International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 5 Issue VII, July 2017- Available at www.ijraset.com

Secure Data Access Control System Using a Robust and Verifiable Threshold Multi Authority Technique with Updating Of Key in Public Cloud Storage

Dr. Shubhangi D.C[1], Suresh [2]
[1]*H.O.D,* [2]*P.G.Student, Department of Computer Science and Engineering,*
VTU Regional Centre for PG studies, Kalaburagi, Karnataka, India.

Abstract: ***Attribute-based Encryption (ABE) is considered as the most commonly used cryptographic conducting tool in order to assure the direct control of the data owner over the cloud storage public data. Previous ABE methods includes single authority to maintain the whole attribute set, due to which the major drawback is seen in performance and security issues. Eventually few multi-authority methods are proposed where several authorities maintain separate attribute sets. In this paper, a threshold multi-authority CP-ABE access control method for public cloud storage, named TMACS is proposed, where several authorities jointly handle same attribute set. In TMACS, advantage of (t; n) threshold secret sharing is taken, the master key will be shared among several authorities, and authorized user can generate his/her secret key by communicating with any number of t authorities. Additionally, by efficiency of combining the traditional multi-authority method with TMACS, we develop a hybrid one, which provides the efficiency of the framework of attributes reaching from several authorities as well as achieving system-level robustness and security.***

Keywords: ***Public cloud storage, access control, CP-ABE, (t,n) threshold secret sharing, multi-authority, Certificate Authority***

I. INTRODUCTION

The important service of cloud computing is the cloud storage [1], which mainly provides the services for cloud data owners to produce data to get saved in cloud though internet. Although there are many advantages of storing the data in cloud but still there are many issues which mainly includes the security and privacy of user's data specially on cloud storage [2][3]. Most commonly, the data owner will save his or her data on trusted cloud servers, these cloud servers are mostly managed by fully trusted administrator. Moreover, cloud providers commonly a semi-trusted third party in public cloud storage systems usually maintain and manage the cloud. Here the data is no longer saved in data owner's trusted domains and the data owner will not have trust on the cloud server for conducting security of data access control. Hence, the security access control issue is becoming a crucial challenge in cloud storage where general security technologies cannot be imposed directly on cloud. The most suitable method for conducting security access control in public cloud storage is Attribute-based Encryption (ABE) [4], [5], [6] which assures the direct control of the data owners over their cloud data and also it provides a simple access control service. Many ABE methods have been proposed which are divided into 2 types namely: Key-Policy Attribute-based Encryption (KP-ABE), such as [7], [8], and Cipher text-Policy Attribute-based Encryption (CP-ABE), such as [9], [10], [11], [12].

In KP-ABE methods, decryption keys are combined with access structures while special attribute sets are used to label cipher texts. On the other hand, in CP-ABE methods, an access control policy is defined by the data owners on every file attribute sets of cloud user's data to assure the direct and complete control on the data. Hence when we compare these 2 types of methods the most commonly used method is CP-ABE.

In the existing systems there is a drawback for public cloud storage, where a single-authority brings a common issue on performance and security on cloud data for all specific attribute sets. In this a multi-authority access control framework is introduces for the first time to handle the problem of single-authority.

Here we proposed a realistic, robust and a valid multi-authority CP-ABE method by introducing a combination of (t, n) threshold secrete sharing in public storage where several authorities combining control a uniform attribute set. Additionally, by efficiency of

combining the traditional multi-authority method with TMACS, we develop a hybrid one, which provides the efficiency of the framework of attributes reaching from several authorities as well as achieving system-level robustness and security.

II. LITERATURE SURVEY

R. Ostrovsky, A. Sahai, and B.Waters developed an Attribute-based Encryption (ABE) method [5] which authorize a user's private key to be generated in the form of any access formula over any attribute. Earlier ABE methods were restricted only for monotonic access frameworks.

Now-a-days most important information is stored by the 3rd party websites on the web[7]; hence there is a need of encrypting the data that is stored on cloud at various websites. The main drawback of encrypting the data is that it is very critical where the private keys are shared at some level. A new cryptosystem have been developed called as Key Policy Attribute-Based Encryption (KP-ABE) for sharing of encrypted data in a fine grained level.

In the cryptosystem, cipher texts are assigned with the sets of attributes and private keys which are related with the access form which controls which cipher text is used by the user to perform decryption. They denote the application of the construction of sharing of audit-log data and later broadcast the encryption. The construction helps in delegating the private keys which assumes Hierarchical Identity-Based Encryption (HIBE). In many distributed systems a user should contain a definite set of attributes or credentials. Presently, a method is used for imposing policies so as to appoint a trusted server to save the information and moderate control over access. Moreover, if a server's storage for data gets compromised then the security for that data will also be compromised.

J. Bethencourt, A. Sahai, and B. Waters proposed a system which helps in realization of access control on the data that is encrypted which is mainly called as Cipher text-Policy Attribute-Based Encryption[9]. Applying these methods the data encrypted will be kept secretly even if the cloud storage server is not trusted; furthermore theses methods are more secure over collision attacks. Earlier Attribute- Based Encryption systems describe the data encryption by using some attribute sets whereas in this system user's credentials are mainly described by the attributes and a third party is used to determine which user can decrypt the data.

B. Waters proposed a new methodology for realizing Ciphertext-Policy Attribute Encryption (CP-ABE)[10] under concrete and non-interactive cryptographic assumptions in the standard model. Our solutions allow any encryptor to specify access control in terms of any access formula over the attributes in the system. In our most efficient system, ciphertext size, encryption, and decryption time scales linearly with the complexity of the access formula.

Data access control is a powerful way to assure the information security inside the cloud. Because of information outsourcing and untrusted cloud servers, the information access control has become a challenging problem in cloud storage structures.A Cipher text-policy attribute- based Encryption (CP-ABE)[18] is appeared as one of the maximum suitable technologies for information access control to manage in cloud storage, as it offers data owners a direct control on the access rules. Moreover, it is problematic to directly apply present CP-ABE methods data access control in cloud systems storage for data due to the attribute revocation issue .Here the authors design an extensive, efficient and revocable information access control method for multi-authority cloud storage structures, in which there are more than one authority co-exist and every authority is capable of issuing attributes independently. Specially, the authors introduce a revocable multi-authority CP-ABE method, and apply it as the underlying strategies to develop the data access control methods. Our attribute revocation technique can efficaciously gain each forward safety and backward security. The analysis and simulation outcomes show that our proposed information access control method is more comfortable within the random oracle version and is extra efficient than preceding works.

III. SYSTEM ARCHITECTURE

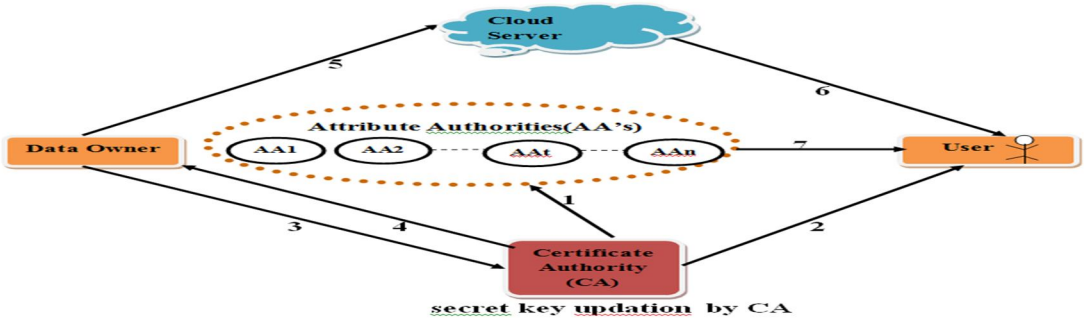


Fig. 1 System Architecture

A. Attribute Authorities (AA's) register to Certificate Authority(CA) to gain Attribute authority id (Aid)
B. Users register to Certificate Authority to gain user id (Uid)
C. Data Owner requests the public key (PK) to Certificate Authority.
D. Certificate Authority sends the public key to the data owner.
E. Data owner encrypts the data using public key and uploads to the cloud server.
F. User downloads the encrypted file.
G. User gains the encrypted secret key from any 't' AA's among 'n' AA's to decrypt the file.

In a strong multi-authority public cloud storage structures, there exist five modules: a global certificate authority, several attribute authorities (AAs), data owners (Owners), data consumers (Users), and the cloud server.
The certificate authority (CA) is a global trusted module in the structure which is responsible for the development of the system by setting up attribute public key of each attribute in the complete attribute set and system parameters. Certificate authority accepts cloud users and authorities registration requests by assigning a distinct uid for every authorized cloud user and a distinct aid for every attribute authority. Certificate authority also determines the parameter t about the threshold of Attribute authority's that are involved in cloud users' secret key generation for each time. However, Certificate authority is not involved in Attribute authority's master key sharing and cloud users' secret key implementation. The secret key will be updated after downloading the file. The updation of the secret key will helpful during proof of storage of data in the cloud server so that the cloud service provider can't get the secret key
The attribute authorities mainly focus on the task of attribute management and key generation. Besides, attribute authority's takes part in controlling the development of the system they also has the capability to monitor or handle the applications of the system.
The data owner mainly encodes his/her file and defines access rules about who can get access to his/her data file. Firstly, every data owner has to encodes his/her data file with a symmetric encryption algorithm like AES and DES. Later the data owner formulates access rules over an attribute set and encodes the symmetric key inside the rules as per the attribute public keys gained from certificate authority.
The data consumer (User**)** is associated with a global cloud user identity uid from certificate authority, and uses it for his/her secret keys from attribute authority's with his/her determination. The cloud user will easily get the cipher texts that he/she is requested in from cloud servers. His/her can decode the encoded information if and only if his/her attribute set satisfies the access rules hidden within the encoded data. The user has to send the request to the attribute authorities for getting the secret key for downloading the same file for the second time because the secret key is updated once the user has downloaded the file by the certificate authority.
The cloud server is mainly responsible for providing a platform for data owners for sharing and storing their encoded data. The encoded data uploaded on a cloud can be easily downloaded by any cloud data user.
The architecture of TMACS is represented in figure 1. In TMACS, firstly the attribute authority's must register to Certificate authority to access corresponding certificate and identity (aid, aid.cert). Later, the attribute authority's will be involving in the development of the architecture, assisting Certificate authority to complete the development of system attributes. The users' registration is accepted by the Certificate authority and provides the certificate (uid, uid.cert) to every authorized user. With this certificate, the authorized user can communicate with any t attribute authority's one by one to access his/her secret key. Data Owners who want share their information in the cloud can access the public key from Certificate authority. Later, the encoded data uploaded on a cloud can be easily downloaded by any cloud data user from any cloud server.
Moreover, he/she can't decode the cipher text unless his/her attribute sets satisfy the access rules hidden within the cipher texts.
One challenging problem in structuring the TMACS is reusable of the master key which is shared among several attribute authorities. In traditional (t, n) threshold secret sharing, once the secret is rebuild among several data users, someone can actually access its value. Likewise, in CP-ABE methods, the single-authority knows the master key and uses it to develop every user's secret key according to a specific to set of attributes. Here, if the attribute authority is adjusted by a contender, it will become the security issue. In order to overcome from this, it uses by means of (t, n) threshold secret sharing, the master key cannot be individually rebuild and gained by any entity in TMACS.

IV. METHODOLOGY

A. *System Initialization*

The operation of System Initialization is divided into three sub-processes: CASetup1, AASetup, and CASetup2. The operation of CASetup1 is mainly responsible for establishment of system parameters and accepting registration of users and AAs. AAs cooperate with each other to share the master key in AASetup, while the corresponding public key is generated by CA in CASetup2.

CASetup1: The operation of CASetup1 is run by CA. First, CA chooses two multiplicative cyclic groups G and G_T with the same prime order p, then defines a binary map e : G X G -> G_T on G. CA chooses a random a $\in Z_p$ as the master key, and then calculates the relevant public key part g^a. Here, the parameter g is a generator of G. CA generates a pair of key (sk_{CA}; vk_{CA}) to sign and verify, in which, vk_{CA} is publicly known by each entity in the system. CA also generates public keys for each attribute Att_i(i=1, 2.. . . . . U): h1, h2; . . . , $h_U \in$ G. To distinguish each user and AA, the following two processes are described separately:

User registration: Each user sends a registration request to CA during the phase of System Initialization. CA authenticates the user, then assigns an identification uid to him/her. The identification uid is a random element in Z_p. CA signs a certificate uid.Cert with sk_{CA} for the user to verify his/her identity.

AA registration: Each AA also sends a registration request to CA during the System Initialization. For each legal authority, CA assigns a unique identity aid $\in Z_p$ and generates a certificate aid.Cert.

According to the total number (marked as n) of AAs involved in the system, CA decides the threshold number (marked as t) of AAs that participate in user's secret key generation for each time.

AASetup: The operation of AASetup is run by each one of all n AAs. These n AAs cooperate with each other to call (t, n) threshold secret sharing as follows:

Each AA (AA_i, i= 1, 2, . . . , n) selects a random number $\alpha_i \in Z_p$ as its sub-secret, in this way, the master key a is implicitly decided: $\alpha=\sum^{n}_{i=1} \alpha_i$. The value of α shouldn't be gained by any entity alone. Then each AA (AA_i, i=1, 2, . . . ,n) separately generates a random polynomial $f_i(x)$ of degree t - 1 that satisfies the formula $a_i = f_i(0)$. AA_i calculates the sub-share $s_{ij} = f_i(aid_j)$ for each of the other AA (AA_j, j = 1, 2, . . . , i – 1,i +1, . . . , n), and sends the sub-share s_{ij} to AA_j securely. Meanwhile, AA_i calculates $s_{ii} = f_i(aid_j)$ for itself.

After receiving n - 1 sub-shares s_{ji} (j = 1, 2, . . . , i – 1, i + 1, . . . , n) from all of the other n - 1 AAs (AA_j, j =1, 2, . . . , i -1, i + 1, . . . , n), AA_i calculates its master key share: $sk_i = \sum^{n}_{j=1} s_{ji}$, then AA_i further calculates its relevant public key share: $pk_i = e(g, g)^{ski}$.

After finishing the operation of AASetup, each AA (AA_i, i = 1,2, . . .,n) gains a pair of keys (sk_i, pk_i). Here, the public key share pk_i can be shared with any other entities, including CA.

CASetup2: The operation of CASetup2 is run by CA. To calculate the global public key, CA randomly chooses t out of n AAs' public key shares, denoted as pk_i, i = 1,2, . . . , t.

Description	$E_{k(M)}$	$C,C', \{C_i, D_i\}_{i=1 \text{ to } l}$

Fig. 2 Ciphertext format

Now, the operation of System Initialization is finished, and it returns the public key PK:

PK = $(g, g^{\alpha}, e(g, g)^{\alpha}, n,t, h1, . . . , h_U)$

B. *Encryption*

The operation of Encryption is implemented by a specific data owner independently. To improve the system's performance, the owner first chooses a random number $k \in Z_p$ as the symmetric key and encrypts the plaintext message M using k with the symmetric encryption algorithm, such as AES. The encrypted data can be denoted as $E_k(M)$, then the owner encrypts the symmetric key k using CP-ABE under an access policy defined by himself/herself. The owner randomly selects r1, r2, . . . , $r_l \in Z_p$ and calculates the ciphertext CT using the public keys gained from CA:

CT = $(C = ke(g, g)^{\alpha s}, C' = g^s,$

$\forall i = 1 \text{ to } l; C_i = (g^{\alpha}) \lambda^i .h_p(i)^{-r}_i , D_i = g^r_i)$

Finally, the owner sends the encrypted data $E_k(M)$ encrypted by the symmetric key k together with cipher text to the cloud server as the format in figure 2.

C. Secret Key Generation

The Secret Key Generation operation is run by one user and any t out of n AAs. Less than t AAs, user's secret key cannot be generated. In this operation, there is no interaction between any two of t AAs, so the user can select t AAs according to his/her own preference, and then separately contact with each of these t AAs to get the secret key share. After getting t secret key shares separately from t AAs, the user can generate his/her secret key. To gain the secret key share from AA_i, the user uid_j first sends his/her signed request including his/her identity and his/her certificate to AA_i. After receiving the request, AA_i verifies uid_j's certificate by using CA's public verification key vk_{CA}, then authenticates the user by verifying his/ her signature over the request. If the user is an illegal one, the operation aborts. Otherwise, AA_i assigns an attribute set
S to the user according to the role he/she plays in the domain and generates the secret key share for him/her. AA_i first chooses a random number $b_i \in Z_p$ and then generates the secret key share as follows:
$K_i = g^{sk}{}_i \cdot g^{a.b}i$, $L_i = g^b{}_i$, $\forall Att_i \in S : K_{Atti} = h_{Atti}{}^{bi}$

D. Decryption

The Decryption operation is run by each user. The user can freely query and download any encrypted data that he/she is interested in from the cloud server. However, he/she can't decrypt the data unless his/her attribute set satisfies the access structure hidden inside the ciphertext. For the user U, let MU be a sub-matrix of M, where each row of M_U corresponds to a
Specific attribute in U's attribute set S_U.

V. RESULT AND PERFORMANCE ANALYSIS

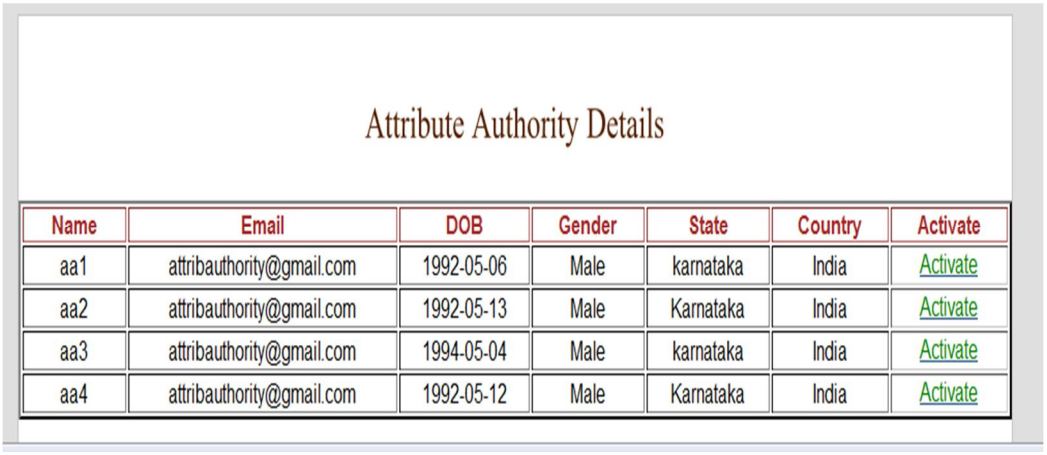


Fig. 3 Attribute Authority Activation by Certificate Authority.

After activating user, owner and Attribute Authorities the id's of the respective entities will be sent to their mail id's.

Fig. 4 Uploading the file by the owner.

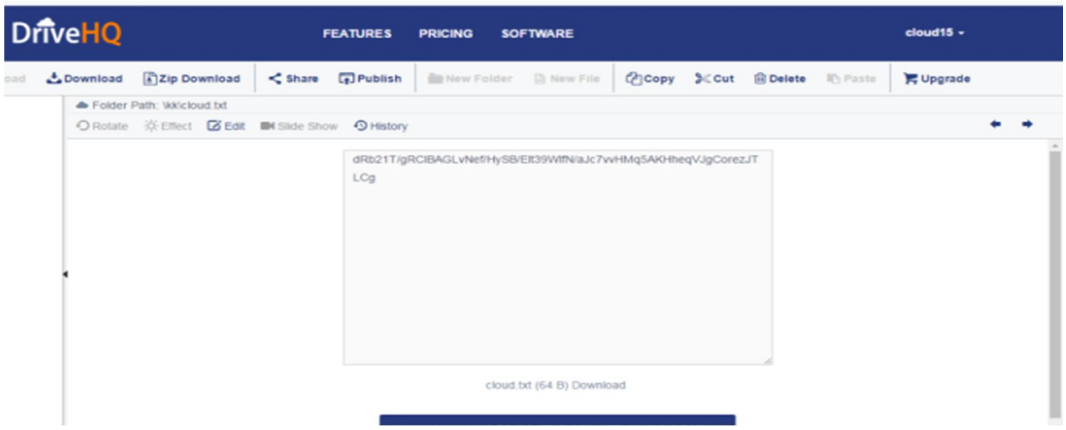
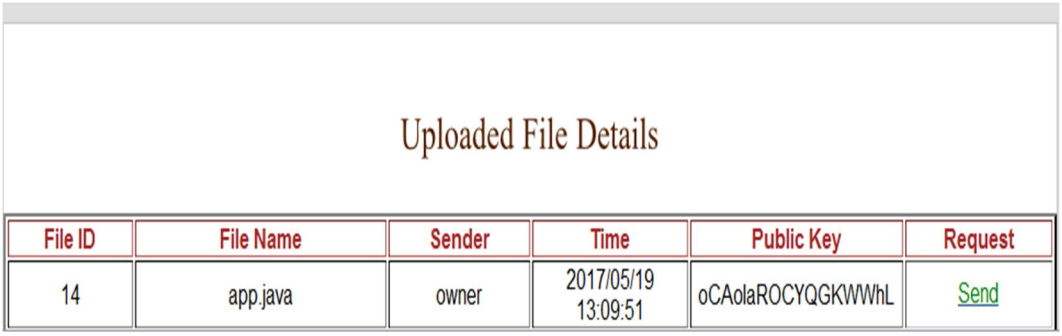


Fig. 5 File stored in Drive HQ cloud server.



Uploaded File Details

File ID	File Name	Sender	Time	Public Key	Request
14	app.java	owner	2017/05/19 13:09:51	oCAolaROCYQGKWWhL	Send

Fig. 6 User will send a request to Attribute authorities for the secret key

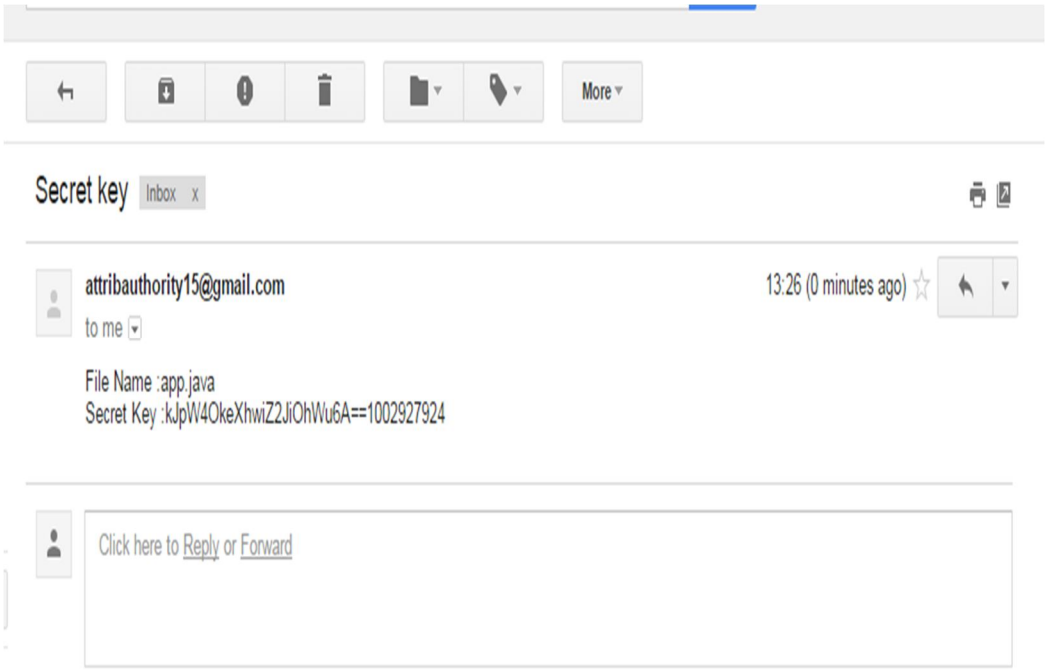


Fig. 7 The secret key will be sent to the user's mail id

A. Security against Collusion Attack

When some malicious users collude with each other, they may share their secret keys to gain more privilege. However, they will be disappointed for the existence of the random element d in the secret key. Due to the existence of the random element, each component associated with the same attribute in different users' secret keys is distinguishable. So that they can't gain more privilege by combining their secret keys.

B. Confidentiality Guarantee

The cloud server can be seen as an adversary for it is "honest and curious", but it doesn't have any advantages compared with malicious users. The cloud server doesn't participate in AAs' master key sharing or any users' secret key generation. All it does is only storing the ciphertext which makes no difference for it to gain the plaintext, even if it colludes with the malicious users.

C. Security against Compromising AAs

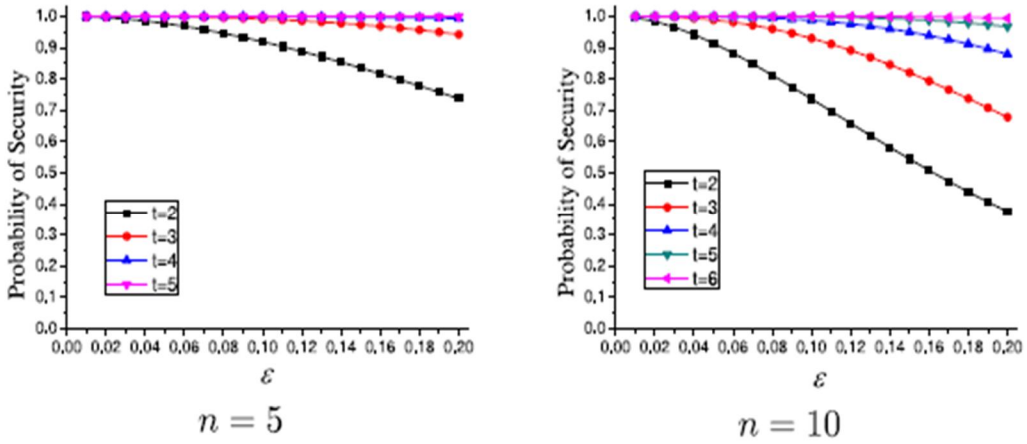


Fig 8: Probability of security against AA compromise.

Here, we give the security analysis against the attack that the adversary compromises AAs. The soundness guarantees that the adversary has to compromise more than t AAs to assign secret key shares for him/her.

To have an intuitive understanding for the security of TMACS against this attack, we draw the probability map in Fig.8, which shows the probability of system security versus the number of AAs n and the probability ". From the figure, we can see that for any number of authorities, we can increase the value of t to make TMACS secure against the above attack as a high probability. But we can also notice that the overlarge value of t will only bring extra overhead rather than increase the security effectively. For example, the system can be secure with probability close to 1, when we set the value of t only equal to 5 rather than a larger value in a system with 10 authorities. We can say, TMACS can be secure against the above attack with an appropriate threshold value t.

D. Robustness against AA Crash or Offline

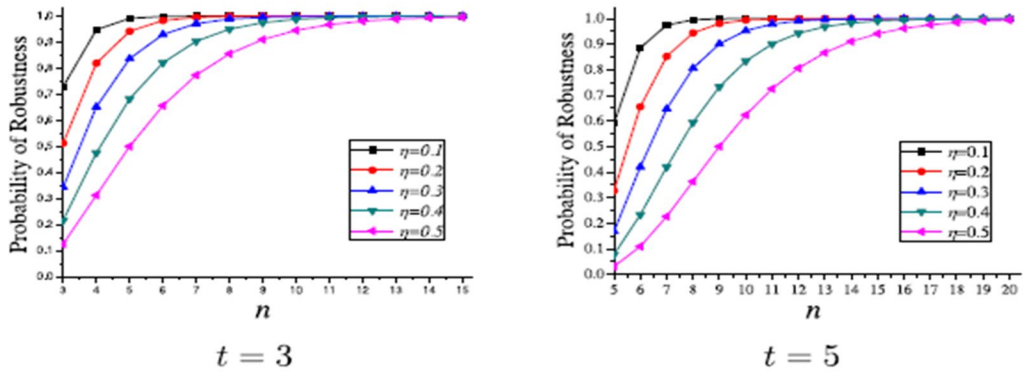


Fig 9: Probability of robustness against AA crash or offline.

In Fig. 9, we draw the probability map versus the number of AAs n and the threshold value t. From the figure, we can see that TMACS can be robust against authorities crashing or offline because of existence of redundant authorities. For example, even if the authority is crashed or offline as a probability of 50 percent, we can still set an appropriate proportion, like t =10; n =30, to make TMACS work properly. By the way, a lager n, like 40, has no use for system robustness, but only increases system overhead.

VI. CONCLUSION AND FUTURE WORK

In this paper we proposed a new threshold multi-authority CP-ABE access control method, namely TMACS, in public cloud structures, in which all AAs combining control the complete attribute set and share the master key a. Taking the advantage of (t, n) threshold secret sharing, through interacting with any t AAs, a legal cloud user can implement his/her secret key. Hence, TMACS avoids any 1 AA being a single-point drawback on both protection and performance.

The analysis results displays that our access control method is secure and robust. We can effortlessly find suitable values of (t,n) to make TMACS not only protected when less than t authorities are compromised, however also robust when no less than t authorities are alive in the structure. Furthermore, based on efficiently combining the traditional multi-authority scheme with TMACS, we also construct a hybrid scheme that is more suitable for the real architecture, in which attributes arriving from different authority-sets and several authorities in an authority-set together manage a subset of the complete attribute set. This enhanced method addresses not only the set of attributes arriving from several authorities but also secured and system-level robustness. How to effectively select the values of (t, n) in theory and develop an optimized interaction rules will be addressed in the future work.

REFERENCES

[1] P. Mell and T. Grance, “The NIST definition of cloud computing,” Nat. Instit. Standards Technol., vol. 53, no. 6, p. 50, 2009.
[2] S. Kamara and K. Lauter, “Cryptographic cloud storage,” in Proceedings of the 14th Financial Cryptography and Data Security. Springer, 2010, pp. 136–149.
[3] K. Ren, C. Wang, and Q. Wang, “Security challenges for the public cloud,” IEEE Internet Computing, vol. 16, no. 1, pp. 69–73, 2012.
[4] A. Sahai and B. Waters, “Fuzzy identity-based encryption,” in Proc. 24th Annu. Int. Conf. Theory Appl. Cryptographic Techn., 2005, pp. 457–473.
[5] R. Ostrovsky, A. Sahai, and B.Waters, “Attribute-based encryption with non-monotonic access structures,” in Proceedings of the 14th ACM conference on Computer and communications security. ACM, 2014, pp. 195–203.
[6] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, “Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption,” in Proc. 29th Annu. Int. Conf. Theory Appl. Cryptographic Techn., 2010, pp. 62–91.
[7] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in Proceedings of the 13th ACM conference on Computer and communications security. ACM, 2006, pp. 89–98.
[8] N. Attrapadung, B. Libert, and E. Panafieu, “Expressive keypolicyattribute-based encryption with constant-size ciphertexts,” in Proc. 14th Int. Conf. Practice Theory Public Key Cryptography, 2011, pp. 90–108.
[9] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” in Proceedings of IEEE Symposium on Security and Privacy. IEEE, 2007, pp. 321–334.
[10] B. Waters, “Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization,” in Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography. Springer, 2011, pp. 53–70.
[11] V. Goyal, A. Jain, O. Pandey, and A. Sahai, “Bounded ciphertext policy attribute based encryption,” in Proc. 35th Int. Colloquium Automata, Lang. Programm., 2008, pp. 579–591.
[12] R. Bobba, H. Khurana, and M. Prabhakaran, “Attribute-sets: A practically motivated enhancement to attribute-based encryption,” in Proc. 14th Eur. Symp. Res. Comput. Security, 2009, pp. 587–604.
[13] M. Chase, “Multi-authority attribute based encryption,” in Proc.4th Theory Cryptography Conf., 2007, pp. 515–534.
[14] H. Lin, Z. Cao, X. Liang, and J. Shao, “Secure threshold multiauthority attribute based encryption without a central authority,” Inf. Sci., vol. 180, no. 13, pp. 2618–2632, 2010.
[15] T. Pedersen, “A threshold cryptosystem without a trusted party,” in Proc. 10th Annu. Int. Conf. Theory Appl. Cryptographic Techn., 1991, pp. 522–526.
[16] A. Shamir, “How to share a secret,” Commun. ACM, vol. 22, no. 11, pp. 612–613, 1979.
[17] M. Ito, A. Saito, and T. Nishizeki, “Secret sharing scheme realizing general access structure,” Electron. Commun. Japan (Part III: Fundam. Electron. Sci.), vol. 72, no. 9, pp. 56–64, 1989.
[18] K. Yang and X. Jia, “Expressive, efficient and revocable data access control for multi-authority cloud storage,” IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 7, pp. 1735–1744, 2013.
[19] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, “Securely outsourcing attribute-based encryption with checkability,” IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 8, pp. 2201–2210, 2014.