International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 01 | Jan -2017 www.irjet.net p-ISSN: 2395-0072

Secure Data Sharing in Cloud through Limiting Trust in Third Party/Server

Kalyanee Patil[1], S. D. Khatawkar[2], Amol Dange[3]

[1] *Student, Dept. of Computer Science and Engineering, Shivaji University Kolhapur, Maharashtra, India*
[2] *Professor, Dept. of Computer Science and Engineering, Shivaji University Kolhapur, Maharashtra, India*
[3] *Professor, Dept. of Computer Science and Engineering, Shivaji University Kolhapur, Maharashtra, India*

Abstract - *Cloud storage is a service of clouds that causes, organizations to switch from establishing in-house data storage systems to the cloud. The advances that proposed recently have given rise to the popularity and success of cloud computing. However, when outsourcing the data and business application to a third party causes the security and privacy issues to become a critical concern. Within an organization, data need to be shared among different users with different credentials. Secure sharing of data among a such group which causes an insider threat from valid or malicious user is an important research issue. Current approach provides a solution on aforementioned issue. Under this approach, the third party is an in charge of security related operations like encryption, decryption, key generation, access control, etc. But there may be a possibility that this third party may show malicious behavior and causes insider threat. A better approach should provide a solution which limits trust in third party while assuring data confidentiality. We propose an approach, based on two layers of encryption that addresses such requirement. Under our approach, the data owner performs a lower layer encryption, whereas the third party performs an upper layer encryption on top of the owner encrypted data. A challenging issue is how to maintain confidentiality of data. To implement it we shifted access control right distribution operation to the owner. The owner sends a key to valid users for encryption and decryption. This ensures only valid user will get an access to the data.*

Key Words: Cloud Computing, Confidentiality, Access control, Secure Sharing, Privacy, Security, etc.

1.INTRODUCTION

CLOUD computing is rapidly emerging due to the provisioning of elastic, flexible, and on-demand storage and computing services for customers. Cloud computing offers an effective way to reduce capital expenditure and operational expenditure. This economic benefit is a main cause of the cloud popularity. However, SECURITY and privacy represent major concerns in the adoption of cloud technologies for data storage. An approach to mitigate these concerns is the use of cryptography where data are usually encrypted before storing to the cloud [1]. Whereas cryptography assures the confidentiality of the data against the cloud, when the data are to be shared among a group, the cryptographic services need to be flexible enough to handle different users, exercise the access control, and manage the keys in an effective manner to safeguard data confidentiality. The data handling among a group has certain additional characteristics as opposed to two-party communication or the data handling belonging to a single user. The existing, departing, and newly joining group members can prove to be an insider threat violating data confidentiality and privacy.

While adopting a cloud for storage, the loss of control over data and computation raises many security concerns for organizations. The loss of control over data and the storage platform also motivates cloud customers to maintain the access control over data (individual data and the data shared among a group of users through the public cloud).The cloud customer encrypts the data before storing to the cloud , this ensures cloud doesn't learn any information about customer's data. The access rights are given to different users by distributing key used for encryption. However, this will result in excessive load over customers. By putting a third party in between customer and cloud and delegating all operational loads to a third party will help to lower load from the customer. But while doing so there is a possibility that third party may show malicious behavior. Hence there should be an approach to overcome this.

In this paper, we propose a methodology named Secure Data Sharing in Clouds through limiting trust in Third party/Server that deals with the aforementioned security. It helps to limit trust in third party/server. While delegating some operational load to a third party this approach ensures data confidentiality. For this the concept of two layer encryption is used where lower layer encryption is performed by the data owner and upper layer encryption is performed by third party. The owner provides the authority of file access to user by proving key used for lower layer encryption,

while encryption or decryption of the file. Hence, by retaining control over operations back to data owner this approach helps to preserve confidentiality [2].The departing member cannot decrypt the data on its own as he/she will not able to get a key used for lower layer encryption from data owner. Similarly, no frequent decryption and encryption are needed for new user inclusion and user's departure.

2. SIGNIFICANCE

Approach addressed in [3] relies completely on third party (CS) for security related operations like encryption, key generation, access control, decryption, etc., but there is a possibility that this third party shows the malicious behavior and cause an insider threat. This approach has presented a way to limit trust level in third party(CS).The lower layer encryption at owner ensures that third party will not get direct access to data. File access authority is given by the owner by distributing key used for lower layer encryption. Even if a third party provides file to any unauthorized users, they unable to access it as they will not able to get a key for decryption from file owner.

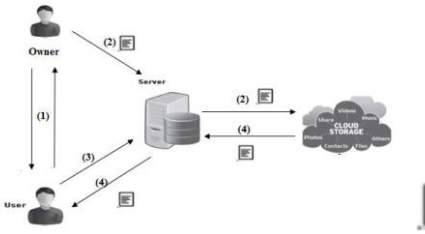
In addition, this approach causes less time consumption. In [3] for each user separate key shares are needed to be calculated during encryption and during decryption original key needs to be computed from shares. Also for new user inclusion separate key shares need to be calculated. In our approach only two keys require one for lower layer encryption and another for upper layer encryption. Hence time for key share generation during encryption, original key computation during decryption and generation of key shares for newly joining members get eliminated.

2. RELATED WORK

Approach addressed in [3] provided a way for secure sharing of data among users with different level of privilege. There is a third party in between customer and cloud who is responsible for performing security related operations like key management, encryption, decryption, and access control. Data is encrypted using a single symmetric key. Two different key shares are generated for each user. One share is given to the user and the other is kept by a third party named Cryptographic server. User with one share ensures security from the insider threat. However, this approach relies completely on third party, there is possibility that third party shows malicious behavior and cause an insider threat. As suggested in [2], to overcome the insider threat to cloud confidentiality one defense strategy is to retain control back to the owner. But it will cause excessive load over owner. As Suggested in [4], the cloud generates the public–private key pairs for all of the users and transmits the public keys to all of the participating users. Partial decryption is performed at the cloud. Due to the fact that key management and partial decryption are handled by the cloud, user revocation is easier to handle. However, the proposed scheme treats the public cloud both as a trusted and untrusted entity at the same time. From a security perspective, it is not recommended to shift the key generation process to the shared multitenant public cloud environment.

This paper presented an approach which helps to counter the aforementioned issue. We have used two layer encryption schemes as suggested in [5]. The lower layer encryption by data owner ensures that third party will not able to access owner's data. Authority to access data is given by data owner by transferring key used for lower layer encryption to the user. This ensures that third party will not able to give an access to any unauthorized user. Even if he does so that unauthorized user will not able to read it in the absence of key used for lower layer encryption. In this way, by retaining control over some operations this approach ensures cloud confidentiality.

3. POPOSED WORK



:- File

(1) User registration; (2) File Upload; (3) Download Request; (4) File Download

Fig -1: System architecture of Secure Data Sharing in the Cloud through limiting trust in Third party/server

Figure1 shows the basic architecture of "secure data sharing in the cloud through limiting trust in third party/ Server". This proposed system will work with four entities as follows: 1) owner; 2) user; 3) server and 4) cloud. The data owner first assigns a unique ID to each user of his files. The users then register with owner by providing his own password. The owner maintains information about each user in list-User List containing unique id and password. The information about users also sends to a third party for storage. While uploading file to cloud the file owner will perform lower layer encryption and submit the encrypted file. Server after receiving a file performs upper layer encryption on it. The encrypted data then subsequently get uploaded to the cloud for storage. The file owner assigns access right (read/write) on file to users. The list of users, their access right and other information like date from which access right is valid will get sent to the third party server .This information gets maintain there in the form of ACL. ACL is maintained for each file containing file id, user id, date, access right. The user who wishes to access the file sends a

download request to the third party server. The third party server receives the unique ID and password from the user, after authenticating the requesting user it downloads the data file from the cloud. The data file gets decrypted and sent back to the user. User after receiving file requests a key to the owner. The owner provides the key he has used to perform lower layer encryption. User decrypts a file with key he has received from the owner. For a newly joining member, the owner assigns a new unique id and user then register with owner. The owner will send the information regarding this user to the server to maintain it in an ACL. For a departing member, the record will deleted from all respective tables.

4. IMPLEMENTATION STRATEGIES

4.1 User Registration:

This operation is performed by the file owner. The owner of a file will first assign a unique ID (ID) to each user of his files. Then user registers with this unique id with owner by providing his own password (PassW). Figure 2 shows user registration operation.

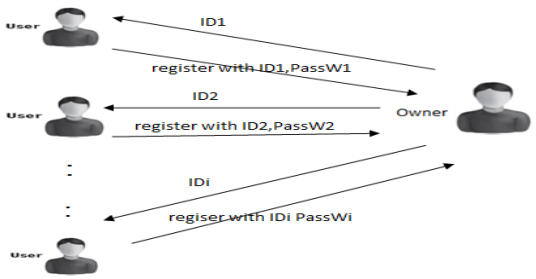


Fig -2: User Registration

4. 2 File Upload

4.2.1 Lower Layer Encryption

Algorithm:
R = {0, 1}^8 // R is any random number
Kf= Hf(R) // after applying hash function the R will completely randomized
F' = SKA(F, Kf)

For each file F separate key Kf is generated. Kf is generated in two steps. In the first step, the random number R of length 8 bits is derived by the file owner. In the next step, *R* is passed through a hash function that could be any hash function with a 128-bit output. The length of Kf is 128 bits. However the length of the key can be altered according to requirement. The file then encrypted with a symmetric encryption algorithm using generated key Kf. After successful encryption, encrypted file F' gets sent to the server.

4.2.2 Upper Layer Encryption:

Algorithm:
R = {0, 1} ^8 // R is any random number
K'f= Hf (R) // after applying hash function the R will completely randomize
C = SKA(F', K'f)

The third party server is responsible for upper layer encryption. For this it generates a key(K'f) of length 256 bits. The key K'f is generated in two steps. In the first step, the random number R of length 8 bits is derived by the third party server. In the next step, *R* is passed through a hash function that could be any hash function with a 256-bit output. The length of K'f is 256 bits. However the length of the key can be altered according to requirement. The file then encrypted with a symmetric encryption algorithm using generated key K'f. After successful encryption , encrypted file C gets sent to cloud for storage. Figure 3 shows a file upload operation which involves both lower layer encryption at owner and upper layer encryption on the server.

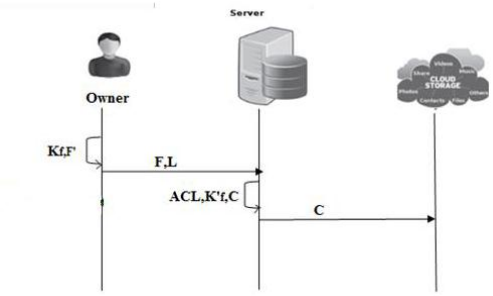


Fig -3: File Upload

4.3 File Download

This operation requires a decryption to be performed twice. First at the server and other by requesting user. Figure 4 shows download operation.

4.3.1 Decryption at the Third Party Server

Algorithm:
Get IDi and PassWi from the requesting user i.
Perform authentication and verify access right.
if authentication failed or access is not valid, then
Return the access denied message to the user.
else
Download C from the cloud.
F' = SKA(C, K'f)
send F' to the user.
end if

Whenever any user wants to download any file, he/she sends a request to the third party server. The third party

server after getting a request authenticate user and after successful authentication download requested file from the cloud, decrypts it and sends it to the user.

4.3.2 Decryption at the Requesting User

User after getting file from a third party server decrypts it. For this, the user requests a key to the file owner. File owner after performing successful authentication sends a key. The user then decrypts a file with a key he/she got from the owner.

Algorithm:
F=SKA(F',Kf)

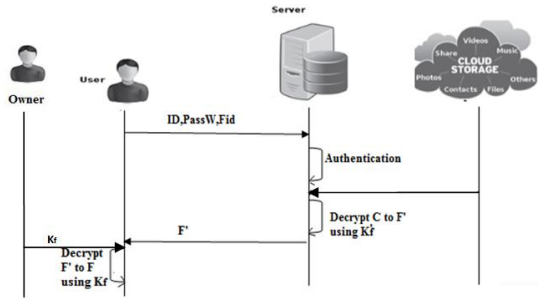


Fig -4: File Download

4.4 File Update

This operation requires lower layer encryption to be performed by the user who updates the file. For this, user requests for a key to file owner. File owner after performing authentication sends a key. The user then performs encryption on data and sends update request to the server. For encryption, user uses symmetric encryption algorithm used in algorithm 1. The server performs authentication and check whether the requesting user has update permission. After a successful authentication server performs upper layer encryption and upload file to the cloud. For encryption, server uses symmetric key encryption algorithm used in algorithm 2. Figure 5 shows update operation.

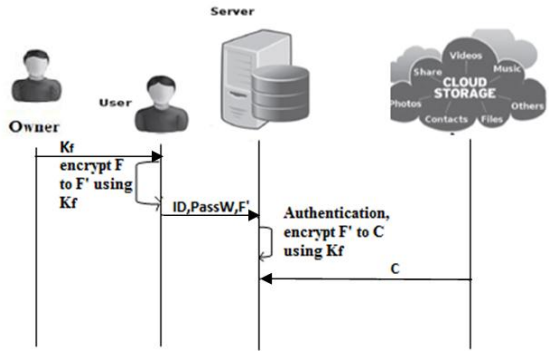


Fig -5: File Update

4.5 New User Inclusion and Departure

The inclusion of new user is made by registering user with owner by providing new id. This information stored by the owner in the user list. This information also gets sent to the third party server for storage. For newly joined member owner assigns access rights on files. This information also gets sent to the third party server to include it into ACL. The information contains file id, user id, date from which access right is valid, access right. The provision of date ensures backward access control. For departing member, the server is notified by the owner. The third party sever then deletes all records related to the user from ACL. The departing member will not able to decrypt data by its own. Hence this ensures forward access control.

5. PERFORMANCE EVALUATION

5.1 Experimental Setup

To evaluate the performance of the proposed methodology, we implemented the methodology in Eclipse using the JAVA framework. As discussed earlier, the proposed methodology consists of four entities, the cloud, the third party server, owner and the users. The jelastic cloud service which serves as the cloud server in our implementation. The third party and file owner are implemented as a server using apache tomcat. This implementation follows the MVC model. Where the view is implemented using jsp/html, control is nothing but servlets. The functionalities required by the user that is user level encryption and decryption are implemented as a TCPclient application that connects with the owner to receive the key. This communication is secured by SSL protocol. The functionality that required by the owner to provide key is implemented as a TCPserver application which always kept in running mode. The communication between the entities Owner and the third party is accomplished using URL class (java.net.URL). The class HTTPsURLConnection is used to open a secure connection to another entity. The scheme uses the SHA-1 hash function for generating keys at the owner and SHA-256 hash function for generating keys at third party server. The AES for encryption and decryption is used. All of the cryptographic operations like encryption and decryption are implemented using a javax.crypto.Cipher. The class java.security.MessageDigest is used to access all of the methods related to SHA. The hardware characteristics for the Third Party, Owner and the user client are shown in Table 1.

Table -1: Hardware Specification

CPU	Intel(R) CoreTM i3-3217U CPU 2 @ 1.80GHz
RAM	4 GB

Storage	100GB
OS	Windows 7 64 bits

5.2 Results

We evaluated the methodology on the basis of the total time consumed to upload/ download a file to/from the cloud. The total time is composed of the time from the time of submission of request to the CS to the point of time at which the file is uploaded/downloaded to/from the cloud. The following times are included in the total time:

1) the key computation time at owner and third party server;
2) the encryption/decryption time at owner/user and third party server;
3) the upload/download time;
4) the time of request and other related data submission to the CS and the cloud.

Fig. 6 shows the results for the upload time. All of the constituent times are represented by separate line graphs. The term "others" refers to the fourth constituent time discussed previously. In general, the time to upload the data increased with the increase in the file size. However, some marginal changes in time are due to network condition at that time. Hence, the file upload time was dependent on the network conditions. The key computation time is independent of file size and almost remained constant. The encryption time increased with the increase in the file size. Fig. 7 shows the results for the download operation involved in downloading the file from the cloud and the subsequent decryption processes. The trend of results is similar as in the case of a file upload. However, the times in decryption and the download are changed. There is no need to compute key during download procedure. Hence Key computation time is eliminated. We have compared our methodology with the scheme presented in [3]. The comparison is on the turnaround time for encryption and decryption. Table 2 shows the turnaround times for upload and Table 3 for download. These tables reveal that the SeDaSC methodology outperforms the other technique.

Table -2: Comparison of Turnaround time for File Upload

File Size (MB)	Encryption Time(Sec)	Key Computation Time(Sec)	Total Upload [3](Sec)	Total Upload (This System)(Sec)
0.5	0.101	0.001	0.94	0.384
1	0.240	0.001	1.24	0.376
10	2.268	0.001	6.43	3.390
50	6.589	0.001	9.01	8.220
100	11.371	0.001	17.37	13.791

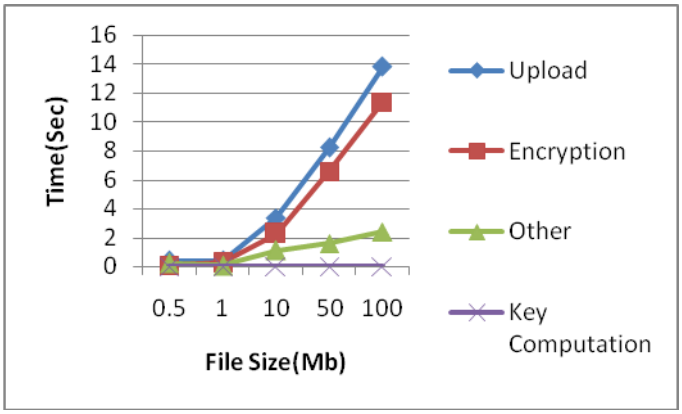


Fig -6: Performance of File Uploads

Table -3: Comparison of Turnaround time for File Download

File Size (MB)	Decryption Time(Sec)	Total Download [3] (Sec)	Total Download(This System)(Sec)
0.5	0.711	0.96	0.871
1	0.231	1.18	0.421
10	1.471	6.48	4.663
50	7.405	10.24	8.016
100	16.939	20.68	17.791

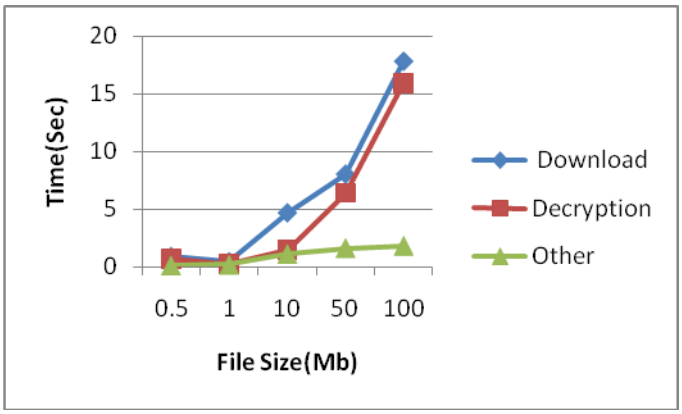


Fig -6: Performance of File Downloads

6. Limiting Trust in Third Party or Server

Approach addressed in this paper works on a future work of previous approach that is limit trust in third party (CS). The previous approach relies on third party for performing security related operations like encryption, decryption, key generation, access control, confidentiality management, etc. However, the third party is trusted one there may be possibility that it can lead to an insider threat. Hence there should be an approach which ensures data security by keeping control over some operations at data owner.

In the previous approach as owner transfer his data as it is to the third party/server. In this case the server can read that data and also can transfer this data to anyone. Unlike previous approach, in the proposed approach, encrypted file is handed over to the third party instead of the original file. This ensures third party will not get direct access to the owner's data. In the previous approach third party assigns permissions to each user by providing one key share. As third party has control over this operation, he/she can able to show any malicious user as an authorized by providing one key share. In the proposed approach owner assigns permissions to each user by providing key used for lower layer encryption. This ensures that only authorized users will get an access to the file. For example, suppose if third party provide files to any malicious user, that user will not able to decrypt them in the absence of key used for lower layer encryption.

7. CONCLUSIONS

We proposed a methodology for secure sharing of data among multiple users with different credentials. The proposed methodology provides data confidentiality, secure data sharing without reencryption, access control for malicious insiders, and forward and backward access control. Moreover, proposed methodology addressed problems in previous approach and provided its effective solution.

The approach provided here can be extended by strengthening accountability. Here users are differentiated by user id's and password. One can provide an approach which uses different way of ensuring accountability of users. One can provide an alternative approach for limiting trust in third party/server.

REFERENCES

[1] Cloud Security Alliance, "Security guidelines for critical areas of focus in cloud computing v3. 0," 2011.

[2] Zhifeng Xiao and Yang Xiao, Senior Member,IEEE, "Security and Privacy in Cloud Computing", IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 15, NO. 2, SECOND QUARTER 2013.

[3] Mazhar Ali, Student Member, IEEE, Revathi Dhamotharan Eraj Khan, Samee U. Khan, Senior Member, IEEE, Athanasios V. Vasilakos, Senior Member, IEEE, Keqin Li, Fellow, IEEE, and Albert Y. Zomaya, Fellow, IEEE,"SeDaSC: Secure Data Sharing in Clouds"IEEE SYSTEMS JOURNAL 2015.

[4] S. Seo, M. Nabeel, X. Ding, and E. Bertino, "An Efficient Certificateless Encryption for Secure Data Sharing in Public Clouds," IEEE Trans. Knowl. Data Eng., Vol. 26, no. 9, pp. 2107–2119, Sep. 2013.

[5] Mohamed Nabeel and Elisa Bertino, Fellow, IEEE,Privacy Preserving Delegated Access Control in Public Clouds", IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 26, NO. 9, SEPTEMBER 2014.