CS6030- Big Data, IoT and Cloud Computing

Data Sharing in Clouds Using Symmetric Cryptography Algorithm

A Cloud Security Project

Names

Akhil Kattepogu

Mohan Siva Krishna Konakanchi

Nikhil Masur

Submitting to

Prof. Dr. Ajay Gupta

Abstract

Now a days IT industry people willing to store their sensitive data in cloud storage server. Because previous days the business people want to store their data in any server then they need to purchase that server and themselves need to be maintaining as well as they unable to store their data for long time by lack of hard disk availability and unable to data sharing in globally. By limitation of these features business people are ready to store their data in cloud, but lack of security they are applying cryptography technique on their data before uploading into cloud storage. Many existing asymmetric cryptography algorithm can be used for encrypting owners data but they all are effecting by time consuming for encryption and storing the multiple keys in any storage area which is chances to lose of security. To overcome these limitation, the present system can use symmetric cryptography algorithm with EX-OR operation which is reduce the time consuming by using of 128-bit key size for encrypting the data and the key should be spilt and share to authorized users for accessing the cloud data with securely.

Introduction

In cloud computing, which is provides huge space for storing the data by cloud user or data owners, the cloud user wants to t provides security and access control over their data. So that cloud user can use cryptography techniques for encrypting, decrypting their sensitive data as well as they can apply access controlling like which data user can read data and writing or editing data as well as the data owner can restrict to data user not to accessing cloud storage data when revoked by data owner. For achieving security in this system, it can implement single key crypto system which is generate a random key with 128 bit key size and making cipher text from plain text and store in cloud storage, later the encryption key can be shared by among the group users for decrypting the cipher text by data user. But there is a problem to lose of security by single key distributions among data users, if any malicious users get the knowledge of encryption key then it can easily compromised and breaching the security, so using of key splitting operation this system can become strong and flexible data sharing securely among data users even the key is compromised by any malicious users.

Project Statement

The scope of project is a reducing time consuming by previous asymmetric crypto techniques while encryption and key management when new or old data user adding and removing in group as well as which is reduced re-encryption and key updating if the data is shared among group, so the main objective of the project which provides single key crypto system with key splitting operation for secure group data sharing in cloud storage.

Project Goals

Problem Statement:

In previous system used asymmetric cryptography algorithm which is time consuming for encryption and symmetric cryptography algorithm also used in many existing system, but using of single key for encryption and decryption there is a chances to breaching the security if any malicious users know the key, so that the current system can resolve the above limitations by AES with EX-OR operations.

Proposed System

In proposed system we implementing AES with EX-OR operation for limitation of previous system. In this system data owner can select the file and data users who want to access the data and send to cryptography server then this server can encrypt the file with single key and split that key, one of key is send to data users email and second one is store in data cloud storage, When data users wants to access the cloud data they first enter half key then remaining key can get from cloud storage if both keys are concatenation with EX-OR operation then it generates original key, with help of that key they can decrypt the data. Even if any malicious users know the data users key they cannot decrypt that encrypted data. Here the original key doesn't store any storage area, So that it can provide strong security for cloud storage.

Project Implementation

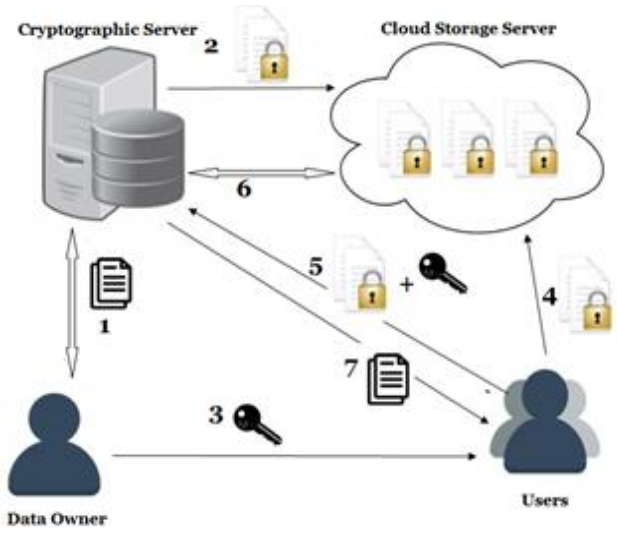

Fig.1 System Model

The Fig.1 illustrates the system process flow by the below steps:

Step-1:

The data owner can register and login in this system later he/she can browse the file and select access permissions to data users like READ or WRITE and the data owner can send uploading request to cryptography server along with file details.

Step-2:

The cryptographic server can login and view the file details which is sent by data owner and generating the random number and convert it hashing code with help of SHA-256 algorithm then it can become 128 bit secret key for encryption done by cryptography server with AES algorithm and store into AWS storage space which name was cloud storage server.

Step-3:

This system can split the secret key with EX-OR operation and share the half encryption key to authorized data users to their respective email ids and remaining half key can store in cloud server with their user identity.

Step-4 & 5:

The data user can login and they accessing the storage files from AWS cloud server which is shared by data owner. The data user can send decryption request to cryptography server along with half key and encrypted file details.

Step-6&7:

The cryptography server can get decryption request by data user then it can first apply the EX-OR operation with data user half key and remaining fetch from cloud server then it can get original secret key and decrypt the cipher

text with AES algorithm and forwarded to data user which can downloaded.

Algorithm

Key Generation:

R=Random ().nextInt(10000)

K=H(R)

Encryption:

F= file data

CT=AES_ENCRYPT (F, K)

CT Store in AWS Server

For each user *uid* in from user list

K1= Random ().nextInt(10000)

K2=K ⊕K1

Send to K2 to user *uid*

K1 Store in AWS

Decryption:

CT=Ciphertext

Get K2 from user request

Get K1 from storage server

K= K2 ⊕K1

F=AES_DECRYPT (CT, K)

F forward *to request user*

Technology Used:

JDK1.8

Html, CSS, JSP

Tomcat Server 8.0

EditPlus IDE

SQLyog

AWS RDS

AWS Elastic Beanstalk

Challenges

In this system challenging as generating random key and converting hashing code with uses of SHA-256 algorithm and for splitting the key it has used EX-OR operation and after making cipher text with AES algorithm, the keys are distributed among group user's email ids.

Conclusion

This system implemented Data Sharing in Clouds Using Symmetric Cryptography Algorithm which can provide data security and data access control over the cloud storage server by data owner and for reducing time consuming and complex of asymmetric technique it has worked by AES-128 with EX-OR operation with single key crypto system and cryptographic server worked as mediator between data owner and cloud storage server for key generation, encryption and decryption process.

Future Work

The cryptographic server is working as mediator between data owner and cloud storage server so that any attacker can compromise it then there is chances to misuses that server, so data owner should be authenticate itself to cryptographic server with **proof of knowledge** algorithm. This algorithm can work like prover and verifier, data owner can generate proof value and send to cryptographic server it can verify that proof then if it is valid then it can receive the

data owner file data but it is not valid then it will not get owner file.

References

[1] A. Abbas and S. U. Khan, "A review on the State-of-the-art privacy preserving approaches in e-health clouds," IEEE J. Biomed. Health Informat., vol. 18, no. 1, pp. 1431–1441, Jul. 2014.

[2] K. Alhamazani et al., "An overview of the commercial cloud monitoring tools: Research dimensions, design issues, state-of-the-art," Computing, DOI: 10.1007/s00607-014-0398-5, 2014, to be published.

[3] R. Chandramouli, M. Iorga, S. Chokhani, Cryptographic key management issues and challenges in cloud services, in: Secure Cloud Computing, Springer, New York, 2014, pp. 1–30.

[4] Y. Chen, J. D. Tygar, and W. Tzeng, "Secure group key management using unidirectional proxy re-encryption schemes," in Proc. IEEE INFOCOM, pp. 1952–1960.

[5] Y. Chen and W. Tzeng, "Efficient and provably-secure group key management scheme using key derivation," in Proc. IEEE 11th Int. Conf. TrustCom, 2012, pp. 295–302.

Screenshots

Fig.2 Application Home Page

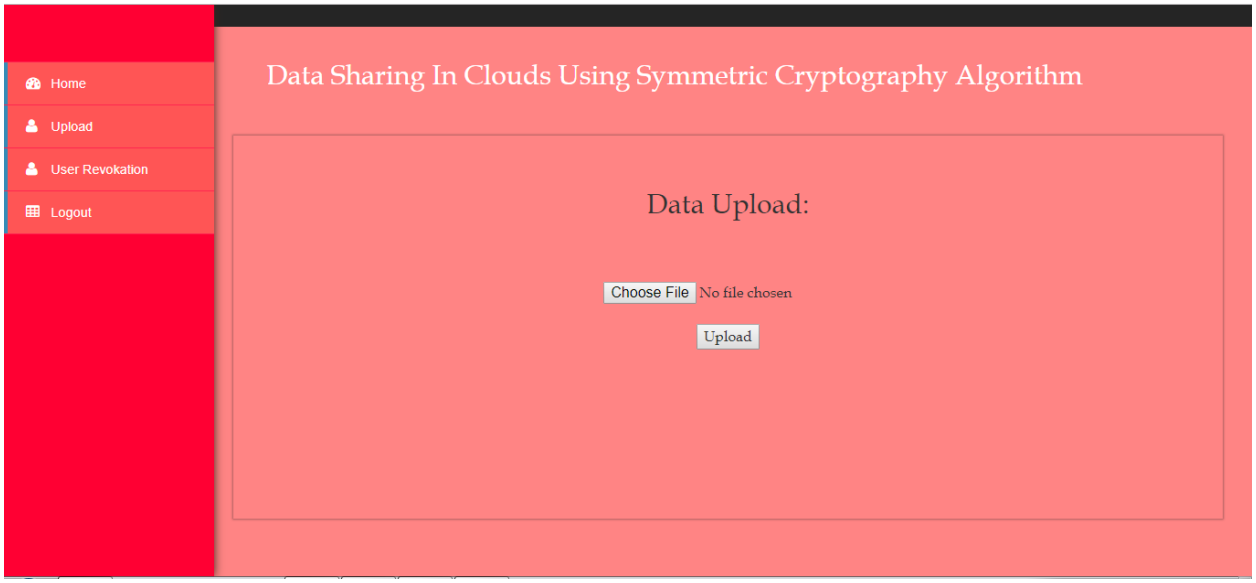

Fig.3 Data File Uploading

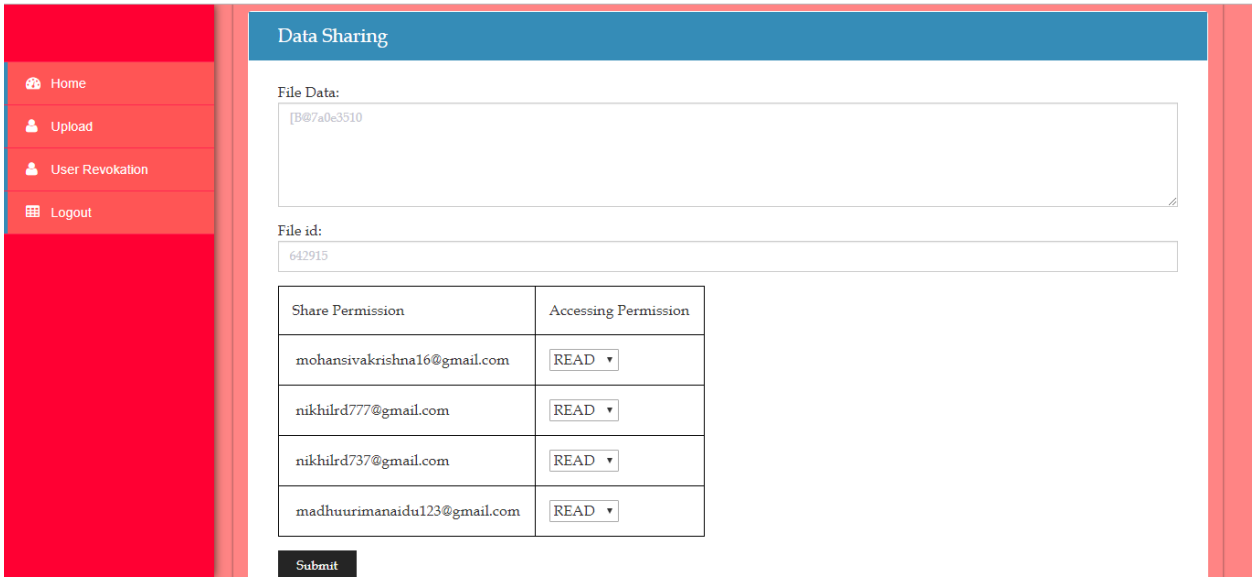

Fig.4 File Data Access Control

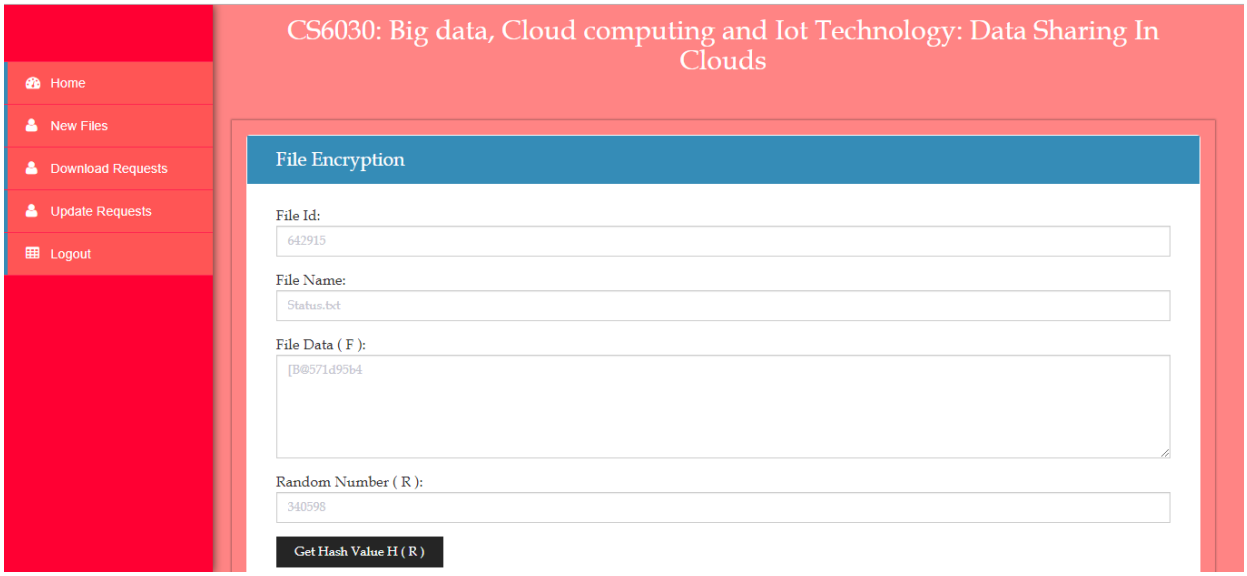

Fig.5 Random Key generation

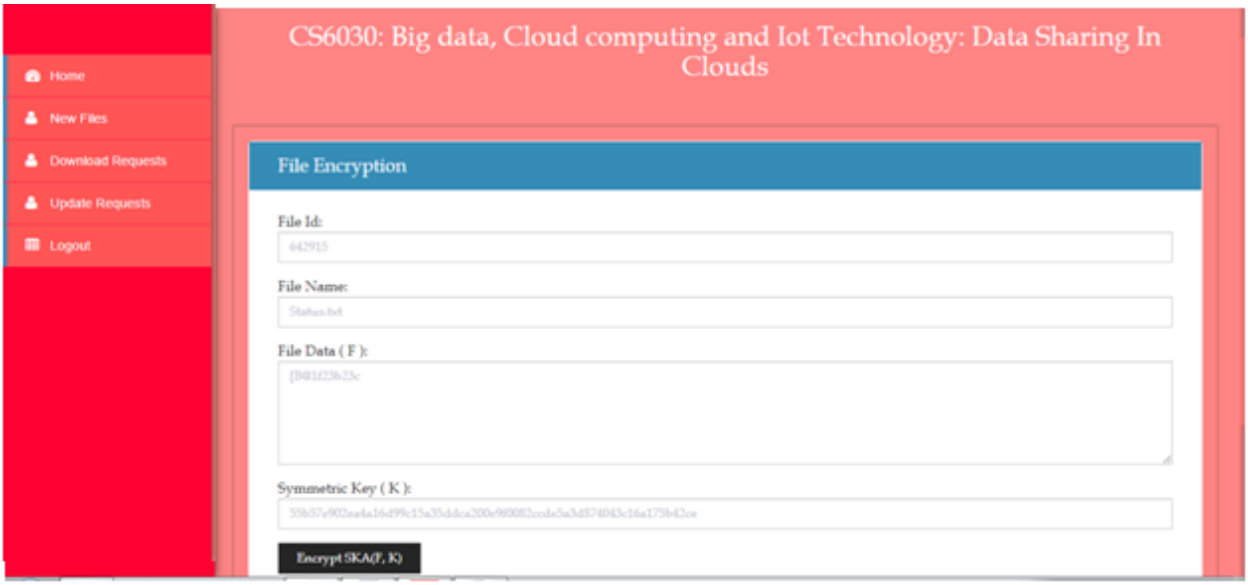

Fig. 6 Generating Hashing key

Fig.7 File data encryption

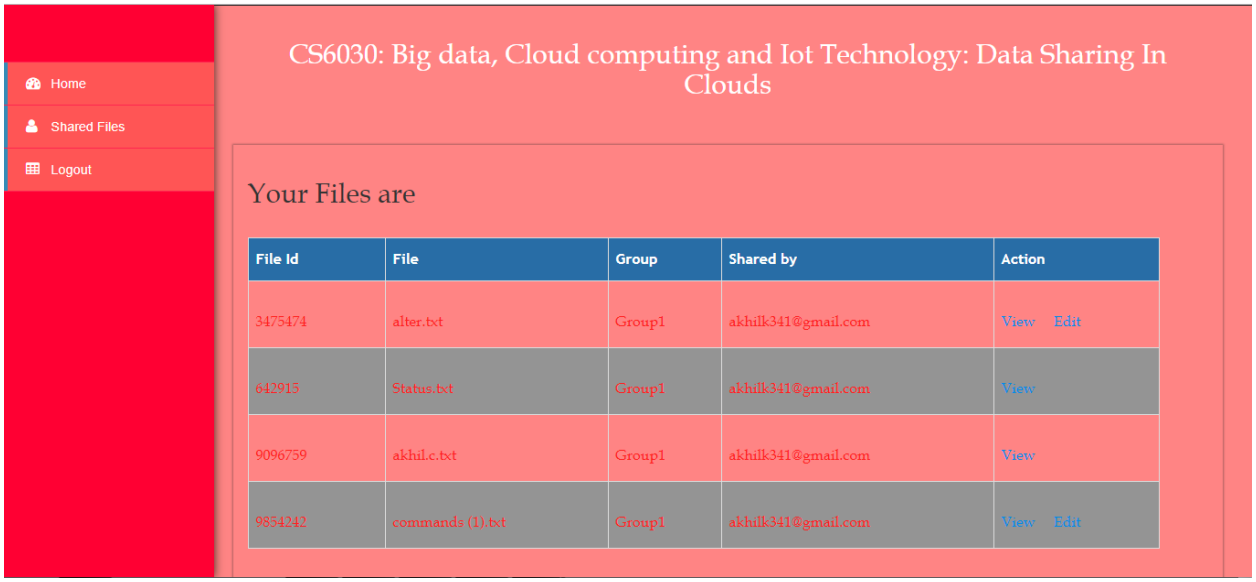

Fig.8 Data Owner Shared Files

Fig.9 File download request

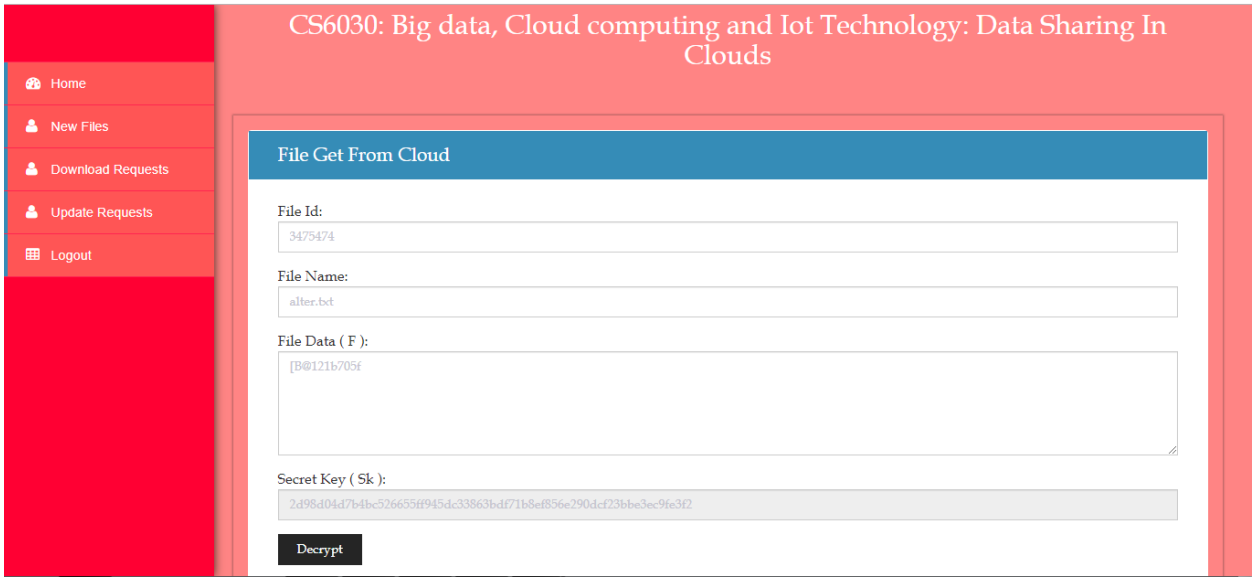

Fig.10 File decryption data

AWS Services:

RDS (MySql):

In this service we can create database for storing the application data and generate data base endpoint for connecting MySql data base from our application. Amazon RDS customer, if you have never created a DB instance before, or if you are creating a DB instance in an AWS Region you have not used before, you are most likely on the EC2-VPC platform and have a default VPC.

Elastic Beanstalk:

AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java and servers such as Apache Tomcat. The users simply upload code and Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, auto-scaling to application health monitoring.

SYSTEM CONFIGURATION:

Hardware requirements:

Processer	:	Any Update Processer
Ram	:	Min 1 GB
Hard Disk	:	Min 100 GB

Software requirements:

Operating System	:	Windows family

Technology	:	Java (1.7/1.8)
Front-End Technologies	:	Html, Html-5, JavaScript, CSS.
Web Server	:	Tomcat 7/8
Database (Back – End)	:	My SQL5.5
IDE	:	NotePad++