IFERP
connecting engineers... developing research

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 5, Issue 3, March 2018

Secured File Sharing In Cloud Computing Using Cryptography

[1] Ahalya Mary J, [2] Sundeep Sharma, [3] Sukrit Sarkar, [4] Devarti Mahakalkar, [5] Riya Roy
[1] Assistant Professor(O.G), SRMIST, [2][3][4][5] IV year B. Tech CSE, SRMIST

Abstract: - **The main objective of this paper is to explain the sharing of data in cloud., we explain how data can be stored and shared in an efficient, secure and convenient way using a system which incorporates Key-Aggregated Cryptosystem to produce cipher texts of a constant size. By using a compact key, which is made by combining a set of secret keys, we can share our files with others or store in a very limited secure storage.**

***Keywords*: Cryptosystem, Data Sharing, Secret Key.**

I. INTRODUCTION

The characteristics of intrinsic information sharing and low maintenance provide a more robust utilization of resources. In cloud computing, cloud service suppliers provide associate degree abstraction of infinite cupboard space for shoppers to host information. It will facilitate shoppers scale back their money overhead of information management by migrating the native management system into cloud servers. However, security issues become the most constraint as we have a tendency to currently source the storage of information that is probably sensitive, to cloud suppliers. To preserve information privacy, a typical approach is to inscribe information files before the shoppers transfer the encrypted information into the cloud. Sadly, it's tough to style a secure and economical information sharing theme, particularly for dynamic teams within the cloud. We present a cryptographical storage system that allows secure information sharing on devious servers supported the techniques that dividing files into file groups and encrypting every file group with a file-block key. When a user is revoked from the group, he is no longer eligible to access the group files. The file-block keys need not be updated again and hence the system has no key distribution overhead.

II. EXISTING MODEL

The existing system only enables the user to store their information in a limited cloud space and not necessarily equips them with the luxury of secure file sharing where their data would be safe from being accessed by someone they don't want to. Also, Cloud storage will not make any sense if users end up downloading the encrypted data and then decrypt them to share that with others. Accessing the data directly from the server is preferable. However, finding a secure way to do so is not easy. Usage of a symmetric key algorithm should help the receiver decrypt the original message. [1] [2].

A. DISADVANTAGES

1) Data can be stolen by hackers and cloud providers in the open channel
2) The secret keys used are relatively expensive
3) With the increase in the number of decryption keys to be shared, the cost and complexity also increase.
4) Transmission and storage of secret keys and decryption keys are expensive.

III. PROPOSED SYSTEM

The idea in the paper is to generate a powerful decryption mechanism. Multiple cipher texts are decrypted keeping the size constant. Public key encryption is taken care by AES algorithm. The key-aggregate cryptosystem (KAC) is used to encrypt messages using a public and an identifier. The cipher texts can be furthered categorized into different classes. Alternatively the secret keys of different classes can he extracted by using a master secret key. The sizes of all the keys in the KAC schemes are kept constant. The maintenance of linear size is required and only a part is required at a time to be fetched from cloud storage. Previously, the same result could be achieved by using a constant size decryption key, but the classes did not perform any pre-defined hierarchical relationship. In our work, we eliminate the requirement of a class relationship, thus making it flexible [4].

A. ADVANTAGES

1) The delegation of decryption can be efficiently implemented with the aggregate key, which is only of fixed size.
2) The number of cipher text classes is large.
3) It is easy to key management for encryption and decryption.

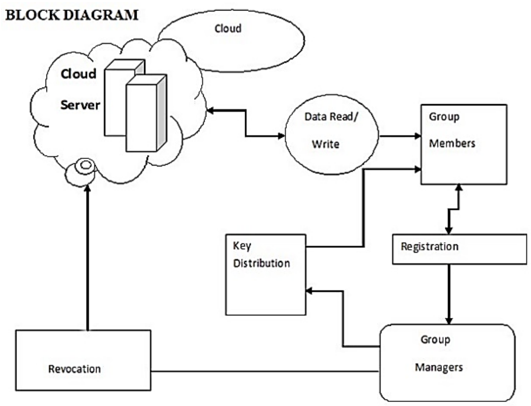

Fig. 1 Block Diagram

IV. SYSTEM ARCHITECTURE

The system architecture in Fig.2 depicts the process of sharing files in the group. The admin creates a group and a master key. The users are then added after their registration is accepted by the admin. After the group is created the users can share their files by using a request-response model. They, of course, need their key for pairing up while making the download request. All the activities in the group are traced by the admin.

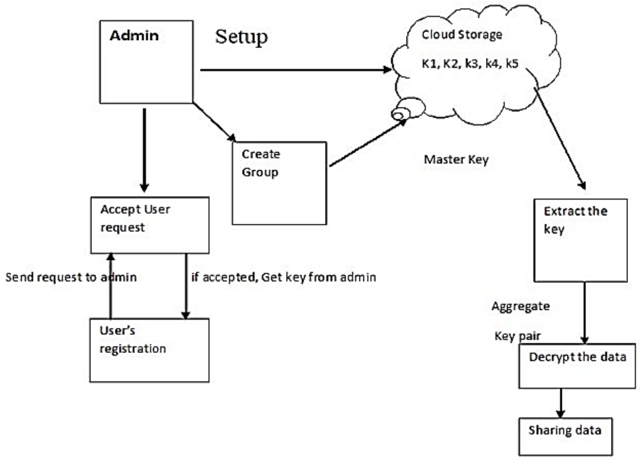

Fig.2 System Architecture

V. HARDWARE AND SOFTWARE SPECIFICATION

A. Software Requirement

1. Language-Java (JDK 1.7), JSP: It allows you to easily create web content that has both static and dynamic components.
2. OS-Windows 7 Ultimate 32-bit : It is required for the optimal performance of the system.
3. Database-MYSQL 5.0,SQLYOG: It is used for making your own Emulation Private Server.
4. Glassfish Server : It delivers a flexible, lightweight, and production-ready Java EE 6 application server.
5. NetBeans IDE 7.1.2 : It is an open-source Integrated Development Environment to create desktop, web, and mobile applications.

B. Hardware Requirement

1. 1 GB RAM: System should have at least 1 GB of RAM.
2. 80 GB Hard Disk: Memory is needed to store the files being shared.
3. Above 2GHz Processor: At least 2GHz processor is needed for fast processing.
4. Data Card: It is used for data operations (storage, transfer, transformation, input, and output).

VI. MODULES

A. User Registration:

Group Manager selects a random number using rand () function. This is issued an ID to the user. A user is added to the group user list. Public key granted to the user which acts as a group signature & decryption.

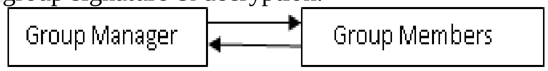

Fig.3 Registration (Key Distribution)

B. User Revocation

User Revocation is done by group manager based on a request. After a user is revoked the publicly available revocation list is updated.
Confidentiality of the existing user from the revoked user is protected by updating the revocation list each day.

Group Manager
Cloud

Fig.4 Revocation

C. File Generation and Deletions

The revocation list is requested from the cloud by using the group identity ID. The data in the cloud is verified against the ID requesting for the same. The data is now accessible and can now be deleted by group manager or the contact owner.

D. File Access and Traceability:

A user is authenticated by their unique signature (ID). The authentication process facilitates tracking, parameter checking, generating logs, accessibility control. During data dispute scenarios tracing can be performed by group manager to identify the data owner.

VII. CONCLUSION

In this paper, we design Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage. Maintaining the secrecy of the identity becomes important when it comes to the cloud. Using Mona, a user can easily share data with others in the group without revealing identity privacy to the cloud. User revocation and user additional becomes extremely simple in Mona. But precisely, a public revocation list, without updating the private keys of the remaining users, can add new users directly to decrypt files stored in the cloud before their participation, thus helping in an efficient user revocation. Moreover, the storage overhead and the encryption computation cost are constant. Analyzing, we have found that the scheme that we have proposed satisfies the necessary security requirements along with proper cryptographic storage system which can help to share files securely on trusted servers; named Plutus. As lockbox keys encrypt file box keys, files can hence be divided into file groups to be shared by data owners to others through delivering the corresponding lockbox key, only after successful encryption of the group using a file block key. This, however, causes a key distribution problem when it comes to sharing files in a bulk and updating of file block key to be distributed again when a user is revoked.

VIII. ACKNOWLEDGEMENT

We would like to thank SRM Institute of Science and Technology for giving us the opportunity to work on this project.

REFERENCES

[1] K. Kent and M. Souppaya. (1992). Guide to Computer Security Log Management, NIST Special Publication 800-92 [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf

[2] U.S. Department of Health and Human Services. (2011, Sep.). HIPAA—General Information [Online]. Available: https://www.cms.gov/ hipaageninfo

[3] PCI Security Standards Council. (2006, Sep.) Payment Card Industry (PCI) Data Security Standard—Security Audit Procedures Version 1.1 [Online]. Available: https://www.pcisecuritystandards.org/pdfs/pci−audit−procedures−v1-1.pdf

[4] Sarbanes-Oxley Act 2002. (2002, Sep.). A Guide to the Sarbanes-Oxley Act [Online]. Available: http://www.soxlaw.com/

[5] C. Lonvick, The BSD Syslog Protocol, Request for Comment RFC 3164, Internet Engineering Task Force, Network Working Group, Aug. 2001.