Attribute based Access for Content Sharing Networks in Cloud

Suresha D
Assistant Professor,
Computer Science & Engg
Dr Ambedkar Institute of Technology
Bangalore, India

Chethana H R
MTech Student,
Computer Science & Engg
Dr Ambedkar Institute of Technology
Bangalore, India

Abstract— **Content sharing or Information sharing networks are very dynamic in nature in terms of number of users, storage space required, network bandwidth, platform, applications required by users etc. It is not easy for traditional client server model to allocate resources for such networks efficiently. Cloud computing is best suited to be used in such scalable networks than traditional client server model. Cloud storage is one of the fastest growing areas under cloud services. Yet people hesitate to get into cloud, because of security reasons. Unless dealt with security issues properly, we cannot utilize the best of cloud storage. In this paper, we propose an access control mechanism for sharing scalable media, based on data consumers' attributes (e.g., age, job, nationality, designation, or gender) rather than an explicit list of the data consumers' names, called Ciphertext policy attribute based encryption. CP-ABE is efficient and flexible because it allows the owner of the content, to specify the access policy. Only the user who possesses attributes specified in the policy, can decrypt the ciphertext. It works like one to many public key encryption, which means encryption need not be performed for individual users. Also flexibility is given, so that a user who do not possess required attributes can raise request for the data(under emergency conditions) and owner can take decision to give or take away the access whenever he wants. Owner will receive the notifications about such requests through email.**

Keywords— *Cloud storage, CP-ABE, encryption, attribute based, policy*

I. INTRODUCTION

Content Sharing environments are very important part of our lives. They can be social networking sites or any information sharing networks within the corporate offices. As we all know, it is not easy to provide services to information sharing networks because they are very dynamic in number of users, storage requirements, applications, computational capabilities etc. Though cloud computing offers services to such environments people hesitate to get into cloud because of security reasons [1].

Access control is the basic security control mechanism to support information sharing. It applies control over which user can access which resource based on a permission relationship between user attributes and resource attributes. These attributes can be any information considered relevant for granting access, such as user's job function and resource quality. Permission is specified in terms of requirements on the attributes of resource and user. If User's attributes matches the previously set requirements then he can access the corresponding resource. As stated by author in [2], it is very difficult to exert access control policies because (1) any individual is able to freely create any number and any kind of online media such as text, image, sound, video, and presentation; (2) any person is able to grant access to his media to anyone, at any time; (3) an individual may reveal too many attributes (e.g., name, age, address, friendship, classmate, fans, hobby, personal interest, gender, and mobility), and some of them can be very dynamics; and (4) individuals may share their files using various devices and bandwidth.

Hence we should be very careful while designing access control mechanisms. A promising approach is to allow data owners, to enforce control over their data. That way we can eliminate the dependency on a central administrator or 3rd party system. The proposed work here allows the owner to specify required policies which will be based on the attributes of the user. If user's attributes matches with the policy specified by the owner, then he will be able to decrypt the ciphertext. Also under emergency conditions, a user without the specified attributes, can raise request to the owner for the required access. Under this scenario, owner will have authority to respond for the request. He can give and take back the access whenever he wants.

II. RELATED WORK

In Naive access control mechanism, one key will be associated with each attribute of the user. Repeated encryption must be done using the keys appropriately and required number of keys should be shared with the data consumer. Though it looks simple and easy, it results in too many keys and collusion attack.

One more option is to use Role based access mechanism. In this approach users are classified based on their roles and role keys are assigned accordingly. This is widely implemented across many organizations where classification is performed based on the users' job role. Permission to perform operations is defined on the job function. Management of individual user rights become easy, since users are not assigned permissions directly. Rather they acquire the permissions through their roles defined by the higher authority. Though some common operations like adding new users and assigning permissions becomes easier, it becomes highly complicated because of huge number of keys that come into picture. Number of keys for every user and the number of ciphertexts for one message will be order of O(2^n), where 'n' is the number of all possible user attributes.

One more approach Persona [3] employs combination of traditional public key cryptography and attribute-based encryption scheme. But this combination results in increased key management complexity.

Sahai and Waters [4] introduced attribute-based encryption (ABE) as a new method for encrypted access control. In an attribute-based encryption system ciphertexts are not necessarily encrypted to only one particular user as in traditional public key cryptography. Instead users' secret keys and ciphertexts will be associated with a set of attributes and a policy over those attributes. A user is able to decrypt a ciphertext if there is a "match" between his secret key and the policy. In their original system Sahai and Waters have presented a Threshold ABE system in which ciphertexts were labeled with a set of attributes S and a user's private key was associated with threshold parameter k and another set of attributes S′. In order for a user to decrypt a ciphertext at least k attributes must overlap between the ciphertext and his private keys. One of the primary original motivations for this was to design an error-tolerant identity-based encryption [5, 6, 7] scheme that could use biometric identities. The primary drawback of the Sahai-Waters threshold ABE system is that the threshold semantics are not very expressive and therefore are limiting for designing more general systems. Goyal et al. introduced the idea of a more general key-policy attribute based encryption system. In their construction a ciphertext is associated with a set of attributes and a user's key can be associated with any monotonic tree access structure. The construction of Goyal et al.can be viewed as an extension of the Sahai-Waters techniques where instead of embedding a Shamir secret sharing scheme in the private key, the authority embeds a more general secret sharing scheme for monotonic access trees. Goyal et. al. also suggested the possibility of a ciphertext-policy ABE scheme, but did not offer any constructions.

EASiER [8] is an architecture that supports fine-grained access control policies and dynamic group membership by using CP-ABE scheme. In addition, EASiER is able to revoke a user without issuing new keys to other users or re-encrypting existing ciphertexts by using a proxy.

Yu et al. [9] employed KP-ABE (Key Policy Attribute based Encryption) to enforce access policies based on data attributes. Their scheme allows data owners to delegate most of the computation tasks involved in fine-grained data access control to untrusted cloud servers without disclosing the underlying data contents by combining techniques of attribute-based encryption, proxy re-encryption, and lazy re-encryption.

Pirretti et al. [10] proposed an information management architecture using CP-ABE and optimized security enforcement efficiency. Furthermore, they employed the architecture and optimization method on two example applications: an HIPAA (Health Insurance Portability and Accountability Act) compliant distributed file system and a content delivery network.

Akinyele et al. [11] designed and implemented a self-protecting electronic medical records (EMRs) using both CP-ABE and KP-ABE. In order to protect individual items within an EMR, each item is encrypted independently with its own access control policy.

In general systems, there are multiple owners who may perform encryption, according to their own ways, possibly using different sets of cryptographic keys. Letting each user obtain keys from every owner whose records he/she wants to read would limit the accessibility since, owners cannot always be online to grant required access. If we take an example of Health Records, under emergency conditions, hospital would want to have patient's records but patient may not be in a position to give required details. An alternative is to employ a central authority (CA) to do the key management on behalf of all record owners, but this requires too much trust on a single authority (causing the key escrow problem). Key escrow (also known as a "fair" cryptosystem) is an arrangement in which the keys needed to decrypt encrypted data are held in escrow so that, under certain circumstances, an authorized third party may gain access to those keys. These third parties may include businesses, who may want access to employees' private communications, or governments, who may wish to be able to view the contents of encrypted communications.
To overcome above mentioned issues we can step into Cipher text policy attribute based encryption method.

III. NEW DESIGN

In our new design, when the author uploads his file into the server, he will also specify, what are the attributes that the user should posses, in order to decrypt that file. A new user should register himself into the cloud server. During this registration process, all his attributes are read and stored. At the same time, required attributes are used in order to generate a secret key. User has to save this key so that he can provide the same while downloading any file from the server. User will not be able to download any file only with his user name and password that he provides during registration. After registering, user can search for the files that he is looking for. If files are present then the list of relevant files will be provided to him. When user chose the file that he is looking for, he will be asked to input the secret key generated during his registration period. If it is correct, then the file will be decrypted for him and he can read the file. If his password is incorrect then user will be blocked from accessing the file. If the user does not possess the required attributes, then he will be denied with decryption process.

Another flexibility given with this work is, under emergency situation, a user can request for access to the required file. For example, if cloud server is used in internal corporate world, assume certain auditing files can be accessed only by managers and not his team members. When manager is away for a long period, then his team member may have to access those files of his manager to perform the necessary tasks. Manager cannot share his secret key because of confidentiality reasons. In such a situation, team member has to gain access from the relevant person. Such requests will be sent to the author and author can grant him access after verifying users' profile. As soon as request is raised by the user, owner will be notified through email. Owner has to login to the cloud server in order to update access to the user. Once the owner provides access to the user, notification will be sent to the user through email. At a later stage the author can go back and deny him the access.

Sequence diagram of the new design is as shown in fig 1.

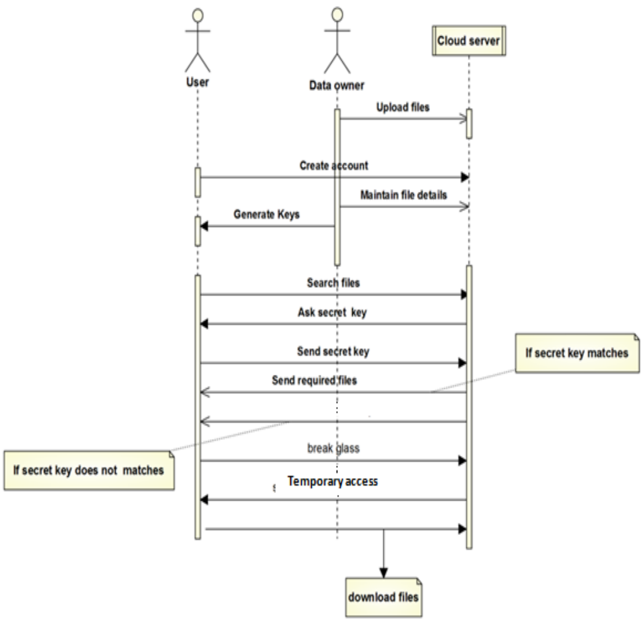

Fig 1 : Sequence diagram

IV. Implementation

The above work has been implemented in ASP .NET with c#. Visual studio has been used to design the front end screens. Azure cloud services have been used to store the data. Main modules and brief description of the modules are given below :

1) User register : Every user must register himself in order to access files on server. During registration process attributes of the user are read and based on required attributes secret key is generated by using MD5 add secret key function.

2) Upload files : Once the owner login into the server he can upload his files using, this module. While uploading files, owner will be asked to specify the policies, which states, attributes required by the data consumers in order to decrypt the file. At the end of this process encrypted file will be saved on the cloud server.

3) Search files : User or data consumer will be able to search for the files that he is looking for. regex class has been used in order to perform search. Related files will be listed out, so that the user can chose the one which he is looking for.

4) Download file : User has to provide his secret key in order to download the file. If his attributes matches with the policy then he can decrypt the file and read the data. If the key is incorrect then the user will be blocked from accessing the file. If the user does not possess the required attributes then message will be displayed saying file is blocked for the user and he will be allowed to request access for that particular file.

5) Manage requests : Requests from different users for different files will be listed for the owner. owner can grant or revoke access to users for corresponding files. An email notification will be triggered for the owner, upon the request submission by the users.

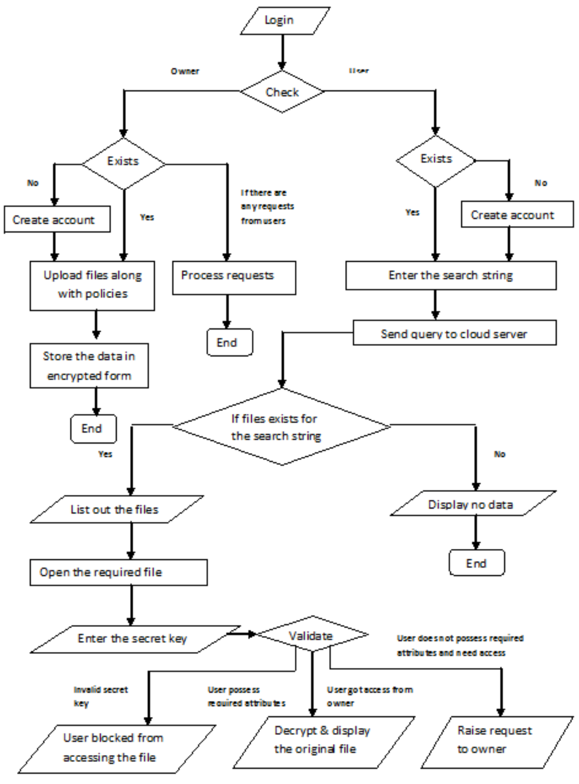
Login
Owner
User
Check
Exists
Exists
No
Create account
Yes
If there are any requests from users
Yes
No
Create account
Upload files along with policies
Process requests
Enter the search string
Store the data in encrypted form
End
Send query to cloud server
End
If files exists for the search string
Yes
No
List out the files
Display no data
End
Open the required file
Enter the secret key
Validate
User does not possess required attributes and need access
Invalid secret key
User possess required attributes
User got access from owner
User blocked from accessing the file
Decrypt & display the original file
Raise request to owner

Fig 2 : Data flow diagram

Fig 2 shows the data flow diagram for the new design. A person who logs into the system can be Admin, user or owner. Admin monitors the owner activities. User and owner should register into the system before performing any actions. When a user registers himself with the system, based on his attributes, secret key is generated and given to him. Owner after registering himself, can upload files. During this upload process he can specify policies which says, who can access his files. When a user logs in, he can search for the files that he is looking for. If files with the search string are present in the server, then they are listed on the screen. He can select the one which he wants. At this point, he has to provide the secret key that he got during his registration process. If the secret key is correct and his attributes match with the policy specified by the owner then he can decrypt the file. If his attributes does not match, then message will be displayed saying "File is blocked for the user" and he will be allowed to raise request for the same file. Once the user raises the request, owner will be notified through email. Now owner can log into the system and he can verify the user details and provide access for his file. Owner can take back the access whenever he wants. If the secret key is wrong then message will be displayed saying "User blocked from accessing files".

V. RESULTS

With this design, encryption need not be performed for individual users. As we know there can be many owners who share their data over the cloud server. In traditional systems, every owner should provide keys to data consumers to decrypt the files. Whereas in our new design, any owner can place his data with pre-specified policies. If user's attributes possess those policies then he will be able to decrypt the data for himself. Problems involved in key distribution of traditional public key encryption are resolved. This works well, even when the cloud server is semi trusted because files will be stored in encrypted form and secret keys are not stored on the database.

VI. CONCLUSION

The above said method can be efficiently used in any content sharing networks since overhead of key distribution is resolved. Also under emergency situations, access can be given to a trusted user. This work has been implemented for a corporate world. Similar concept can be easily applied to provide better security and privacy to both, the users who are looking for jobs as well as the companies searching for potential employees.

REFERENCES

[1] E. Messmer, "Are security issues delaying adoption of cloud computing?," Network World, Apr. 2009 [Online]. Available: http://www.networkworld.com/news/2009/042709-burning-security-cloud-computing.html .

[2] Yongdong Wu, Zhuo Wei, and Robert H. Deng "Attribute-Based Access to Scalable Media in Cloud-Assisted Content Sharing Networks" in IEEE transactions on multimedia, vol. 15, no. 4, june 2013 pp. 778–788.

[3] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, "Persona: An online social network with user defined privacy," in Proc. ACM SIGCOMM Conf. Data Commun., 2009, pp. 135–146.

[4] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attributebased encryption," in Proc. IEEE Symp. Security Privacy, 2007, pp.321–334.

[5] D. Boneh and M. Franklin. Identity Based Encryp-tion from the Weil Pairing. In Advances in Cryptology – CRYPTO, volume 2139 of LNCS, pages 213–229.Springer, 2001.

[6] A. Shamir. Identity Based Cryptosystems and Signa-ture Schemes. In Advances in Cryptology – CRYPTO,volume 196 of LNCS, pages 37–53. Springer, 1984.

[7] C. Cocks. An identity based encryption scheme based on quadratic residues. In IMA Int. Conf., pages 360–363, 2001.

[8] S. Jahid, P.Mittal, and N. Borisov, "EASiER: Encryption-based access control in social networks with efficient revocation," in Proc. ACM Symp. Inf. Computer Commun. Security, Mar. 2011, pp. 411–415.

[9] S.Yu, C. Wang,K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in Proc. IEEE Int. Conf. Comput. Commun., 2010, pp. 1–9.

[10] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, "Secure attribute based systems," in *Proc. ACMConf. Comput. Commun. Security*, 2006,pp. 99–112.

[11] J. A. Akinyele, C. U. Lehmanny, M. D. Green, M. W. Pagano, Z.N. J. Petersonz, and A. D. Rubin, "Self-protecting electronic medical records using attribute-based encryption," in *Proc. ACM Conf. Comput. Commun. Security: Workshop on Security and Privacy in Smartphones and Mobile Devices*, Oct. 2011, pp. 75–86.