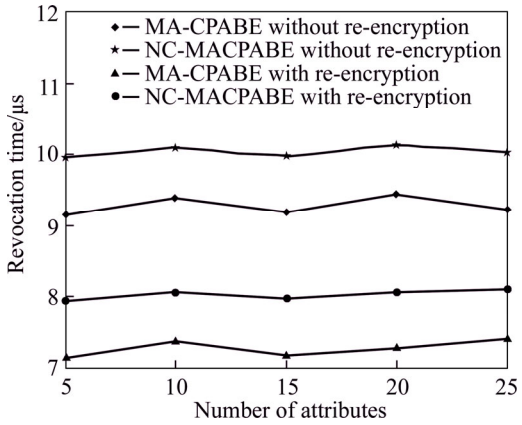

Fig. 9 Access right revocation time of NC-MACPABE and MPRE-CPABE

7 Conclusions

Cloud storage extends the concept of cloud computing, which has been rapidly used in various fields. The character of data management and ownership separation, which means the cloud storage service provider is not completely trusted, leads to the difficulties of data security and privacy protection. According to the data privacy based on data partition and classification, this work proposes NC-MACPABE, a novel multi-authority access control scheme without a central authority based on CP-ABE. An optimization of WAS allows different levels of operation for the same data in cloud storage system. The concept of identity dyeing is introduced to improve the users' information privacy further. The re-encryption algorithm is improved in the scheme so that the data owner can revoke user's access in a more flexible way. The scheme is proved to be secure, efficient and suitable for public cloud storage systems. However, the time cost of NC-MACPABE is a little higher than the time cost of MPRE-CPABE during symmetric key decryption and access right revocation, which is our focus of future research.

References

[1] XU Xiao-long, TU Qun, NIK B, YANG Geng, WANG Xin-heng. SATVPC: Secure-agent-based trustworthy virtual private cloud model in open computing environments [J]. Journal of Central South University, 2014, 21(8): 3186−3196.

[2] ZHANG Jiang, ZHANG Zhen-feng. Secure and efficient data-sharing in clouds [J]. Concurrency and Computation: Practice and Experience, 2015, 27(8): 2125−2143.

[3] FENG Deng-guo, ZHANG Min, ZHANG Yan, XU Zhen. Study on cloud computing security [J]. Journal of Software, 2011, 22(1): 71−83. (in Chinese)

[4] ZOU De-qing, JIN hai, QIANG Wei-zhong, XU Peng. Challenge and practice of cloud computing Security [J]. Communications of the China Computer Federation, 2011, 7(12): 55−61. (in Chinese)

[5] YU Neng-hai, HAO Zhuo, XU Jia-jia. Review of cloud computing security [J]. Chinese Journal of Electronics, 2013, 41(2): 371−381. (in Chinese)

[6] HU Fei, QIU Mei-kang, LI Jia-yin, GRANT T, TAYLOR D, MCCALEB S, BUTLER L, HAMNER R. A review on cloud computing: Design challenges in architecture and security [J]. Journal of Computing and Information Technology, 2011, 19(1): 25–55.

[7] RISHI I. Apple to Strengthen Security After iCloud Nude Celebrity Photos Leak [EB/OL]. [2014−09−02]. http://time.com/3271667/ apple-jennifer- lawrence-icloud-leak-security.

[8] Cloud Security Alliance. Security guidance for critical areas of focus in cloud computing v2.1 [EB/OL]. [2013−03−26]. http://www. cloudsecurityalliance. org/guidance/csaguide.v2.1.pdf.

[9] SU Jin-shu, CAO Dan, WANG Xiao-feng, SUN Yi-pin, HU Qiao-lin. Attribute-based encryption schemes [J]. Journal of Software, 2011, 22(6): 1299−1315. (in Chinese)

[10] XUE Wei, SHU Ji-wu, LIU Yang, XUE Mao. Corslet: A shared storage system keeping your data private [J]. Science China Information Sciences, 2011, 54(6): 1119−1128.

[11] MAHAJAN P, SETTY S, LEE S, CLEMENT A, ALVISI L, DAHLIN M, WALFISH M. Depot: Cloud storage with minimal trust [J]. ACM Transactions on Computer Systems, 2011, 29(4): 1−38.

[12] SHRAER A, CACHIN C, CIDON A, KEIDAR I, MICHALEVSKY Y, SHAKET D. Venus: Verification for untrusted cloud storage [C]// Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop. New York: ACM, 2010: 19−30.

[13] ROY I, SETTY S T V, KILZER A, SHMATIKOV V, WITCHEL E. Airavat: Security and privacy for mapreduce [C]// Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation. San Jose: DBLP, 2010, 10: 297−312.

[14] YAN Liang, RONG Chun-ming, ZHAO Gan-sen. Strengthen cloud computing security with federal identity management using hierarchical identity-based cryptography [C]// Proceedings of 1st International Conference. Berlin: Springer, 2009: 167−177.

[15] WANG Qian, WANG Cong, LI Jin, REN Kui, LOU Wen-jing. Enabling public verifiability and data dynamics for storage security in cloud computing [C]// Proceedings of Computer Security–ESORICS 2009. Berlin: Springer, 2009: 355−370.

[16] DAMIANI E, PAGANO F, PAGANO D. iPrivacy: A distributed approach to privacy on the cloud [J]. International Journal on Advances in Security , 2011, 4(3): 185−197.

[17] MAO Jian, LI Kun, XU Xian-dong. Privacy protection scheme for cloud computing [J]. Journal of Tsinghua University(Science and Technology), 2011, 51(10): 1357−1362. (in Chinese)

[18] HUANG Ru-wei, GUI Xiao-lin, YU Si, YU Wei. Privacy-preserving computable encryption scheme of cloud computing [J]. Chinese Journal of Computers, 2011, 34(12): 2391−2402. (in Chinese)

[19] SHAMIR A. Identity-based cryptosystems and signature schemes [J]. Lecture Notes in Computer Science, 1984, 21(2): 47−53.

[20] BONEH D, FRANKLIN M. Identity-based encryption from the Weil pairing [C]// Proceedings of the 10th Cryptology Conference on Advances in Cryptology. Berlin: Springer, 2001: 213−229.

[21] SAHAI A, WATERS B. Fuzzy identity-based encryption [C]// Proceedings of the 2005 Annual Eurocrypt Conference. Berlin: Springer, 2005: 457−473.

[22] GOYAL V, PANDEY O, SAHAI A, WATERS B. Attribute-based encryption for fine-grained access control of encrypted data [C]// Proceedings of the 13th ACM Conference on Computer and Communications Security. New York: ACM, 2006: 89−98.

[23] BETHENCOURT J, SAHAI A, WATERS B. Ciphertext-policy attribute-based encryption [C]// Proceedings of the 2007 IEEE Symposium on Security and Privacy. Washington: IEEE Computer Society, 2007: 321−334.

[24] CHASE M. Multi-authority attribute based encryption [C]// Proceedings of Theory of Cryptography Conference. Berlin: Springer, 2007: 515−534.

[25] BOZOVIC V, SOCEK D, STEINWANDT R, VILLANYI V I. Multi-authority attribute-based encryption with honest-but-curious central authority [J]. International Journal of Computer Mathematics, 2012, 89(3): 268−283.

[26] LIN Huang, CAO Zhen-fu, LIANG Xiao-hui, SHAO Jun. Secure threshold multi authority attribute based encryption without a central authority [J]. Information Sciences, 2010, 180(13): 2618−2632.

[27] BLAZE M, BLEUMER G, STRAUSS M. Divertible protocols and atomic proxy cryptography [C]// Proceedings of the 1998 International Conference on the Theory and Application of Cryptographic Techniques Espoo. Berlin: Springer, 1998: 127−144.

[28] ATENIESE G, FU K, GREEN M, HOHENBERGER S. Improved proxy re-encryption schemes with applications to secure distributed storage [J]. ACM Transactions on Information and System Security, 2006, 9(1): 1−30

(Edited by FANG Jing-hua)

Cite this article as: XU Xiao-long, ZHANG Qi-tong, ZHOU Jing-lan. NC-MACPABE: Non-centered multi-authority proxy re-encryption based on CP-ABE for cloud storage systems [J]. Journal of Central South University, 2017, 24(4): 807−818. DOI: 10.1007/s11771-017-3483-z.