Adopting Multi-mode Access Control for Secure Data Sharing in Cloud

Chunhua Li[✉], Ronglei Wei, Zebang Wu, Ke Zhou, Cheng Lei, and Hao Jin

Wuhan National Lab for Optoelectronics, Huazhong University of Science and Technology, Wuhan 430074, China
{li.chunhua,hust_wrl}@hust.edu.cn

Abstract. Cloud data sharing introduces a new challenge to the enforcement of security controls. The existing approaches are not flexible and low efficiency while performing access control. In this paper, we propose a multi-mode access control scheme, which can support multiple access strategies for data distributed at different areas in cloud. Meanwhile, we introduce the concept of dynamic attribute into the access policy to adjust user's access privileges timely according to his changeable characteristics. Specifically, we present an efficient revocation method which uses confusion token to process the ciphertext at the server. We apply these techniques to design a muti-mode access control system and implement the prototype based on the Openstack platform. Furthermore, we devise a Uniform Access Control Markup Language (UACML) based on XACML, which greatly improves the expressiveness of our multi-mode access control policies. The experimental results show that our scheme has low computational overhead for revocation as well as good flexibility.

Keywords: Cloud · Access control · Dynamic attribute · Revocation

1 Introduction

Data sharing is becoming an important application of cloud storage service [1, 2], which offers service for data owners to conveniently share their data with other users within the cloud. Many cloud service providers, such as Dropbox, Amazon and Aliyun, provide this kind of service. However, this new paradigm of data sharing and access service introduces a great challenge to data access control. First, the means of shared data in cloud is usually diverse, different ways of data sharing should adopt different access control policies. Second, a user may be added into or withdrawn from the cloud frequently, so his access permissions to data should be changed accordingly, thus yielding great performance overhead.

According to ways of data sharing, the space of storage in cloud servers can be logically divided into three categories: private area, group area and public area. In private area, only a data owner has rights to access his/her data at any geographical location or on any terminals, while other users are not allowed to access private data. In group area, users in the same group usually have many common characteristics like interests, research field and educational background, so every user in the same group has identical

DOI: 10.1007/978-3-319-27137-8_39

permissions to data shared by group users, while other users cannot access these data. Thus all the shared resources can be downloaded by group users with no strict limitation. This access control scheme for group is popular adopted by current cloud storage systems. Let's consider a practical scenario: an assistant wants to share a notice with all the party members and class leaders who are only part of users in the grade group. It is obviously that using the same access privilege for all the members in the group cannot meet such kinds of fine-gained access requirement. In public area, no effective security solution is taken in the existing clouds, all users have free access to public resources as long as they log into the system legally, which is obviously not conducive to encouraging users to share valuable resources. What' more, some users may want to dominate the usage of their data shared in the public environment. For example, Bob hopes that his contributed resources only be read by those who frequently access the similar resources. To meet the demand of such kind of situation, we need to capture the user's access behavior, and then to adjust his access privilege dynamically. Therefore, the dynamic characteristics of user should be taken into account in order to encourage them to share more interesting resources. Similarly, the access to group resources should also be treated discriminatorily for different members in the same group.

Three types of access control system are commonly suggested for cloud environment, that is, Role Based Access Control (RBAC), Attribute Based Access Control (ABAC), Attribute and Role Based Access Control (ARBAC). Sirisha and G. Kumari have proposed RBAC at API level in cloud [3], which follows the least privilege principle by assigning rights according to role specification and user attributes. Data can be accessed by users who have matching roles. RBAC is mostly used for commercial organizations and enterprises. In RBAC, roles are defined in a static manner and cannot be modified dynamically according to the change of organization security requirements. Privacy aware access control system (ARBAC) is proposed for cloud which is composed of two models: RBAC and ABAC [4]. In ARBAC, user needs to provide corresponding subject, resource and environment attributes that are required for the service when he requests to access data. Cloud service provider verifies the given attributes according to defined privacy policy. User and data classification levels are defined according to which privacy preferences and access policies are formulated. Hence complexity of defining policies becomes high with the increase of user classification levels.

To provide fine grained access control in cloud, attribute based encryption (ABE) has been suggested in [5–7]. There are two kinds of ABE: key-policy ABE (KP-ABE) and cipher-text policy ABE (CP-ABE). In KP-ABE, access policy is defined in private key which is assigned to users and can decrypt only those files whose attributes match with this policy. On the other hand, in CP-ABE, access policy is given in cipher text with each file and each user is issued a secret key associated with its attributes, where access structure is defined over attributes assigned to each file. Only users that his attributes' list matches with the structure can access the data. This system requires to define access structure for each user which may become complex because of their varying access requirements. Data owner defines the threshold value in policy specification that represents the number of attributes to be matched for each user request. Therefore, ABE systems introduce large overhead in terms of mathematical operations and algorithms which affect performance of enforcement mechanism. Although CP-ABE is regarded as one of the most desirable technologies for data access control in cloud environment, existing CP-ABE schemes [8–10] are still inefficient for coping with the privilege revocation and inflexible for expression of access control policies.

In this paper, we address aforementioned issues and present a multi-mode access control scheme which combines the advantages of IBAC (Identity-Based Access Control), RBAC and ABAC. Meanwhile, we extend the connotation of attribute by introducing the dynamic characteristic into the ABAC to fulfill the dynamical access control. Besides, an efficient revocation method is proposed to solve the privilege revocation problem in the system. We apply our proposed techniques to construct a prototype of multi-mode access control system based on the Openstack platform, and express the corresponding polices with our defined uniform access control markup language.

2 Notations and Concepts

In this section we mainly define some notations involved in this paper.

Definition 1. In this paper, **attribute** is specified as the user's characteristics. It is represented with a two-tuple consisting of attribute name and its value, and categorized as two kinds of types: static attribute and dynamic attribute.

Static attribute, refers to the attributes whose value are relatively fixed or rarely changed within a period of time. For instance, user's identification number is generally unchangeable, user' email address is usually stable, and user's name rarely alters, so these attributes can be regarded as static attributes. A static attribute set can be represented as

$$S_{static\_attr} = \{attr_{name}{:}attr_{value}\},$$

here, $attr_{name}$ is a set of user's stable characteristics, $attr_{value}$ is the corresponding value, they are one-to-one relationship.

Dynamic attribute, refers to the attribute whose value often changes or cannot be quantified within a certain time range. For example, it is difficult to quantify the user's skill level, which varies along with user's experience. Other similar characteristics like user's activity level and location information are often considered as dynamic attributes. In some situations, dynamic attribute cannot be ignored. For instance, location information is a very important decision factor in the system of geographic information service. A dynamic attribute set can be represented as:

$$S_{dynamic\_attr} = \{attr_{name}{:}attr_{value1}, attr_{value2}, \ldots\},$$

here, $attr_{name}$ is a set of user's variable characteristics, $attr_{value1}$, $attr_{value2}$, …, is the possible value, they are one-to-many relationship.

We can set the user's attribute according to the specific requirements. Supposed the following actual scenario.

A mall is offering online discount, getting coupons must satisfy the following rules: (1) super member of the mall (2) at the specified areas (3) online user registered with his phone number.

From above description, we can extract the following set of user attributes:

$$\{memberlevel,\ geographic\ information,\ phone\ number\}$$

Here, *phone number* can be regarded as a static attribute because its value doesn't change within a period of time, *geographic information* is a distinct dynamic attribute, *member level* is a non-quantitative feature which corresponds to consumption range, and it will be adjusted with user's consumption.

Therefore, it is obviously inappropriate that only static attribute is considered while formulating access control policy.

Definition 2. **UACML** (Uniform Access Control Markup Language), a kind of access control policy description language, is designed to uniformly describe different kinds of access control policies such as identity-based access control (IBAC), static attribute-based access control (ABAC) and dynamic attribute-based access control (DABAC). It can express and enforce complex access control policy in a simple and flexible way. Table 1 gives a concrete instance of policy description.

Table 1. An example of access policy description

```
<DACML>
<method value="DABAC" /> <!-- one of in [ IBAC | ABAC | DABAC | NONE] -->
<white list="Bob, Jim, Lucy" />
<black list="Lily, Sally" />
<rule>
    <item name="item1" attr="programming language" value="one in {android, object-c,
c#}" weight=1/>
    <item name="item2" attr="job" value="client developer" weight =1 />
    <item name="item3" attr="skill" value="junior" weight =1 />
    <item name="item4" attr="work years" value=">3" weight =2 />
    <item name="item5" attr="os" value="linux" weight =1 />
    <item name="item6" attr="job" value="server developer" weight =1 />
</rule>
<policy
    value="(weight >2=in {item1,item2,item3})or(weight>2=in{item4,item5,item6})">
</policy>
</DACML>
```

The meanings of labels in Table 1 are described below.

method, used to set the mode of access control, it can be one of the *IBAC*, *ABAC*, *DABAC* and *NONE*. If method is set to *NONE, it will* mean that the resource is open for all users.

white, used to set users in the whitelist. Users in the whitelist has priority access to the data, even if his attributes do not satisfy the access policy.

black, used to set users in the blacklist. Users in the blacklist cannot access the data, no matter whether his attributes meet the access policy.

rule, used to describe all kinds of atom-unit of access policy. Each atom-unit is expressed with *item* label which indicates attribute's name and value.

policy, used to describe the logical expression of access policy. If *method* is set to DABAC, *item* label within *rule* label will represent the dynamic attribute, at this moment it will not be a simple 'yes' or 'no' decision while matching the logical expressions (detailed decision processes will be shown in Sect. 3.2).

Definition 3. **Auth_Token**, a kind of data structure for privilege decision. It is generated by system and returned to the user after a user logs into the system legally. **Auth_Token** has a life cycle, it includes two kinds of types: Basic_Auth_Token and Extend_Auth_Token. Basic_Auth_Token only contains user's identity, Extend_Auth_Token is an extension to the Basic_Auth_Token, it includes user's identity, static attributes set and dynamic attributes set.

Definition 4. **Confusion Token**, a kind of data structure for privilege revocation. It describes the rules to create, insert and delete the confusion block, and we use these rules to scramble the ciphertext whose access permissions need to be updated.

3 Our Multi-mode Access Control

Figure 1 gives the model of our multi-mode access control for data sharing in cloud. According to the figure, IBAC scheme is used to quickly verify user's identity in private area, ABAC scheme is chiefly adopted in group area for its flexible and fine-gained access control, and DABAC scheme is considered for uncertain users in public area.

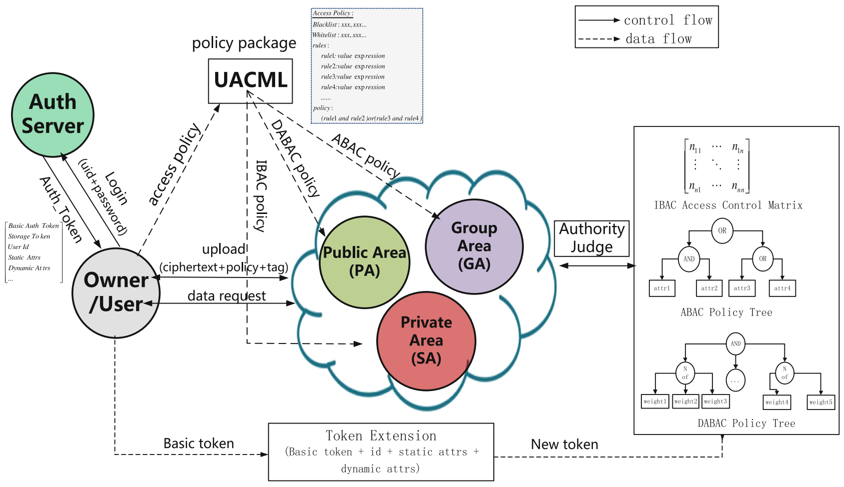


Fig. 1. Multi-mode access control model for data sharing

From Fig. 1 we can find that the process of access control mainly involves four parts: authentication, access policy expression, privilege decision and privilege revocation.

Authentication is used to determine whether the user's request is valid when he logs in the system. Every user should register to the *Auth Server* during the system initialization. If the user is a legal user, *Auth Server* will assign a global unique user's identity (UID) to the user. We use the structure Auth_Token to describe UID which is unique and timeliness. When Auth_Token comes to its end of life, the system will reassign a new Token to the user. UID is used for IBAC scheme to verify the validity of access. User's static attributes set in the structure Extend_Auth_Token is associated with his group characteristics, which means that the element in a user's S_{static_attr} comes from his multiple different groups. User's dynamic attributes set in the structure Extend_Auth_Token is only associated with user's behavior and environment, which means that the element in a user's $S_{dynamic\_attr}$ may be changed with the user's access trace.

3.1 Access Policy Formulation

We have given a general description on the access policy with our UACML and illustrated the meanings of all tags and labels in Sect. 2. Here, we give the corresponding structure schematic about the policies of IBAC, ABAC and DABAC (shown in Fig. 2).

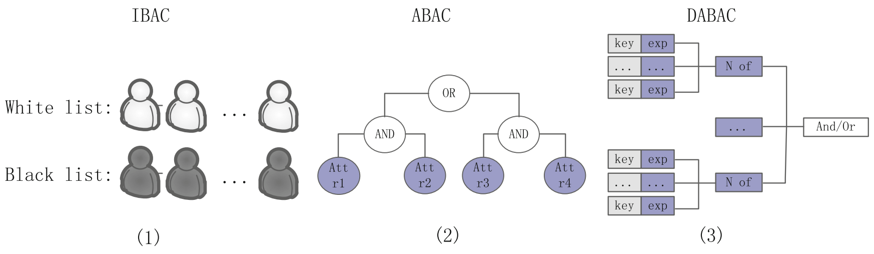


Fig. 2. Structure schematic for different policies

According to the rule of UACML, data owner only need to define the whitelist or blacklist when formulates the access policy for data stored in private area, and to write the valid users' identities into ACL. In our system, the priority of blacklist's decision is higher than whitelist's, namely, if a user is both in whitelist and blacklist, he still has no access to the data.

A binary tree is usually used to represent the policy of ABAC, where the leaf nodes indicate the static attributes, while the non-leaf nodes are logical operators, such as *and*, *or*, *not*, etc. The main work of policy formulation is to set the atom-unit of *rule* labels in UACML, each *item* in rule label implies a static attribute, the policy can be finally formulated through the logical combination of the *item* based on logical operators, e.g.

$$Policy = ((developer\ and\ java)\ or\ (designer\ and\ \max 3d))\ and\ work\_years > 2$$

The policy of DABAC is also represented in a form of tree structure, but it is different from ABAC's because the policy tree of DABAC is a multi-tree, its leaf nodes are generally expressed with the conditional expressions that the dynamic attributes should meet, and the non-leaf nodes are logical operators or weight value, etc. Generally, the *item* is a conditional expression of dynamic attributes attached a weight options, more detailed description can be seen in Sect. 3.3.

3.2 Privilege Decision

The process of decision is related to the access control policies. In this section we will describe the processes of policy matching by taking an example of DABAC. Due to the limitation of space, the decision processes of IBAC and ABAC are not discussed. Figure 3 shows the multi-tree structure of DABAC's policy which is corresponding to the policy formulation given in Table 1.

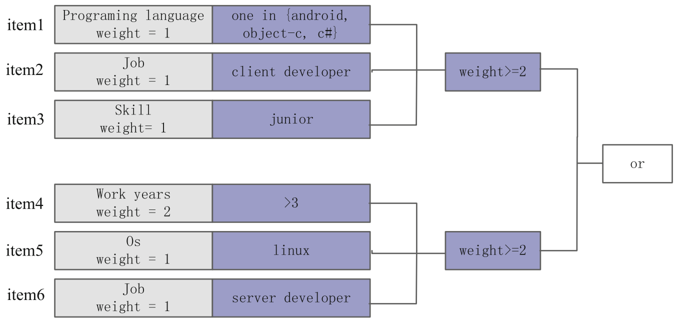


Fig. 3. The multi-tree structure of DABAC's policy

In Fig. 3, the policy tree is a three-tier multi-tree, and its root is logical word 'or'. Left subtree requires weight ≥ 2, in which the weights of item1, item2 and item3 are all required to be equal to 1. Therefore, in order to meet the access condition of left subtree, the user's attribute set should satisfy at least two or more access conditions among item1, item2, and item3. The expression of item1 requires that user's programming language must be object-c or android or c#, so if a user's attribute value of programming language is in this range, he will meet this condition. The expression of item2 requires that the user's job is a client developer, so if the values of a user's working attributes contain 'client developer', he will also meet this condition. The expression of item3 requires that the user's skill proficiency should be at least junior, here junior is a range rather than an exact value, thus if the value of a user's skill attribute is not under the lower limitation of junior, he will meet the access requirements. The decision processes of the right subtree are similar to the left subtree.

After the authority judgment, the system will send feedback to the user attached with his dynamic attributes value. It is generally required that TAG labels in UACML should be set for DABAC scheme, and the value of TAG labels combines dynamic attribute with correction value. After a user accesses the data, the system should not only respond to his request, but also return the TAG labels to the user, and then the user updates his dynamic attribute value according to the TAG label value.

As shown in Fig. 4, the user Job has an uncertain attribute 'dev' in his $S_{dynamic\_attr}$, its original value is 126 and its correction value is 2. So after Job accesses the certain object, the value of attribute 'dev' will be added 2 and become to 128 according to the returned result of TAG label. When the value of attribute ‘dev’ is added to be greater than the threshold (here, assumed to be 127), uncertain attribute ‘dev’ will become the user’s official attribute and thus affect the authority decision.

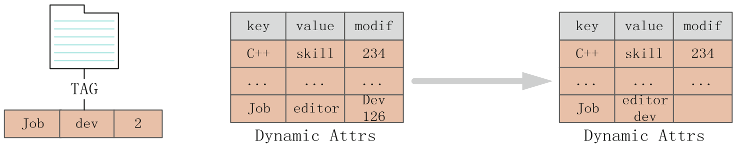


Fig. 4. The processes of dynamic attribute’s change

3.3 Privilege Revocation

Most revocation methods need to re-encrypt the data and update the access keys of valid users. Therefore, for user’s frequent entrance or revocation in cloud, it always confronts the dilemma of low efficiency. To solve this problem, we present a new data retreatment solution instead of re-encrypting the data. Figure 5 shows the process of data retreatment with confusion token for revocation.

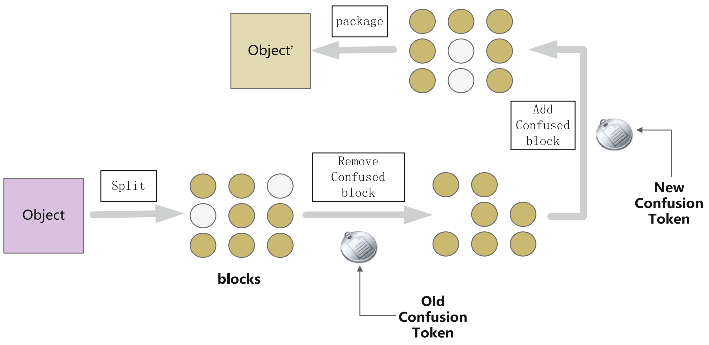


Fig. 5. The process of data re-procession

As described in Sect. 2, confusion token is a structure which defines the rules to create, insert or delete the confused block. When a user’s privilege is revoked, an old token and a new token will be send to the cloud server by authorization center, the cloud server then utilizes the old confusion token to scramble the split blocks (insert some new blocks or delete the shuffled blocks) in ciphertext, and the new confusion token is used to recover the structure of scrambled blocks. In the end, only valid users receive the new token, he can access the data with the new token and the secret key. Whereas, the revoked users will be unable to access the raw data because they don’t have the new token to recover the structure of scrambled ciphertext even if they own the secret keys. Since the ciphertext needn’t be decrypted and re-encrypted, the overhead of revocation is greatly reduced.

3.4 Data Encryption

Data encryption is a basic method to protect data confidentiality. In order to ensure system performance and practicability, we adopt symmetric encryption algorithm AES to encrypt data, in which the random key K_{AES} will be used for data encryption and decryption. To enforce the data access control, K_{AES} is encrypted by public key produced from the PKI, i.e. RSA, CP-ABE. In cloud storage system, we use different asymmetric encryption algorithm to encrypt K_{AES}. RSA is used in private area while CP-ABE is used in group area and public area. Meanwhile, the digital signature generated by RSA will be used in our system to make sure the integrity of data stored in untrusted cloud servers. The process of data encryption is as follows:

$$K_{AES} = rand()^{\wedge} rand()$$
$$E(Data) = aes\_encrypt(Data,\ K_{AES})$$

if private area:

$$PK, SK = rsa\_key\_generate()$$
$$E(K_{AES}) = rsa\_pk\_encrypt(K_{AES}, PK)$$

else:

$$PK, MK = cpabe\_setup()$$
$$E(K_{AES}) = cpabe\_encrypt(PK, Data, T).$$

4 Performance Analysis

In this section, we analyze the performance of our scheme in terms of storage overhead and access performance. We implement the prototype based on the Swift module of OpenStack platform. Swift is a scalable redundant storage system which only provides a basic security mechanism for authentication and access control. So, we modify and replace the related modules with our access control module. The assumption of experiment is as follows: the size of user's UID is 8 bytes, each attribute (including attribute name and its value) is 16 bytes, weight value is 2 bytes and logical operator is 4 bytes.

4.1 Storage Overhead

Figures 6 and 7 show the storage overhead of IBAC, ABAC and DABAC. From them we can see the storage overhead of IBAC has a linear relationship with the number of UID, while the overhead of ABAC and DABAC is mainly associated with the access tree structure. Because of the structural characteristics of the binary tree, the number of access policies that a n-layer binary tree can represent is $2^{n-1} - 1$ (not considering the sort of the leaf nodes). In contrast, access matrix of IBAC needs $2^{n-1} - 1$ ACLs to represent the same number of access policies. We take a 5-layer full binary tree as an example for discussion.

The number of access policies that a 5-layer full binary tree can represent is $2^4 - 1 = 15$, the number of associated attributes is 16, and its storage overhead is $16 * 8 + 15 * 4 = 188$ bytes, while the storage overhead of IBAC is $16 * 15 * 8 = 1920$ bytes under the same circumstances. Hence we can conclude that the larger the number of user is, the more complicate the access policy is, and the better ABAC and DABAC are.

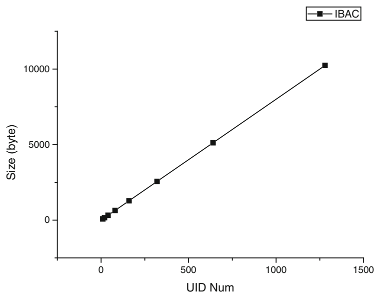


Fig. 6. Storage overhead of IBAC

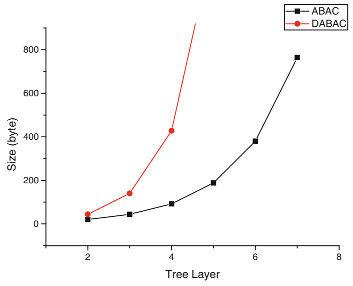


Fig. 7. Storage overhead of ABAC and DABAC

4.2 Performance Evaluation

The two principal factors that affect the security performance are privilege decision and data encryption and decryption. Since our work mainly focuses on the performance analysis of different access control scheme, we utilize the access time to evaluate the performance. It's necessary to emphasize that access time in our experiments is calculated from positioning the object at Server to getting the object at Client which includes the cost of judging a user's permission and network communication.

Figure 8 shows that the access cost of IBAC grows linearly as the number of user increases. That is because the volume of ACL is becoming larger and larger as the number of user grows, so the additional overhead of reading access policy increases accordingly.

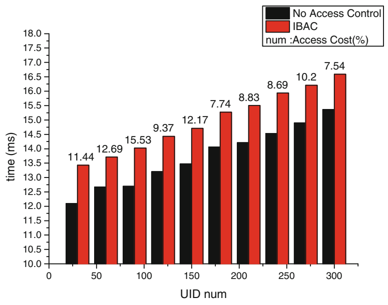
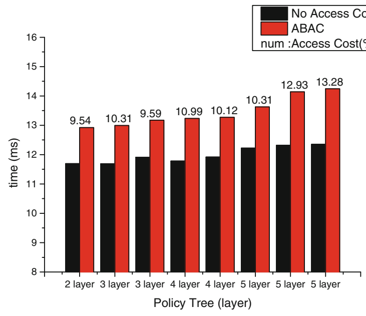
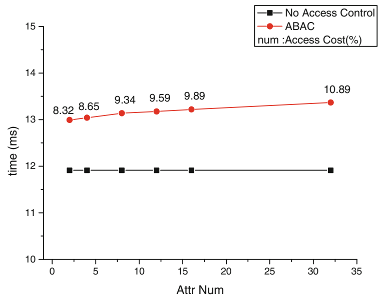


Fig. 8. Access cost of IBAC

Figure 9 shows the access cost of ABAC under two different situations, in which the significant factors are the depth of the access tree and the number of the user's static attribute. As shown in the figure, the access cost becomes larger as the depth of the access tree increases, while the number of user's static attribute has little influence on the cost of decision-making. The reason is that user attributes are organized in a dictionary order in the system, which makes little influence on the query efficiency.



(a)



(b)

Fig. 9. Access cost of ABAC

DABAC has a similar impact factors with ABAC's, but DABAC has higher access cost than ABAC (shown in Fig. 10). The reasons that cause the difference mainly lie in two aspects. (1) The algorithm that DABAC used to judge the user permission is much more complicated than ABAC's. (2) After authorization, some callback functions will revise the values of a user's dynamic attributes which incurs extra performance overhead. So, DABAC's access cost is higher than ABAC's, but it can execute dynamic access control.

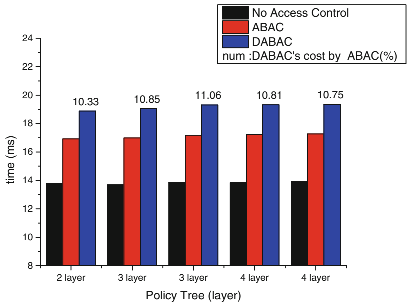


Fig. 10. Access cost of DABAC

As shown in the test results, the net cost of the user authorization is in millisecond level (<10 ms), and when there is no access control, the time from locating the object at Server to getting the object at Client is generally between 10 ms and 20 ms. Contrasting to the time cost (often in second level) of network transmission and data encryption and decryption, our system maintains high performance.

4.3 Revocation Efficiency

From Figs. 11, 12 and 13, we show the time cost of revocation using AES re-encryption and our token scheme respectively. Here, the process of re-encryption contains two steps: to decrypt the ciphertext and then to encrypt the raw data using the new keys. In our test, the delay of network transmission is ignored, and 128-bit keys are adopted in re-encryption scheme, the size of plaintext is from 4 KB to 128 MB, and the ciphertext is divided into twenty pieces of data blocks.

Figure 11 shows that the scheme of re-encryption costs about 25 ms for each MB data when enforcing the revocation, while our token scheme only costs about 6 ms for inserting confused blocks and less than 1 ms for reordering blocks. Hence it is very obvious that the computational overhead of our revocation scheme is far less than that of re-encryption scheme, even though the network transmission overhead is not taken into consideration.

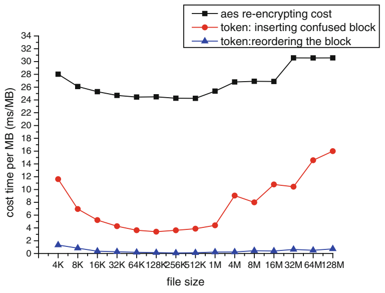


Fig. 11. Time cost per MB data

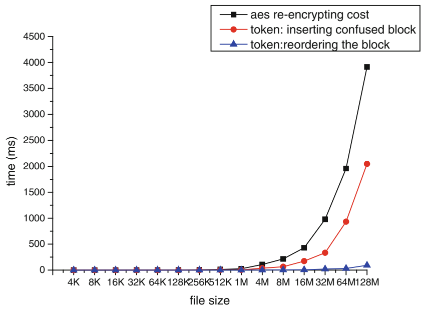


Fig. 12. Time cost for 20 blocks

Figure 12 shows that when the size of file is bigger than 4 MB, the time cost increases sharply for the scheme of re-encryption, and the reordering token scheme almost keeps stable. For example, the time cost for 128 MB-size file of our reordering token scheme is 92.80 ms, while the re-encryption scheme costs 3913.14 ms, and the confused block scheme needs 2047.03 ms. So our revocation scheme is much more efficient than the re-encryption scheme.

From Figs. 11 and 12, we can find the different rules of token has also influence on the efficiency of revocation. We shows the effects of block number on revocation time using two kinds of basic rules, that is inserting the confused blocks into the ciphertext and scrambling the ciphertext (seen in Fig. 13, given 4 MB-size data). From Fig. 13 we can see, with the increase of block number, the time cost of confused blocks shows a linear growth trend, while that of reordering blocks moves in a tight range. That is because the rules of confusion block are much more complicated than that of reordering block, and the former algorithm is also robust enough against attacks, so has better security than the latter.

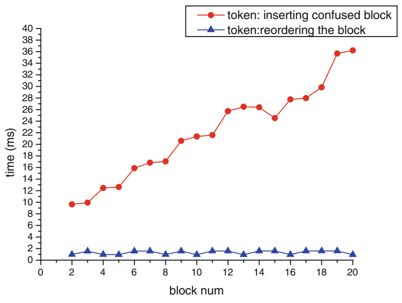


Fig. 13. Time cost vs block number (for 4 MB data)

5 Related Work

Ferraiolo, D.F. and Sandhu, R. provided a kind of RBAC [11], where roles are systematically defined. An access control list (ACL) is attached to the data, which contains the list of users authorized to access data. This is not feasible in cloud where the number of users is enormous and the status of users also often changes dynamically. Slimani et al. [12] expanded the description of access policy and semantic interoperability, which refers to the idea of XACML to set formalized constraints for access policies, but does not consider the change of attribute. What's more, XACML s access control-related web services standards which language is designed without a formal ABAC model. Literature [13] combines role-based access control and attribute-based access control in a private cloud to achieve the mandatory and discretionary access control, but it loses the flexibility of ABAC because of its combination with role-based access control. Literature [7] proposes an attribute-based access control scheme with several authorization centers to deal with the performance bottleneck caused by a single authorization center, however it causes attribute conflict between different authorization centers when recovering attribute, and it also does not give the sound solution to the revocation. In order to solve the problem of revocation, Hur and Noh [14] proposes a lazy permission revocation scheme from key refreshing with the combination of attribute encryption algorithm, but the security problem caused by the uncertainty of writing time is still potential in the scheme, and it does not discuss the way of revocation for read-only data. G. Wang, Q. Liu and J. Wu presented a hybrid access control model in [15] involving attribute based encryption (ABE), proxy re-encryption and lazy encryption, where complexity and overhead for policy specification increases with the number of attributes and steps required to execute mathematical operations.

6 Conclusions

In this paper, we built a multi-mode access control system for data sharing in cloud that enables only the authorized users to decrypt a ciphertext. Our proposed construction is efficient for revocation and supports both backward and forward security because it is based on the CP-ABE scheme. Specially, our system can support dynamic access control which is very suitable for frequent changeable cloud environment. Our token revocation scheme is a promising technique, which can be applied in any remote storage systems and online social networks etc. In the next phase of the research, we will focus on the algorithm of scrambling confusion blocks to further improve our performance and security.

Acknowledgments. Firstly, the authors would like to thank the anonymous referees of ICA3PP 2015 for their reviews and suggestions to improve this paper. Secondly, the work is supported by the National High Technology Research and Development Program (863 Program) of China under Grant No. 2013AA013203, and also supported by the National Natural Science Foundation of China under Grant No. 61232004.

References

1. Masood, R., Shibli, M.A.: Comparative analysis of access control systems on cloud. In: 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel and Distributed Computing (SNPD), pp. 41–46. IEEE (2012)
2. Ruj, S.: Attribute based access control in clouds: a survey. In: Signal Processing and Communications (SPCOM), pp. 1–6 (2014)
3. Sirisha, A., Kumari, G.: API access control in cloud using role based access control model. In: Trendz in Information Sciences and Computing (2010)
4. Sanka, S., Hota, C., Rajarajan, M.: Secure data access in cloud computing. In: International Conference on Internet Multimedia Services Architecture and Application (2010)
5. Lee, C.-C., Chung, P.-S., Hwang, M.-S.: A survey on attribute-based encryption schemes of access control in cloud environments. IJ Netw. Secur. **15**(4), 231–240 (2013)
6. Yu, S., Wang, C., Ren, K., Lou, W.: Achieving secure, scalable, and fine-grained data access control in cloud computing. In: IEEE INFOCOM, pp. 534–542 (2010)
7. Chase, M., Chow, S.S.M.: Improving privacy and security in multi-authority attribute-based encryption. In: Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS 2009), pp. 121–130 (2009)
8. Yu, S., Wang, C., Ren, K., Lou, W.: Attribute based data sharing with attribute revocation. In: Proceedings of the 5th ACM Symposium Information, Computer and Comm. Security (ASIACCS 2010), pp. 261–270 (2010)
9. Hur, J., Noh, D.K.: Attribute-based access control with efficient revocation in data outsourcing systems. IEEE Trans. Parallel Distrib. Syst. **22**(7), 1214–1221 (2011)
10. Xu, Z., Martin, K.M.: Dynamic user revocation and key refreshing for attribute-based encryption in cloud storage. In: 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 844–849 (2012)
11. Ferraiolo, D.F., Sandhu, R., Gavrila, S.: Proposed NIST standard for role based access control. ACM Trans. Inf. Syst. Secur. **4**, 224–274 (2001)
12. Slimani, N., Khambhammettu, H., Adi, K., et al.: UACML: unified access control modeling language. In: 2011 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pp. 1–8 (2011)
13. Mon, E.E., Naing, T.T.: The privacy-aware access control system using attribute-and role-based access control in private cloud. In: 2011 4th IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT), pp. 447–451 (2011)

14. Hur, J., Noh, D.K.: Attribute-based access control with efficient revocation in data outsourcing systems. IEEE Trans. Parallel Distrib. Syst. **22**(7), 1214–1221 (2011)
15. Wang, G., Liu, Q., Wu, J.: Achieving secure, scalable, and fine-grained data access control in cloud computing. In: IEEE Proceedings of INFOCOM (2010)