IEICE TRANS. INF. & SYST., VOL.E95–D, NO.12 DECEMBER 2012

LETTER

Keys Distributing Optimization of CP-ABE Based Access Control in Cryptographic Cloud Storage*

Yong CHENG[†a)], ***Student Member***, **Jiangchun REN**[†], **Zhiying WANG**[†], **Songzhu MEI**[†], ***and*** **Jie ZHOU**[†], ***Nonmembers***

SUMMARY In this letter, we introduce a novel keys distribution optimization scheme for CP-ABE based access control. This scheme integrates roles, role hierarchies and objects grouping to accelerate keys distribution, meanwhile the CP-ABE encrypting overhead is reduced by adopting deterministic cryptographic function. Experiments show that our scheme obtains noticeable improvement over the original one, especially when the number of objects is much greater than that of users.

key words: *keys distribution, access control, cryptographic cloud storage*

1. Introduction

Nowadays, public cloud storage is rapidly developing as an emerging storage outsourcing paradigm. More and more customers begin to adopt the cloud storage for online data storing and sharing. However, the Cloud Storage service Provider (CSP) is "untrusted" and may leak users' data. So users have to build an additional access control system to enforce authorizations and protect their sensitive data.

S. Kamara and K. Lauter [1] introduced cryptographic cloud storage as a solution for both access control enforcement and data confidentiality protection. The cryptographic cloud storage is built on a public cloud infrastructure and its access control is achieved as follows. 1) the Data Owner (DO), encrypts file F with randomly chosen symmetric key k, and then sends the ciphertext $E_k(F)$ to remote servers; 2) when a Data User (DU) wants to access F, he/she will first send a request to DO, then DO will reply the k and access credential via a secure channel; 3) DU retrieves $E_k(F)$ by the credential and decrypts it using k. This method can achieve strong access control but it does not specify the method of distributing symmetric keys.

The Ciphertext-Policy Attribute Based Encryption (CP-ABE) [2] is a new promising encryption algorithm for keys delivering in cryptographic cloud storage. Based on CP-ABE, many fine-grained access control models have been proposed by researchers for online data sharing, such as HABE [3]. The core of these models is encrypting the symmetric keys with a certain access tree, and then publishing it with the ciphertext. A DU can obtain the symmetric key if and only if its attributes set satisfies the access tree.

The details of CP-ABE based access control in cryptographic cloud storage is illustrated in Fig. 1: 1) The system starts up via running the Setup algorithm and generating the public parameters PK and a master key MK; 2) A keys generator runs $KeyGen$ for generating each DU's private key SK, and then sending SK to DU via a secure channel; 3) DO encrypts a message M with the access tree T. The ciphertext CT is published to cloud storage; 4) A DU can decrypt CT if and only if its attributes set Au satisfies T. In this model, the keys generator is an independent third party. The details of CP-ABE algorithm can be find in [2].

However, the key distribution of the CP-ABE based scheme is not efficient enough. First, as a type of asymmetric cryptographic algorithms, CP-ABE consumes a lot of resources which thin DO cannot afford. DO needs to generate a unique symmetric key and encrypt it for each file. Thus DO may become the bottleneck when a huge-scale of files need to be shared online. Second, the time consumption of a single CP-ABE encrypting operation may be too large. The runtime needed by the CP-ABE encrypting is precisely linear with the number of leaf nodes in the access tree, and the size of the access tree may grow enormously when a lot of DUs are authorized. What's more, the symmetric key is updated periodically in some enterprise applications, so the key distribution overhead may consume too much time.

Our main goal is to speed up the key distributing mechanism of CP-ABE based access control. At a high level, our work is similar to recent papers [4] and [5]. [4] presented how to optimize role-based access control model by using role attributes; [5] built an access control model and made access decisions based on each role's attributes. In our scheme, roles are employed to group users and substan-

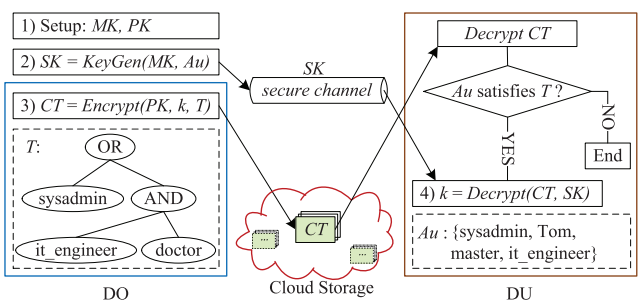

Fig. 1 The CP-ABE based access control model.

Manuscript received April 5, 2012.
Manuscript revised July 29, 2012.

[†]The authors are with the School of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China.

*This work was supported in part by the National Natural Science Foundation of China under grant No. 60903204.

a) E-mail: ycheng@nudt.edu.cn

DOI: 10.1587/transinf.E95.D.3088

tially new techniques are still needed. Approaches related to our scheme are [6] and [7]. [6] proposed a large-scale content delivering system whose performance was improved by dividing the user population into groups. In [7], selective encryption was raised for enforcing access control in data outsourcing scenario. In this scheme, each object (files or other data) is encrypted with a symmetric key and each authorized user can derive this key via his/her private key and public tokens.

In this letter, we propose a new approach named Role Based Keys Distribution (RBKD) which integrates several optimization techniques. First, RBKD combines Role Based Access Control [8] and objects grouping. This method is practical because in an actual system, users are always described by roles and objects are usually authorized to roles directly. Second, RBKD accelerates keys distribution by using deterministic cryptographic function and role hierarchies. Instead of allocating a random key for each object, RBKD only generates a key for each role. The size of roles is usually less than that of objects, thus using role hierarchies can minimize the amount of symmetric keys. It is noteworthy that RBKD only optimizes the existing methods instead of proposing a new cryptographic system. To keep the original scheme's security, we are planning to adopt grant/revoke procedure for maintaining the correctness of the role hierarchy.

2. RBKD Scheme

2.1 Basic Concepts and Notation

Given a set U of DUs and a set O of objects, we introduce the basic definitions as follows.

Definition 1 (authorization policy [7]): An authorization policy over U and O, denoted by A, is a triple $<U, O, P>$, where $u \in U$ and $o \in O$, and P is a set of permissions in the form $<u, o>$, stating that u can access o.

Definition 2 (authorization policy matrix): An authorization policy matrix over $A = <U, O, P>$, denoted by M_A, is a matrix $(a_{ij})_{m\times n}$, where m is the size of U and n is the size of O, $a_{ij} = 1$ iff $<u_i, o_j> \in P$, otherwise $a_{ij} = 0$.

Definition 3 (role): A role over U, denoted by R, is a set satisfies $R \subseteq 2^U \setminus \{\emptyset\}$. And a role R is granted to access an object $o \in O$ iff for each $u \in R$, $<u, o> \in P$.

Definition 4 (role hierarchy): A role hierarchy over R_i and R_j, denoted by H, where $R_i \in R_j.H$ iff $R_i \subset R_j$.

Let A_{all} denote the collection of all attributes in the system. And the user u_i's attributes set is represented as Au_i, $Au_i \subseteq A_{all}$. We assume each u_i has a unique attribute, denoted by $userID$, then the role R's attributes set can be described as $R.attrs = \{u_i.userID \mid u_i \in R\}$. In the CP-ABE based access control model, all objects will be encrypted by the symmetric encryption algorithm (e.g. AES) first, and then the symmetric key is encrypted by CP-ABE algorithm. Let K denote a set of random symmetric keys and a key $k_i \in K$ is used to encrypt an object o_i. The CP-ABE encrypting procedure can be expressed as $Encrypt(PK, k_i, T_i)$, where PK is the public parameters and T_i is the corresponding access tree. Let R_i denote the role granted to access object o_i, the access tree T_i can be expressed as $T_i = [\mathsf{OR} userID \mid userID \in R_i.attrs]$.

Before presenting the details of the proposed scheme, we introduce an example to simplify the description. Figure 2 illustrates the example of an authorization policy matrix with 6 users, 9 objects and 30 permissions.

2.2 The RBKD Scheme

The RBKD scheme is divided into two phases: roles creating and keys distributing. Considering a DU set U with the size of m, the amount of all possible roles is $2^m - 1$. So in the first phase, we only create the necessary roles and use role hierarchies to accelerate keys distributing. The details of roles creating algorithm is given as Algorithm 1.

Considering the example in Fig. 2, we create 7 roles,

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9
u_1	1	0	1	1	1	1	1	0	1
u_2	0	1	0	1	1	1	1	0	1
u_3	0	1	0	1	1	0	1	0	1
u_4	0	0	0	0	1	1	0	1	1
u_5	0	1	0	0	1	1	0	1	0
u_6	0	1	0	0	1	1	0	1	0

Fig. 2 An example of authorization policy matrix.

Input: authorization policy matrix $(a_{ij})_{m\times n}$
Output: roles chain $Chain_R$

1 $Chain_R \leftarrow \emptyset$;
2 $cnt_j \leftarrow \sum_{i=1}^{m} a_{ij}$;
3 sort all $o_j \in O$ by cnt_j in ascending order;
4 **for** $j \leftarrow 1; j \le n; j \leftarrow j + 1$ **do**
5 $R_{new} \leftarrow$ create a role corresponding to o_j ;
6 **if** $R_{new} \notin Chain_R$ **then**
7 $R_{new}.attrs \leftarrow \{u_i.userID \mid u_i \in R_{new}\}$;
8 **for** *each role* $R_* \in Chain_R$ **do**
9 **if** $R_* \subset R_{new}$ **then**
10 delete all $u.userID$ from $R_{new}.attrs$ if $u \in R_*$;
11 $R_{new}.H \leftarrow R_{new}.H \cup R_*$
12 **end**
13 **if** $R_{new}.attrs = \emptyset$ **then**
14 break;
15 **end**
16 **end**
17 for every $R_p \in R_{new}.H$, if $\exists R_q \in R_{new}.H$ satisfies $R_p \subset R_q$, delete R_p from $R_{new}.H$;
18 $PUSH(Chain_R, R_{new})$;
19 **end**
20 **end**
21 return $Chain_R$;

Algorithm 1: roles creating.

Table 1 Details of roles generated for the example in Fig. 2.

role	user	object	R.attrs	R.H
R_1	u_1	$o_{1,3}$	$u_1.userID$	ø
R_2	$u_{1,2,3}$	$o_{4,7}$	$u_{2,3}.userID$	R_1
R_3	$u_{4,5,6}$	o_8	$u_{4,5,6}.userID$	ø
R_4	$u_{2,3,5,6}$	o_2	$u_{2,3,5,6}.userID$	ø
R_5	$u_{1,2,3,4}$	o_9	$u_4.userID$	R_2
R_6	$u_{1,2,4,5,6}$	o_6	$u_2.userID$	$R_{1,3}$
R_7	$u_{1,2,3,4,5,6}$	o_5	ø	$R_{2,3}$

Input: roles chain $Chain_R$
Output: the token table $Table_{token}$
1 $Table_{token} := \emptyset$;
2 generate a random key k_i for each role R_i;
3 **while** $Chain_R \neq \emptyset$ **do**
4 $R_i \leftarrow POP(Chain_R)$;
5 add a row_i with label R_i in $Table_{token}$;
6 **if** $R_i.attr \neq \emptyset$ **then**
7 $owner \leftarrow \{u \mid u.userID \in R_i.attr\}$;
8 $T_i = [\mathsf{OR} userID \mid userID \in R_i.attrs]$;
9 $cip \leftarrow Encrypt(PK, k_i, T_i)$;
10 add $<owner, cip>$ to row_i in $Table_{token}$;
11 **end**
12 **for** *each role* $R_j \in R_i.H$ **do**
13 $owner \leftarrow \{R_j\}$;
14 $cip \leftarrow h(k_j) \oplus k_i$;
15 add $<owner, cip>$ to row_i in $Table_{token}$;
16 **end**
17 **end**
18 return $Table_{token}$;

Algorithm 2: keys distributing.

and each role is granted to access corresponding objects. The details of roles, including mappings to users and objects, are shown in Table 1.

In the second phase, a random key is assigned to each role for encrypting corresponding objects. Take R_1 in Table 1 as an example, we generate a random symmetric key k_1 for encrypting o_1 and o_3. Now the question to be asked is "how to distribute the symmetric key to authorized users securely and efficiently?"

We build a token table for distributing keys in RBKD scheme. A token is the ciphertext of the key presented in the form of $<owner, cip>$, meaning that the *owner* (a user or a role) can decrypt the *cip* for retrieving the key. Each role's symmetric key is encrypted to form some tokens, and tokens generating procedure is described in Algorithm 2. In this procedure we adopt a deterministic cryptographic function (e.g. MD5 and SHA), denoted by h, for keys encrypting.

The token table of the example in Fig. 2 is presented in Table 2. Authorized users can derive the symmetric key by downloading corresponding tokens from the table. For instance, if u_1 wants to read o_6, he/she will first obtain $<R_1, h(k_1) \oplus k_6>$ since $u_1 \in R_1$, and then retrieves $<u_1, Encrypt(PK, k_1, [u_1.userID])>$ recursively, finally k_6 can be computed by decrypting the two tokens.

In summary, we accelerate the keys distributing by the following methods. First, we try to diminish CP-ABE encrypting operations. As shown in Table 2, the amount of CP-ABE operations is 6, which is less than the original one (which is 9). Second, we attempt to downsize the access tree in RBKD scheme. For example, there is only one leaf node in $Encrypt(PK, k_6, [u_2.userID])$, while the original scheme will use at least 5 leaf nodes for the same purpose. Compared to the primitive one, the RBKD scheme introduces more deterministic encrypting and bitwise *xor* operations, however, it is still of high efficiency because these procedures consume only a little time.

Table 2 The token table corresponding to the example in Fig. 2.

object	role	token
$o_{1,3}$	R_1	$<u_1, Encrypt(PK, k_1, [u_1.userID])>$
$o_{4,7}$	R_2	$<u_{2,3}, Encrypt(PK, k_2, [\mathsf{OR} u_{2,3}.userID])>$,
		$<R_1, h(k_1) \oplus k_2>$
o_8	R_3	$<u_{4,5,6}, Encrypt(PK, k_3, [\mathsf{OR} u_{4,5,6}.userID])>$
o_2	R_4	$<u_{2,3,5,6}, Encrypt(PK, k_4, [\mathsf{OR} u_{2,3,5,6}.userID])>$
o_9	R_5	$<u_4, Encrypt(PK, k_5, [u_4.userID])>$,
		$<R_2, h(k_2) \oplus k_5>$
o_6	R_6	$<u_2, Encrypt(PK, k_6, [u_2.userID])>$,
		$<R_1, h(k_1) \oplus k_6>$,
		$<R_2, h(k_3) \oplus k_6>$
o_5	R_7	$<R_2, h(k_2) \oplus k_7>$,
		$<R_3, h(k_3) \oplus k_7>$

Grant and Revoke In the RBKD scheme, we have built a role hierarchy according to an authorization policy matrix. Once the access policy is changed, we need to update the hierarchy via a grant and revoke procedure. Consider a grant/revoke request for a user u on an object o. First the DO will decrypt and re-encrypt the o with a new key k', then the token table will be updated for distributing the k' to the new authorized role. Using the grant/revoke procedure, our scheme can prevent malicious user from decrypting the unauthorized ciphertext. We will omit the details of grant and revoke due to space limitations.

3. Experimental Results

To evaluate the effectiveness and scalability of the RBKD scheme, a significant test extracted from a complex authorization policy of a large system is required. Unfortunately, there is no such an access control benchmark available today. As a result, simulated scenarios are usually created in related researches [6], [9] for experiments. In this section, we also build two simulated scenarios to compare RBKD scheme with the original one.

The first scenario is based on DBLP [10] bibliography, a well known bibliographic database which currently lists more than 1.9 million publications. Each article recorded in DBLP includes title, authors, publisher and other elements. We suppose that all articles need to be stored in cryptographic cloud storage, and each article must be accessible to its all authors. We simplify the scenario by assuming that a user u_i can access an article o_j iff u_i is one of the authors of o_j.

Our prototype is a C++ program which collects 30000 authors to form a set $\mathbb{U}$, and the corresponding article set

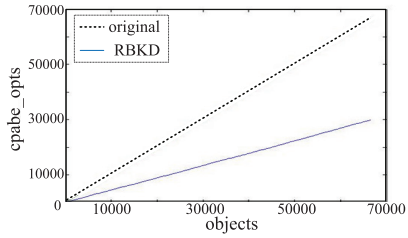

Fig. 3 The comparison of the amount of *cpabe_opts*.

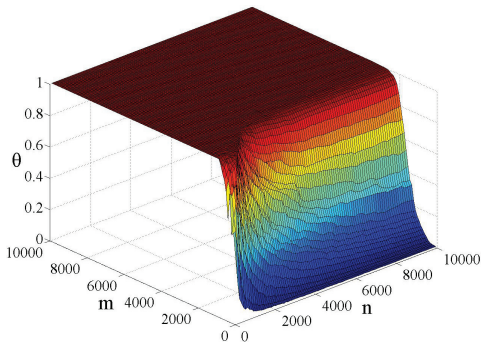

Fig. 4 The value of θ corresponding to different m and n.

$\mathbb{O}$'s size is 66854. We build the RBKD scheme over $\mathbb{U}$ and $\mathbb{O}$ with the access policy described above, and then compare our scheme with the original one. Because the main cost of keys distributing is CP-ABE encrypting operations (we call it *cpabe_opts*), we only consider the amount of *cpabe_opts* in our comparison. In the original scheme, we need to encrypt a symmetric key for every object. But the RBKD scheme only performs *cpabe_opts* for roles to reduce the number of operations significantly. Figure 3 shows the comparison of the amount of *cpabe_opts* between the RBKD scheme and the original one. The experimental result shows that the amount of CP-ABE encrypting operations in the new scheme is about 44.9% of the original one.

In order to evaluate the RBKD scheme's performance varying the size of U and O, we design the second simulated scenario by the following steps: First, we randomly choose $m, n \in \mathbb{Z}^+$, where m is the size of U and n is the size of O; Second, for each element in matrix $(a_{ij})_{m\times n}$, we set $a_{ij} = 1$ with a certain probability, denoted as p; Third, we perform the RBKD scheme over U and O with the authorization policy matrix $(a_{ij})_{m\times n}$.

Let *rbkd_cnt* represent the amount of *cpabe_opts* in our scheme, we compute the ratio $\theta = \frac{rbkd\_cnt}{n}$ for evaluating the optimization rate of the RBKD scheme over the original one. Let $m, n \in [1, 10000]$ and $p = 0.05$, for each (m, n) pair we repeat the simulation 10 times and compute the average θ. Figure 4 illustrates how the value of θ changes with different m and n and Table 3 lists out partly numerical results. The experimental result shows that the optimization rate θ can be reduced by increasing the ratio n/m.

Discussion Generally speaking, the RBKD scheme achieves better performance when n is greater than m. This is because that under this condition, the amount of role hierarchies included in keys distributing will increase, and the probability of different objects authorized to the same role will be larger. Actually, the kind of situation where $n \gg m$ is very common in enterprise applications and the data users is usually organized in hierarchies, implying that our scheme is well suitable for data sharing in enterprise.

Table 3 Partly numerical results in the second simulated scenario.

m	8797	231	128	242	184	59	5
n	6482	4240	3065	8786	6902	3002	8842
n/m	0.737	18.35	23.95	36.31	37.51	50.88	1768.4
θ	1.000	0.999	0.995	0.634	0.303	0.079	0.0005

4. Conclusions

There is an emerging trend that more and more customers are using public cloud storage for online data storing and sharing. In order to prevent data leakage when the CSP is not trusted, adopting CP-ABE based access control is taken as a common solution. Unfortunately, key distribution of the original scheme is inefficient; our scheme optimizes it by taking advantage of role hierarchies and deterministic cryptographic function. Experiments showed that the RBKD scheme can achieve noticeable speedup over the original one especially in enterprise applications.

References

[1] S. Kamara and K. Lauter, "Cryptographic cloud storage," Proc. 14th International Conference on Financial Cryptograpy and Data Security, pp.136–149, Tenerife, Canary Islands, Spain, 2010.
[2] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," 2007 IEEE Symposium on Security and Privacy, pp.321–334, Oakland, USA, 2007.
[3] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," Proc. 17th ACM Conference on Computer and Communications Security, pp.735–737, Chicago, USA, 2010.
[4] J. Huang, D. Nicol, R. Bobba, and J. Huh, "A framework integrating attribute-based policies into role-based access control," Proc. 17th ACM Symposium on Access Control Models and Technologies, pp.187–196, 2012.
[5] B. Cha, J. Seo, and J. Kim, "Design of attribute-based access control in cloud computing environment," Proc. International Conference on IT Convergence and Security 2011, pp.41–50, 2012.
[6] P. Traynor, K. Butler, W. Enck, and P. Mcdaniel, "Realizing massive-scale conditional access systems through attribute-based cryptosystems," Proc. 16th Annual Network and Distributed System Security Symposium, 2008.
[7] S. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Encryption policies for regulating access to outsourced data," ACM Trans. Database Systems, vol.35, no.2, pp.1–45, 2010.
[8] D. Ferraiolo, J. Cugini, and R. Kuhn, "Role-based access control (RBAC): Features and motivations," Proc. 11th Annual Computer Security Application Conference, pp.241–248, 1995.
[9] W. Wang, Z. Li, R. Owens, and B. Bhargava, "Secure and efficient access to outsourced data," Proc. 2009 ACM Workshop on Cloud Computing Security, pp.55–66, 2009.
[10] D. Team, "The DBLP computer science bibliography," http://dblp.uni-trier.de/db/index.html, 2012.