Enforcing Cloud Security from Broker using Delegated Access Control

Aashutosh Bhadauriya, Pushkar Hagawane, Swapnil Jagdale, Tejas Gujar

Abstract:- Public clouds are most vulnerable to unauthorized access. Information security in the public clouds can be an issue if the privacy is not preserved. Unauthorized users can access the data and manipulate it which could make the data useless. Existing systems use fine grained encryption approach for providing fine grained access control to maintain the confidentiality of data hosted in the public clouds. In existing systems data owners are responsible for encrypting data before uploading to cloud and re-encrypting data whenever the user credentials change. This makes existing approaches inefficient as data owner suffer from high communication and computation cost. An efficient approach will delegate the authority of fine grained encryption of data towards the cloud still assuring the confidentiality of data. Proposed system will address this requirement of delegation by providing a two-layer encryption. Cloud will perform fine grained encryption on data while data owners will perform coarse grained encryption on data hosted in cloud. Performing two layer encryption can be an issue if the access control policies (ACP's) are not decomposed properly. We propose an efficient algorithm for decomposition of ACP's.

Index Terms:- Access Control, Broker, Cloud, Privacy, Owner, Policy Decomposition, User Identity Attributes.

1. Introduction

CLOUD is a informal expression used to describe a variety of different types of computing concepts connected through a real time communication network such as the Internet involving large number of computers. With the help of cloud computing, sharing data through a third-party cloud service provider is getting better in terms of usability and economic factors. However, one could not rely on such cloud providers for protection of data confidentiality. In fact, organizations using such technologies are concerned more about security and privacy. Data often contain sensitive information which should be protected as per various organizational policies and legal regulations. Encryption is commonly adopted approach but it is not sufficient alone as organization always want to enforce fine grained access control. This control comes in the form of user identity attributes which are security relevant properties of users. These access control systems are indicated as attribute based access control (ABAC) systems. Therefore, an important requirement is to support fine-grained access control, based on policies specified using identity attributes, over encrypted data.

As the third party servers are involved, the user identity attributes in access control policies may reveal privacy-sensitive information about users and organizations and leak confidential information about content. So the identity attributes need to be protected in order to maintain the confidentiality of data and privacy of users.

2. Literature Survey

An approach to support fine grained selective ABAC is to identify set of data items to which the same access control policy applies and then encrypt each such set with the same encryption key. The encrypted data is uploaded to cloud and each user is given the key according to the policies which allow users to access the data to which they are eligible.

Such approach addresses two requirements: (a) protecting data confidentiality from the cloud; (b) enforcing fine-grained access control policies with respect to the data users. A major issue in such an approach is represented by key management, as each user must be given the correct keys with respect to the access control policies that the user satisfies.

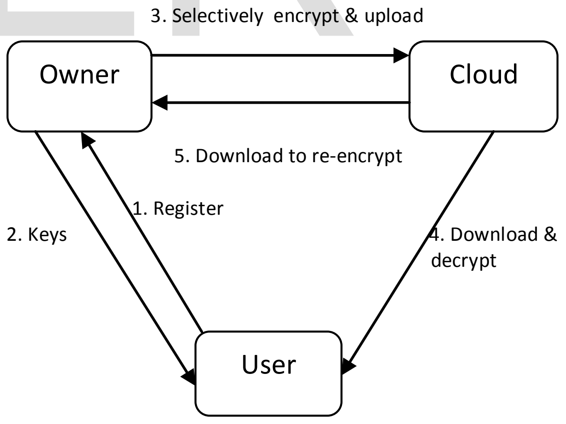


Fig. 1: Traditional Approach

Such an approach is termed as traditional approach.
Limitations of traditional approach are as follows:

- Owner suffers high communication and computation cost.
- Privacy of identity attributes of users are neglected.
- Inefficient in supporting fine grained ABAC policies.

To overcome some of the above limitations, different approaches were proposed which was based on broadcast key management schemes. These approaches were termed

as Single Layer Encryption (SLE) approaches. It required owner to perform fine grained encryption based on ACP's.
Access Control Policies (ACP's) defines the resources being protected and the rules that control access to them.
SLE also required owner to perform encryption but it assured privacy of users. Still as owner was involved in fine grained encryption based on ACP's, it incurred high communication and computation cost at the owner side. When user were added or removed, owner had to download the data and upload it again to the cloud by re-encrypting for maintaining forward secrecy and backward secrecy which proved inefficient when continuous users were added or revoked.

Forward secrecy requires that a user who left the group should not be able to access any future keys.
Backward secrecy requires that a newly joining user should not be able to access any old keys.

As shown in the figure SLE approach contains four entities, Owner, IdP, User, Cloud.

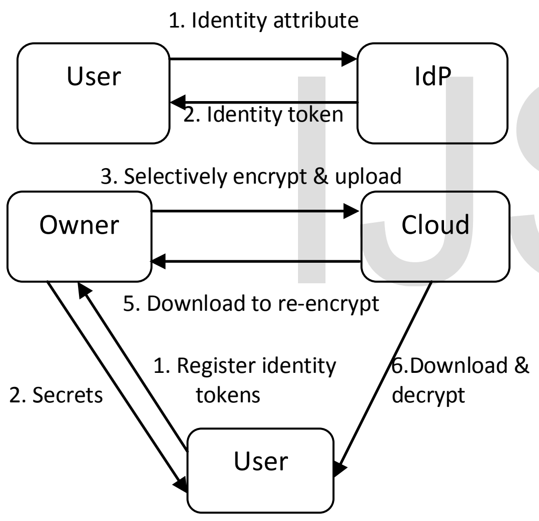


Fig. 2: Single Layer Encryption

- Owner, the data owner defines ACPs, and uploads encrypted data to the Cloud, the cloud storage service.
- Cloud hosts the encrypted data of the Owner.
- IdP, the identity provider, a trusted third party, issues identity tokens to users based on the attribute eattributes that users have. An identity token is a signed Pedersen commitment that binds the identity attribute value to a Usr while hiding it from others. There can be one or more certified IdPs. We assume that all IdPs issue identity tokens in the same format.
- Usr, the user, uses one or more identity tokens to gain access to the encrypted data hosted in the Cloud.

As shown in the figure, Owner enforce all the ACP's through selective encryption and uploads encrypted data to the untrusted Cloud.
The system goes through five phases:
Identity token issuance: Based on identity attributes IdP's issue identity tokens to the Users.
Identity token registration: Usrs register all their identity tokens to obtain secrets in order to later decrypt the data that they are allowed to access.
Data encryption and uploading: Based on the secrets issued and the ACPs, the Owner encrypts the data using the keys generated using the AB-GKM::KeyGen algorithm and uploads to the Cloud.
Data downloading and decryption: Usrs download encrypted data from the Cloud and decrypt using the key derived from the AB-GKM::KeyDer algorithm.
Encryption evolution management: Over time, either access control polices or user credentials may change. Further, already encrypted data may go through frequent updates. In such situations, it may be required to re-encrypt already encrypted data. The Owner alone is responsible to perform such re-encryptions. The Owner downloads all affected data from the Cloud, decrypts them and then follows the data encryption and upload step.

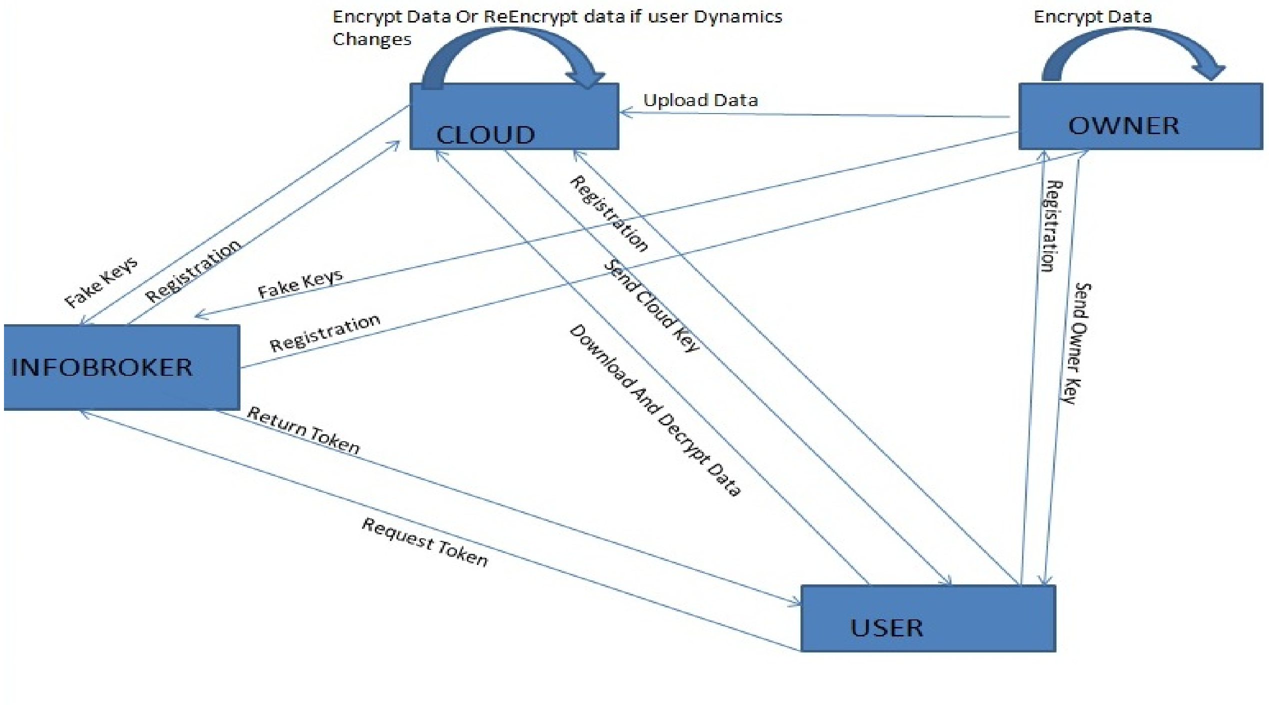
3. Proposed System

In this section we explain the proposed solution for limitations of the above systems explained.
In our proposed system we consider four entities: Owner, Cloud, Info-Broker, and User. Unlike SLE, here ACP's are enforced collectively by performing encryption twice on each item. Thus the load on owner side is reduced and also much of the enforcement of access control duties are delegated towards the cloud.
Handling of data updates and user dynamics can be done in a better way in this approach.
As shown in the figure above we consider four entities:

- Owner, the data owner defines ACPs, and uploads encrypted data to the Cloud, the cloud storage service.
- Cloud hosts the encrypted data of the Owner.
- Usr, the user, uses one or more identity tokens to gain access to the encrypted data hosted in the Cloud.
- Info-Broker, issues identity tokens to the users according to their identity attributes. Infobroker can be prevented from accessing the data in the cloud as he also knows the identity attributes of users.

The two layer encryption approach consist of six phases:
Identity Token Issuance: In these phase the user requests for the token from the infobroker and infobroker returns a token to the user. These token is nothing but a unique identifier provided to the user.

Policy decomposition: In these phase the owner decomposes each ACP into two sub ACP's . These decomposition lead to two ACP's such that owner enforces



SYSTEM ARCHITECTURE

minimum number of attribute to assure confidentiality from the cloud.

User Registration phase: Using the identity token provided by the infobroker the user has to register himself to cloud as well as to the owner. Based on the registration information the owner allocates the user two secrets . These secrets are used for encryption by the owner and the cloud. The secrets allotted must be such that there must minimum overhead towards owner.

Encryption phase: In this phase the data to be stored is encrypted first by the Owner using the owner ACP's. This layer of encryption is called as coarse grain encryption. And then second layer of encryption is done by the Cloud using the remaining subset of the ACP's called as Cloud ACP's. This layer of encryption is called as fine grained encryption. The ACP's are divided using the policy decomposition algorithm (discussed below).

The algorithm leads to the two subset of the ACP's:-

1. One which consists of owner confidentiality related information called as Owner ACP's used by the owner for encryption.

2. Another that consists of remaining subset of the ACP's called as Cloud ACP's used by the cloud for encryption.

Decryption phase: Users download encrypted data from the Cloud and decrypt the data using the derived keys. Users decrypt twice to first remove the encryption layer added by the Cloud and then by the Owner. As access control is enforced through encryption, Users can decrypt only those data for which they have valid secrets.

Encryption management phase: Suppose after some amount time the user dynamics changes thus the cloud performs a second layer encryption again on the encrypted file send by the owner using some different secrets and passes the keys to the valid users. Thus here owner does not have to re-encrypt the data which reduces the owner overhead to download and re-encrypt the data.

Broker management phase: Here if a infobroker tries to sign in and misuse the data then the infobroker is provided with the fake key's. Thus this would prevent our data from infobroker as well.

4. Policy Decomposition

In the SLE approach the owner has the biggest communication and computation overhead because he has to manage all the authorizations. If the owner only incrypts the data single time and all the access control is performed at the cloud then much of the owner's overhead will be reduced but the information exposure risk will be great , due to collusions between users and cloud the information security risk will be great.

An alternative approach is to decompose the policies to cloud and owner, in this approach both the key management and information exposer risk will be balanced. This approach will allow the owner to manage minimum number of attributes. In this approach owner manages the set attributes to which it belongs and the remaining set of attributes will be managed by cloud. The owner manages the minimum number of attributes in order to ensure the confidentiality of data from the cloud. In order to achieve consistency in this approach the owner rewrites the access control policies that the cloud should enforce such the conjunction of both the decomposed policies will form the original access control policy. Owner has to initially encrypt the data and upload the encrypted data on to the cloud. Therefore, in order to avoid re-encryption by the Owner, the data may have to be encrypted again by cloud to have two encryption layers.

Policy decomposition algorithm

Input: access control policies.
Output: decomposed policies.

Method:-
1. First ACP(owner) = null and ACP(cloud)= null.
2. Convert the given ACPi into DNF form to from an expression of attributes.
3. if (any conjunctive term appears into the expression) Then decompose that term into c1 and c2
Such that c1=ACP(owner) and c2=ACP(cloud). And c1 AND c2=c.
4. if (multiple conjuctive terms are appeared) Then decompose them seperately. As c1 and c2.
5. Stop.

Vigenere Algorithm:

This algorithm will be used for encryption and decryption at the owner side. It consists of matrix of alphabets with 26 rows and columns in which alphabets are written out 26 times in different rows, and in each row alphabets are cyclically shifted to the left compared to the previous alphabet. This matrix is known Vigenere square.

Vigenere Encryption:

Input: Text File
Output: Vigenere encrypted file.

Method:
1. Read file to be encrypted.
2. Choose a key.
3. Duplicate the key as many times so that the length of key matches length of the original text.
4. Search the cipher-text letter from Vigenere square using letters from text and key corresponding to row and column.
5. Encrypt each letter until the text ends.
6. Stop.

Vigenere Decryption:
Input: Reverse circle cipher decrypted file.
Output: Original text file**.**

Method:-
1. Read the decrypted reverse circle cipher decrypted file.
2. Duplicate the key which was used for encryption as long as the text file.
3. Search the row corresponding to the key.
4. Find the cipher-text letter in that row.
5. Column label corresponding to the cipher-text letter is the original letter from the text.
6. Repeat steps 3,4,5 till all the encrypted text is decrypted.
7. Stop.

Reverse Circle Cipher Algorithm:

This algorithm will be used for encryption and decryption at the cloud.

Reverse Circle Encryption:

Input: Vigenere Encrypted File.
Output: Reverse Circle Cipher Encrypted File.

Method:
1. Read the File to be encrypted.
2. Divide the complete file into parts (called tokens) the length of the token is predefined.
3. Rotate the tokens according to its index or any predefine value.
4. These rotated tokens are applied to the encryption function. Encryption function could be anything like adding each character with any predefined constant number.
5. Concatenate all the sequence of tokens and we get the encrypted file.

Reverse Circle Decryption:

Input: Reverse Circle Cipher Encrypted File.
Output: Reverse Circle Cipher Decrypted File.

Method:
1. Read the encrypted file.
2. Divide the complete file into parts (called tokens) the length of the token is predefined.
3. De-rotate the tokens according to its index or any predefine value.
4. These de-rotated tokens are applied to the decryption function. Decryption function could be anything like subtracting each character with any predefined constant number.
5. Concatenate all the sequence of tokens and we get the original file.

5. Analysis

In this section we will compare SLE and TLE approaches.

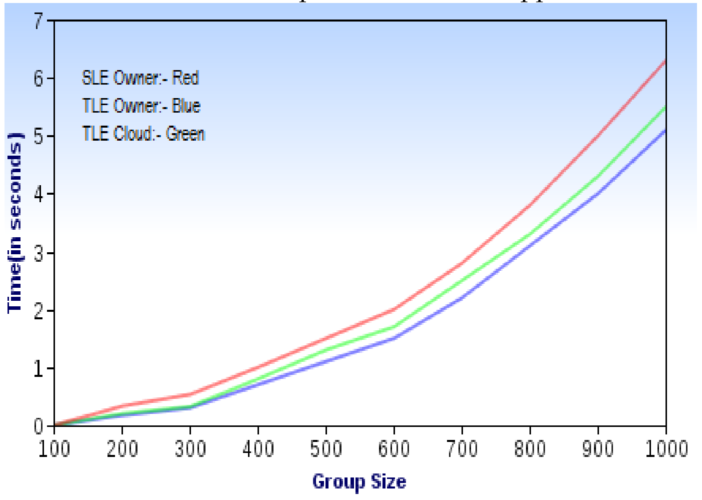


Fig. 4: Mean time to generate keys for two approaches

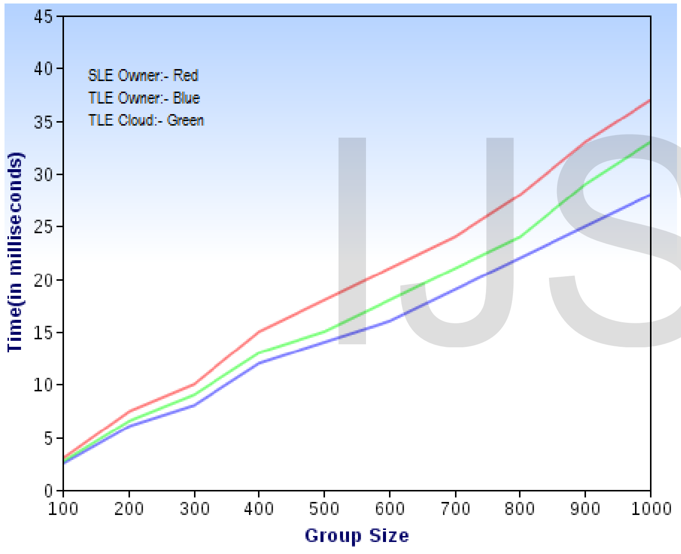


Fig. 5: Mean time to derive keys for the two approaches

As we know in SLE approach owner does fine grained encryption for enforcing all ACP's. Cloud acts only as repository storage for data. If user dynamics change, owner is responsible for re-encryption and updating of keys. Advantage of SLE approach is that ACP's remain hidden from the cloud. Also any user is unable to access the data that he is not allowed to access if he colludes with cloud. However the performance is degraded as owner suffers from high communication and computation cost. Further, it is unable to perform optimizations such as delayed ABGKM::ReKey or re-encryption as the Owner has to download, decrypt, re-encrypt and re-upload the data, which could considerably increase the response time if such optimizations are to be performed.
The TLE approach reduces the cost suffered by owner as it handles only the minimum attributes required for coarse grained encryption. Further if identity attributes are added or revoked owner does not have to re-encrypt the data. Its responsibility of the cloud to re-encrypt the data for enforcing ACP's. Thus TLE reduces computation and communication cost suffered by the owner. However the cloud learns some of the identity attributes of the users.
As shown in Figure 4 shows mean time required for generating keys using AB-GKM::KeyGen for SLE and TLE approaches. We will have number of attribute conditions 1000 and maximum number of attribute conditions per policy is 5. Thus as in figure we can see that execution time is larger in SLE than in TLE. Also execution time for TLE Owner is smaller than TLE Cloud as owner has to handle minimal set of attributes to perform coarse grained encryption. Same is the case in deriving keys for both approaches using AB-GKM::KeyDer.

6. Conclusion

Existing approaches to enforce ACPs on outsourced data using selective encryption require organizations to manage all keys and encryptions and upload the encrypted data to the remote storage. Such approaches incur high communication and computation cost for managing keys and re-encryptions. We proposed an approach based on two layer encryption to solve the problem by delegating much of the access control enforcement responsibilities to the Cloud while privacy of data due to colluding Users and Cloud. We also introduced a broker system instead of identity providers to prevent hacking the cloud using fake identity attributes. Decomposition of ACP's was a problem in this context so as to allow owner handle minimum number of attributes while allowing data to be hidden from the cloud. Based on the decomposed ACPs, we proposed approach for preserving privacy of data by delegating access control towards the cloud. Proposed approach is based on a privacy preserving attribute based key management scheme that protects the privacy of users while enforcing attribute based ACPs. As future work, we plan to investigate the alternative choices for the TLE approach further. We also plan to further reduce the Computational cost by exploiting partial relationships among ACPs.

7. Acknowledgment

Our sincere thanks to Siddhant College of Engineering for providing a strong platform to develop our skill and capabilities. We would like to thank our teachers, seniors who guided us in our project.

8. References

[1] M. Nabeel and E. Bertino, "Privacy preserving delegated access control in the storage as a service model," in *EEE International Conference on Information Reuse and Integration(IRI)*, 2012.
[2] M. Nabeel, E. Bertino, M. Kantarcioglu, and B. M. Thuraisingham, "Towards privacy preserving access control in the cloud," in *Proceedings of*

the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing, ser. Collaborate Com '11, 2011, pp. 172–180.

[3] M. Nabeel, N. Shang, and E. Bertino, "Privacy preserving policy based content sharing in public clouds," *IEEE Transactions on Knowledge and Data Engineering*, 2012.

[4] M. Nabeel and E. Bertino, "Towards attribute based group key management," in *Proceedings of the 18th ACM conference on Computer and communications security*, Chicago, Illinois, USA, 2011.

[5] M. Nabeel and E. Bertino, "Privacy Privacy Preserving Delegated Access Control in Public Clouds" *IEEE Transactions on Knowledge and Data Engineering*, 2012.

[6] Fengjun Li, Bo Luo, Peng Liu, Dongwon Lee, and Chao-Hsien Chu, "Enforcing Secure and Privacy-Preserving Information Brokering in Distributed Information Sharing," *IEEE Transactions On Information Forensics And Security*, VOL. 8, NO. 6, JUNE 2013.

[7] G. Miklau and D. Suciu, "Controlling access to published data using cryptography," in *VLDB '2003: Proceedings of the 29th international conference on Very large data bases. VLDB Endowment*, 2003, pp. 898–909.

[8] Mohamed Nabeel, Elisa Bertino , "Privacy-Preserving Fine-Grained Access Control in Public Clouds," *IEEE Computer Society Technical Committee on Data Engineering*, 2012.

[9] Fiat and M. Naor, "Broadcast encryption," in *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '93. London, UK: Springer-Verlag, 1994, pp. 480–491.