IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Volume-4, Special Issue-7, Dec 2016 ISSN: 2347-2693

Secure Data Sharing in Cloud Storage Using Key Aggregation Cryptography

(Secure Cloud Storage using Java Keystore)

Tulip Dutta*
Girijananda Chowdhury Institute of Management and Technology, Guwahati
Email: princetulip8@gmail.com

Amarjyoti Pathak
Girijananda Chowdhury Institute of Management and Technology, Guwahati
Email: amar_cse@gimt-guwahati.ac.in

Available online at: www.ijcseonline.org

***Abstract*— Secure Cloud Storage is an online platform for storing user data with sharing facilities among other users. The data is optionally encrypted before storing and decrypted on download by the system and the process is transparent to the end user. The platform also provides marking data private or public such that all public data are visible to end user and can perform search or download request to the owner. The data owner or user can audit all such request and grant download access to individual or multiple files accordingly. Each file on our system is encrypted using different AES key and the real challenge come when multiple file needs to be shared between users. This constraint is addressed by applying a concept known as Key Aggregation which states the generation of constant size key for a single or multiple files and thereby end user can easily decrypt the shared files.**

***Keywords*- Data Sharing, Key Aggregation, Java Keystore, Private Key Cryptography, Encryption, Decryption.**

I. Introduction

Data Sharing is an important functionality of a cloud storage. In our cloud storage, we are optionally giving features to end user to encrypt their data on uploading. It is also important to share these encrypted data securely among the users that are registered in our cloud storage. we are going to focus how we can make private key cryptography secure to share encrypted data among users. It is very important not to share a users' private key to other users. So we have to share the encrypted data to users without sending the private key of the data owner user.

II. Comparative Analysis

A. *Case 1*

We can share encrypted data in cloud storage by generating a single secret key for all the data. Following example will visualise it perfectly.

Let us consider, Alice encrypts her data 1,2,3,4 and uploads in the cloud storage as in figure 1.

Now, Alice wants to share the data 1 and 2 with her friend Bob. Alice sends the single secret key to Bob to decrypt the data as in figure 2.

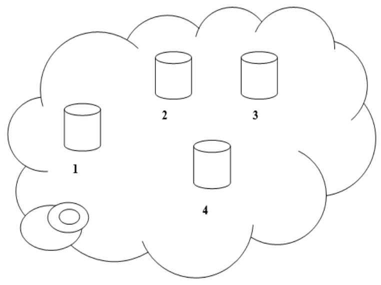

Fig 1.1: Cloud Storage having encrypted data 1,2,3,4.

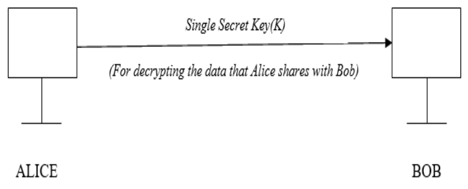

Fig 1.2: Data sharing Between Alice and Bob.

But using this single secret key, Bob can decrypt all the data that have uploaded by Alice i.e. Bob can decrypt the data 3 and 4 which are not yet shared by Alice.

B. *Case 2*

We can share the encrypted data in cloud storage by generating different keys for different data as shown in the figure 2.1.

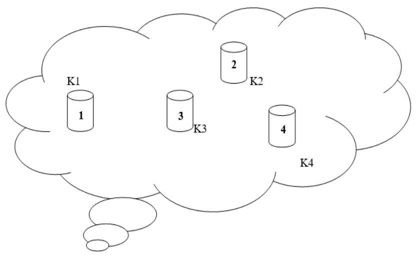

Fig 2.1: Cloud storage having Encrypted data 1,2,3,4 with different keys K1, K2, K3, K4.

Let us consider, Alice wants to share the encrypted data 1,2,3 with Bob. To decrypt the data, Alice has to send the keys K1, K2, K3 as shown in figure 2.2.

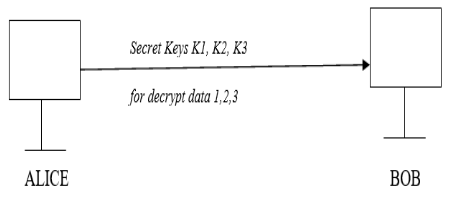

Fig 2.2: Data sharing Between Alice and Bob by sending different secret keys.

In this scenario, Alice has to send different secret keys repeatedly for different data that to be shared by Alice. This leads to wastage of memory and resources.

C. Case 3

We can therefore have the solution that is to encrypt the different data with different keys but decrypt data with a single secret key. Following example will illustrate the mechanism.

We encrypt the data 1,2,3,4,5,6 with different keys K1, K2, K3, K4, K5, K6 as shown in the figure 3.1.

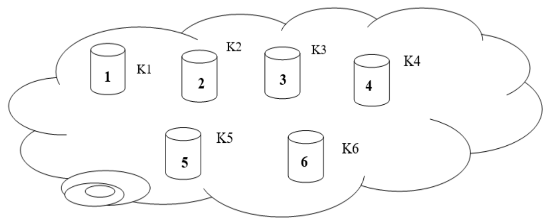

Fig 3.1: Encrypted data 1,2,3,4,5,6 with different keys K1, K2, K3, K4, K5, K6.

Suppose Alice want to share the encrypted data 1,2,3 with Bob. She will send Bob a single secret key, K123 which can decrypt the data 1,2,3 as shown in figure 3.2.

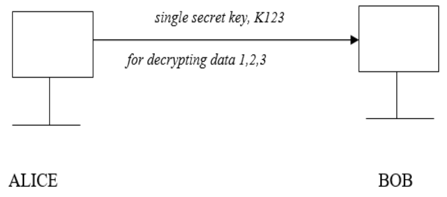

Fig 3.2: Data sharing Between Alice and Bob by sending single secret key, K123.

Again let us consider Alice want to share the encrypted data 1,2,3,4,5 with Denial. She will send Denial a single secret key, K12345 which can decrypt the data 1,2,3,4,5.

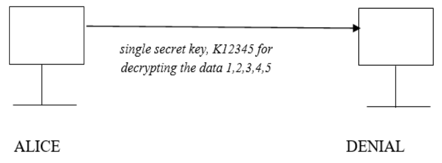

Fig 3.3: Data sharing Between Alice and Bob by sending single secret key, K12345.

In this scenario, as the number of data to be shared is increased, the size of the secret key is also increased. It causes more resource allocation in the cloud storage.

So we have to obtain a secret key that will be of constant size and able to decrypt the shared encrypted data.

III. Proposed System

In propose system, we are going to focus how we can make private key cryptography secure to share encrypted data among users. It is very important not to share a users' private key to other users. So we have to share the encrypted data to users without sending the private key of the data owner (user).

In our project, we generate different private keys for different encrypted data using AES-128 algorithm. The major advantage of AES is that it handles encryption of large data efficiently and the performance is much higher than compare to other encryption algorithms.

We have to generate an aggregate key which will be the secret decryption key in the sense that is allowed to decrypt multiple encrypted data without increasing its size. We introduce a new private key cryptography algorithm where the size of the secret key remains constant. The new private key cryptography algorithm is called as Key Aggregation cryptography. It provides us efficiently and securely data sharing mechanism in our cloud storage. This algorithm helps us to generate a constant aggregate key. By sending this aggregate key to users, the recipients can easily decrypt the data that are shared by the data owner. The aggregate key will be generated for only the shared data and other encrypted data except these shared data will not be able to decrypt by the users.

We are going to implement this scenario by using Java Keystore. Keystore is a specialized data structure for managing cryptographic keys. The Keystore is consists of a key entry contains the private key and also every entry has a unique alias name. However Key entries are protected by separate passwords.

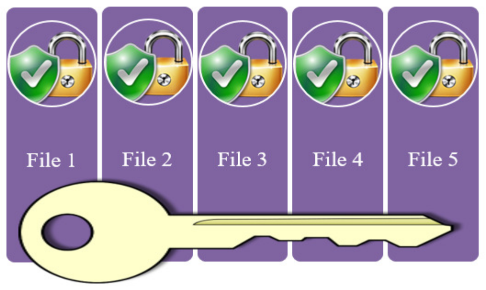

Fig 4.1: Architecture of a Keystore.

In our algorithm, the Keystore works as a locker box and each encrypted file is stored inside it as shown in figure 4.1.

When a user signs up on our system, an empty Keystore is generated for managing his/her encrypted keys and locked by the account's holder password. Once the Keystore is locked, it is encrypted by the password and we store the corresponding binary format data in database. The binary format is called as Binary Large Object (BLOB). A BLOB is a collection of binary data stored as a single entity in a database management system. Blobs are typically images, audio or other multimedia objects. The end users will not be able to access this binary data as it contains the binary data format of the encrypted data.

The primary Key Aggregation Algorithm is listed below:

Step 1: Generate AES Secret Key

$K_i \longleftarrow$ GenerateKey( )

Step 2: Encrypt Data

$Cipher_i \longleftarrow$ Encrypt($Data_i$, K_i)

Step 3: Load User KeyStore

KeyStore $\longleftarrow$ loadKeyStore(K_{pin})

Step 4: Save key

SaveKey($index_i$, K_i, $Pass_i$, KeyStore)

Step 5: Load Key

$K_i \longleftarrow$ GetKey($index_i$, $Pass_i$, KeyStore)

Step 6: Decrypt data

$Data_i \longleftarrow$ Decrypt($Cipher_i$, K_i)

Step 7: KeyStore extraction

$KeyStore_n \longleftarrow$ Extract(KeyStore, $index_n$)

Step 8: Generating a unique reference key

$K_{ref} \longleftarrow$ getUniqueKey( )

Step 9: Key aggregation generating constant size key

$K_{ag} \longleftarrow$ generateAggregateKey(userId, K_{ref})

Step 10: Lock the extracted KeyStore with the aggregate key

Lock($KeyStore_n$, K_{ag})

There will be two types of key aggregation scenario we are going to implement using our primary algorithm.

A. On requesting files and approving the request by the owner

1) User 1 Request for file 1, file 2
2) Owner Approve for file 1, file 2
3) Proceed to Step 7
4) Proceed to Step 8 and generate the aggregate key. First, we generate a reference key by using hash of random number, unique id and timestamp on which the owner approves the request.
5) Proceed to Step 9 and generate the aggregate key by using constant size key generator consisting the hash of user email and the reference key.
6) Request approval completed.
7) User 1 will initiate download file 1
8) System check for file 1 is encrypted or not
9) System ask for Aggregate Key
10) User 1 submit the Aggregate Key
11) System check Aggregate Key requested file 1 for user 1
12) On Matched, Get Index for file 1
13) Proceed to Step 5 by using the Index
14) Proceed to Step 6 and download the decrypted file 1.

B. On sending files to an another user of the system

1. Owner Select & Send two files- File 1, File 2
2. Proceed to Step 7
3. Proceed to Step 8 and generate the aggregate key. First, we generate a reference key by using hash of random number, unique id and timestamp on which the owner sends file 1 and file 2.
4. Proceed to Step 9 and we generate the aggregate key by using hash of recipient's email to which the owner sends the file 1 and file 2 and the reference key.
5. Recipient will initiate download file 1
6. System check if file 1 is encrypted or not
7. System ask for Aggregate Key
8. Recipient submit the Aggregate Key
9. System check Aggregate Key requested file 1 for Recipient
10. On Matched, Get Index for file 1
11. Proceed to Step 5 by using the Index
12. Proceed to Step 6 and download the decrypted file 1.

In case A, we are using hash of timestamp and user email for Aggregate key generation because there will be a situation for the owner after receiving two different request from a same user. Suppose the owner approves the both requests at the same time. In this situation if we don't use the hash of timestamp, although we use random number generator and a unique key generator then there may be generation of same aggregation keys for these two requests. In our system, we generate distinct and constant size aggregate keys. For each and every request that an owner receives, there will be a distinct and constant size aggregate key.

Similarly, in case B, we are using the hash of the email of the recipient and timestamp. If the owner sends some files to different users at the same time, then there will be different hash for different users. Our system doesn't allow to create two or more account by using same email. So the aggregate keys for different users are different.

TABLE I. Comparative analysis of previous papers

Paper	Description
1. Key-Aggregate crypto system for Scalable Data Sharing in Cloud Storage	A public key cryptographic algorithm is proposed in this paper.
2. A review on key aggregate crypto system for scalable data sharing in cloud storage	Public key cryptography is introduced.
3. Storing Shared Data on the Cloud via Security Mediator	Encrypted data are shared by using a security mediator.
4. Improving Privacy and Security in Multi-Authority Attribute-Based Encryption	A key-policy based on some attributes such as the country the user lives or the kind of data the user wants to share.

IV.Conclusion

In this paper, we addressed an important issue of secure data sharing on un trusted storage using private key cryptography. We investigated the challenges pertained to this problem and proposed data security in cloud storage using key aggregate cryptography.

In this project, proposed system is found to be very efficient for sharing the data on cloud storage. we are able to avoid and provide more security by using key aggregate algorithm with the help of Keystore and without sharing the private keys.

In future, we are expecting the implementation of file rating system that any user can rate a file on the cloud storage. We also try to work in implementing a chatting system that owner and users can interact more fluently.

References

[1]. Cheng-Kang Chu, Sherman S.M. Chow, Wen-Guey, Tzeng, Jianying Zhou and Robert H. Deng, "Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage" IEEE Transactions On Parallel and Distributed System, Vol 25, No. 2 February 2014.

[2]. Mewada, Shivlal, Arti Sharivastava, Pradeep Sharma, S. S. Gautam, and N. Purohit. "Performance Analysis of Encryption Algorithm in Cloud Computing.", International Journal of Computer Sciences and Engineering Vol.-3(2), PP(83-89) Feb 2015,

[3]. Ramakrishna Jadhav1, Snehal Nargundi,"A Review On Key-Aggregate Cryptosystem For Scalable Data Sharing In Cloud Storage", IJRET: International Journal of Research in Engineering and Technology, Volume: 03 Issue: 11 | Nov-2014.

[4]. Shivlal Mewada, Sharma Pradeep, Gautam S.S., "Exploration of Efficient Symmetric Algorithms", 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)", pp(663 – 666), March, 2016,

[5]. Wang, S. S. M. Chow, M. Li, and H. Li, "Storing Shared Data on the Cloud via Security Mediator," in International Conferenceon Distributed Computing Systems - ICDCS 2013. IEEE, 2013.

[6]. M.Chase and S. S. M. Chow, "Improving Privacy and Security in Multi-Authority Attribute-Based Encryption," in ACM Conferenceon Computer and communications security Pp 121-130.

[7]. Anjali Nigam, Vineet Singh, "Securing Data Transmission in Cloud using Encryption Algorithms", International Journal of Computer Sciences and Engineering, Volume-04, Issue-06, Page No (21-25), Jun -2016, E-ISSN: 2347-2693

[8]. Shivlal Mewada, Sharma Pradeep, Gautam S.S., "Classification of Efficient Symmetric Key Cryptography Algorithms", International Journal of Computer Science and Information Security (IJCSIS) USA, Vol. 14, No. 2, pp (105-110), Feb 2016

[9]. Sadiya Shakil and Vineet Singh, "Security of Personal Data on Internet of Things Using AES", International Journal of Computer Sciences and Engineering, Volume-04, Issue-06, Page No (35-39), Jun -2016, E-ISSN: 2347-2693

[10]. R. Durga Prasad, R.N.D.S.S Kiran, Andey Krishnaji, "Advanced Data Encryption/Decryption using Multi Codes for One Character", International Journal of Computer Sciences and Engineering, Volume-03, Issue-09, Page No (34-38), Sep -2015, E-ISSN: 2347-2693

[11]. Shivlal, P Sharma, S.S Gautam, "Exploration of Efficient Symmetric AES Algorithm", 2016 Symposium on Colossal Data Analysis and Networking (CDAN), pp(1-5), Mar, 2016. Doi: 10.1109/CDAN.2016.7570921

Author Profile

Mr. Tulip Dutta has done his BE in Computer Science & Engineering from NITS MIRZA under Gauhati University and completed M.Tech from GIMT Guwahati under ASTU. His research interest is on Cloud Computing and Cryptography.

Mr. Amarjyoti Pathak has completed his B.E. from VTU, Karnataka and M.Tech in Information Technology from Tezpur central University, Napaam. He is currently working as an Assistant Professor in the department of Computer Science & Engineering, GIMT Guwahati. He has more than 6 years of experience in academic as well as in research. His research interest is on system and network security.