SECURE DATA SHARING SYSTEM IN A CLOUD

CS - 490 Master's Project

Safi Ur Rahman Mohammed

Computer Science Master's Project Report

Northeastern Illinois University

Secure Data Sharing System in a Cloud

Submitted by Safi Ur Rahman Mohammed

In partial fulfillment of the requirements for the degree of Master of Science in Computer Science - Fall 2018

Approved by:

Primary Advisor: Dr. Xiwei Wang _____ Date: _____

Co-Advisor: Dr. Graciela Perera _____ Date: _____

Co-Advisor: Dr. Rachel Trana _____ Date: _____

TABLE OF CONTENTS

ABSTRACT

Cloud computing has emerged as a convenient platform for data sharing in recent years. However, one of the main security challenges faced in cloud computing is the secure sharing of data. As the data contained may be valuable, to prevent users from directly sourcing the data to and from the cloud server, it is necessary to control a user's access through cryptological means. However, access control itself is not efficient, as the user can still have access to previously shared data. To be efficient, there should be a mechanism that removes the user from the system and also removes access from the previously and subsequently shared data.

To combat these disadvantages, we propose a notion called Revocable-Storage Identity-Based Encryption (RS-IBE). RS-IBE features a mechanism that enables a sender to append a certain ID value (sessionID) and use it to encrypt to the ciphertext such that the receiver can decrypt the ciphertext only under the condition that the file is not revoked at that time period. This can be done by requiring the cloud server to re-encrypt its shared data based on the user's current revocation status. As a result, RS-IBE provides access control, as well as security for previously and subsequently shared data.

By merging the functionalities of file revocation and ciphertext update simultaneously, the RS-IBE scheme has advantages over the current system in terms of security, functionality and efficiency.

CHAPTER I: INTRODUCTION

1.1 Objective:

The reliance of consumers on Cloud storage services has increased in recent years. With Cloud computing coming to the fore as a convenient, cheap and efficient alternative to traditional methods of storing data such as on physical devices, there is an increased need to secure and monitor how data is shared on it. There are various challenges in determining the security of a cloud service. Firstly, outsourcing data to the cloud server implies that data is out control of the users. This may cause users' hesitation since the outsourced data usually contains valuable and sensitive information. Secondly, data sharing is often implemented in an open and hostile environment, and any server could become a target of attacks. Even worse, the cloud server may reveal users' data for illegal profit by itself. Thirdly, when a user's authorization gets expired, he/she should no longer possess the privilege of accessing the previously and subsequently shared data. Therefore, while outsourcing data to cloud server, users also want to control access to these data such that only those currently authorized users can share the outsourced data.

A natural solution to conquer the aforementioned problem is to use cryptographically enforced access control such as identity-based encryption (IBE). Furthermore, to overcome the above security threats, such kind of identity-based access control placed on the shared data should meet the following security goals[1]:

• Data confidentiality: The plaintext of the shared data stored in the cloud server should be protected from unauthorized users. In addition, the cloud server, which is supposed to be honest but curious, should also be deterred from knowing plaintext of the shared data.

• Backward secrecy: Backward secrecy states that, when a user's authorization or user's secret key is compromised, he/she should be prevented from accessing the plaintext of the subsequently shared data that are still encrypted under his/her identity.

• Forward secrecy: Forward secrecy means that, when a user's authority or secret key is compromised, he/she should be prevented from accessing the plaintext of the shared data that can be previously accessed by him/her.

The specific problem addressed in this project is how to construct a fundamental identity-based cryptographical tool to achieve the above security goals. We also note that there exist other security issues that are equally important for a practical system of data sharing, such as the authenticity and availability of the shared data.

1.2 Problem definition:

There is a security issue in Identity-Based Encryption systems and to avoid it, efficient revocation suggested that users renew their private period. Only the PKG's (Private Key Generator) public key and the receiver's identity are needed to encrypt, and there is no way to communicate to the senders that an identity has been revoked, such a mechanism to regularly update users' private keys seems to be the only viable solution to the revocation problem. This means that all users, regardless of whether their keys have been exposed or not, should regularly get in contact with the PKG, prove their identity and get new private keys. The PKG must be online for all such transactions, and a secure channel must be established between the PKG and each user to transmit the private key[2].

Taking scalability of IBE deployment into account, we observe that for a very large number of users this may cause a bottleneck issue[2]. We note that alternatively, to avoid the need for interaction and a secure channel, the PKG may encrypt the new keys of non-revoked users under their identities and the previous time period, and send the cipher texts to these users (or post them online). With this approach, for every non-revoked user in the system, the PKG is required to perform one key generation and one encryption operation per key update[4].

We note that this solution, just as the original suggestion, requires the PKG to do work linear in the number of users, and does not scale well as the number of users grow.

The non-revocable data sharing system can provide confidentiality and backward secrecy. Furthermore, the method of decrypting and re-encrypting all the shared data can ensure forward secrecy.

However, this brings new challenges. Note that the process of decrypt-then-reencrypt necessarily involves users' secret key information, which makes the overall data sharing system vulnerable to new attacks. In general, the use of secret key should be limited to only usual decryption, and it is inadvisable to update the cipher text periodically by using secret key. There is also the question of efficiency, as this process brings great communication and computation cost, and thus is cumbersome and undesirable for cloud users with low capacity of computation and storage.

1.3 Existing System:

Identity Based Encryption (IBE) is a recent and effective approach to the problem of encryption key management. IBE can use any string as a public key, enabling data to be protected without the need for certificates. Protection is provided by a key server that controls the generation of private decryption keys[8]. By separating authentication and

authorization from private key generation through the key server, permissions to generate keys can be controlled dynamically.

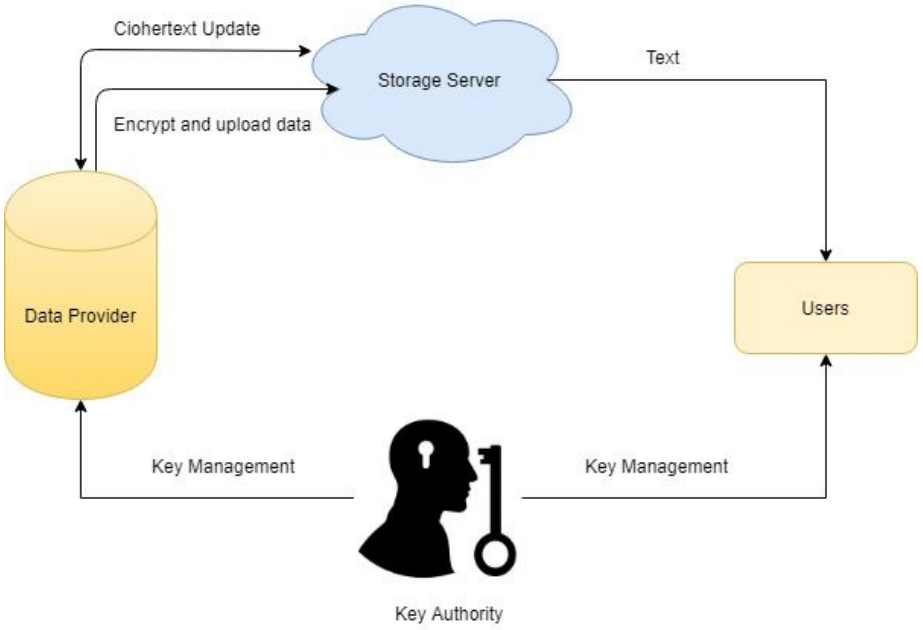


Fig: IBE Representation

Identity-based systems allow any user to generate a public key from a known identity value such as an ASCII string. The keys are generated by a trusted third party, called the Private Key Generator (PKG). The encryption/decryption procedure is as follows:

1.3.1 IBE Encryption/Decryption Process:

- The Private Key Generator (PKG) publishes a master public key and retains the master private key.
- User computes a public key corresponding to the ID value by combining the master public key with identity value.
- User obtains private key by contacting PKG, which uses master private key to generate the private key for ID value. To decrypt or sign messages, this private key obtained from the PKG is used.

1.3.2 Drawbacks of Existing System:

If a Private Key Generator (PKG) is compromised , all messages protected over the entire lifetime of the public-private key pair used by that server are also compromised. Since PKG generates private keys for users, it may decrypt and/or sign any message without authorization. A secure channel between a user and the Private Key Generator (PKG) is required for transmitting the private key on joining the system. It is important to observe that

users that hold accounts with the PKG must be able to authenticate themselves. In principle, this may be achieved through username, password or through public key pairs managed on smart cards.

1.4 Proposed System

In the proposed system , we used a concept called revocable-storage identity-based encryption (RS-IBE) for building a cost-effective data sharing system that fulfills the three security goals. The security goals are data confidentiality, forward secrecy and backward secrecy. Consequently, the user cannot access both the previously and subsequently shared data. To this end, we propose a notion called revocable-storage identity-based encryption (RS-IBE), implementing a symmetric block cipher (AES), which can provide the forward/backward security of ciphertext by introducing the functionalities of user revocation and ciphertext update simultaneously. As AES encrypts one block at a time, we applied our mentioned scheme to an existing IBE scheme to develop a block-wise deniable RS-IBE scheme. The technique to convert the encryption scheme from composite order groups to prime order groups can be used to pair the time difference of the bilinear operation of both groups to achieve a better computational performance.

1.5 Advantages of Proposed System

One method to avoid problems of an IBE system is to require the cloud server to directly re-encrypt the cipher text of the shared data. In addition, the technique of proxy re-encryption can also be used to conquer the problem of efficiency. However, this may introduce cipher text extension, namely, the size of the cipher text of the shared data is linear in the number of times the shared data have been updated[8]. RS-IBE also provides data confidentiality, forward and backward security.

CHAPTER II: LITERATURE REVIEW

2.1 Cryptography and its techniques

Cryptography is a method of storing and transmitting data in a particular form so that only those for whom it is intended can read and process it. In today's technologically-advanced world, cryptography is most often associated with scrambling plaintext (ordinary text, sometimes referred to as cleartext) into ciphertext, a process called **encryption**, and then back again which is known as **decryption**.

Modern cryptography concerns itself with the following four objectives[7]:

1) **Confidentiality**: The information cannot be understood by anyone for whom it was unintended.

2) **Integrity**: The information cannot be altered in storage or transit between sender and intended receiver without the alteration being detected.

3) **Non-repudiation**: The creator/sender of the information cannot deny at a later stage his or her intentions in the creation or transmission of the information.

4) **Authentication:** The sender and receiver can confirm each other's identity and the origin/destination of the information.

A **cipher** (or *cypher*) is a pair of algorithms that create the encryption and the reversing decryption. The detailed operation of a cipher is controlled both by the algorithm and in each instance by a "**key**".

The key is a secret (ideally known only to the communicants), usually a short string of characters, which is needed to decrypt the ciphertext.

Plaintext vs. Ciphertext:

Plaintext and ciphertext are typically opposites of each other. Plaintext is any information before it has been encrypted. Ciphertext is the output information of an encryption cipher.

Many encryption systems carry many layers of encryption, in which the ciphertext output becomes the plaintext input to another encryption layer.

The process of decryption takes ciphertext and transforms it back into the original plaintext.

2.2 Symmetric vs Asymmetric Encryption Systems

Symmetric Encryption	Asymmetric Encryption
Symmetric encryption uses only one key for encryption and decryption.	Asymmetric encryption uses 2 keys: a "Public" key for encryption and a "Private" key for decryption.
It is a relatively simple technique as one key is used for both operations.	The use of separate keys for encryption and decryption makes it a complex process.
Due to its simplistic nature, encryption and decryption can be carried out pretty quickly.*	Its higher complexity makes it slower.*
Algorithms Used[9]: ● RC4 ● AES ● DES ● 3DES	Algorithms Used[9]: ● RSA ● Diffie-Helman ● ECC

*= Application/Use-specific. Some Asymmetric algorithms are faster for different no. of inputs or application-specific use.

2.3 Algorithms used by RS-IBE:

Setup(1λ, T,N)*:*
The setup algorithm takes as input the security parameter λ, the time bound T and the maximum number of system users N, and it outputs the public parameter PP and the master secret key MSK,associated with the initial revocation list RL=**Φ**and state st[5].

• **PKGen**(PP,MSK, ID)*:*
The private key generation algorithm takes as input PP, MSK and an identity ID ∈I, and it generates a private key SKID for ID and an updated state st[6].

• **KeyUpdate**(PP,MSK,RL, t, st)*:*
The key update algorithm takes as input PP,MSK, the current revocation list RL,the key update time t ≤ T and the state st, it outputs the key update KUt[6].

• **DKGen**(PP, SKID,KUt)*:*
The decryption key generation algorithm takes as input PP, SKID and KUt, and it generates a decryption key DKID for ID with time period t or a symbol ⊥ to illustrate that ID has been previously revoked[6].

• **Encrypt**(PP, ID, t,M)*:*
The encryption algorithm takes as input PP, an identity ID, a time period $t \leq T$, and a message $M \in M$ to be encrypted, and outputs a ciphertext CTID,t[6].

• **CTUpdate**(PP,CTID,t, t′)*:*
The ciphertext update algorithm takes as input PP, CTID,tand a new time period $t' \geq t$, and it outputs an updated ciphertext CTID,t′ [6].

• **Decrypt**(PP,CTID,t,DKID,t′)*:*
The decryption algorithm takes as input PP, CTID, t, DKID, t′, and it recovers the encrypted message M or a distinguished symbol $\perp$ indicating that CTID is an invalid ciphertext[6].

• **Revoke**(PP, ID,RL, t, st)*:* The revocation algorithm takes as input PP, an identity $ID \in I$ to be revoked, the current revocation list RL, a state stand revocation time period $t \leq T$, and it updates RL to a new one[6].

CHAPTER III: METHODOLOGY

3.1 Algorithms

3.1.1 AES(Advanced Encryption Standard):

Input: Message or file/cipher text

Output: cipher text/original message or File

In our project encryption process takes place as the file is being uploaded to the cloud. The more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). It is found at least six time faster than triple DES[7].

A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow[7].

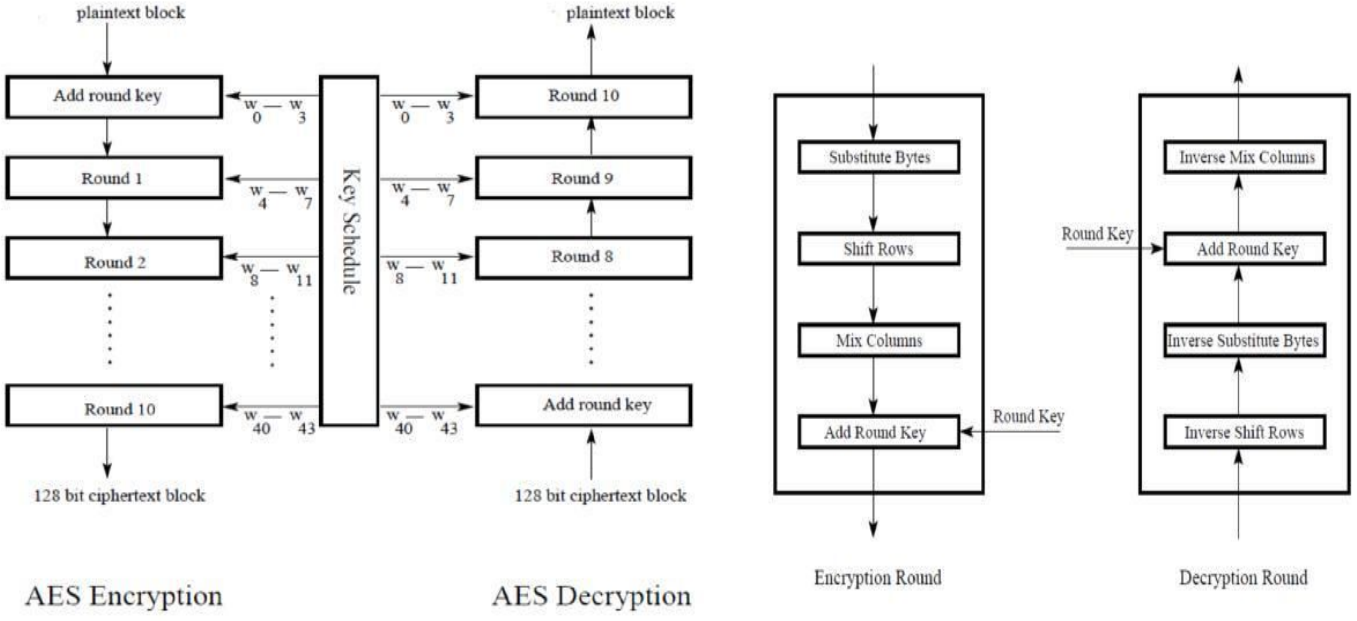


Fig 3.1.1 AES Encryption and Decryption

The features of AES are as follows −

- Symmetric key block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java

3.1.2. Why AES?

AES is an algorithm that was developed by the National Institute of Standards and Technology (NIST) as a successor to DES. It is a symmetric key algorithm, must be a block cipher encryption algorithm capable of handling 128 bit blocks, using keys sized at 128, 192, or 256 bits.

AES is one of the most widely used and trusted encryption standards in the world. In fact, the U.S. Government has used AES as its default encryption algorithm since 2003 to protect its classified information. AES is also the first publicly accessible and open cipher to be approved by the NSA for protecting top secret information. Symmetric ciphers must use the same key for encryption and decryption, so the sender and user must know and use the same key. The United States Government employs AES-128 to protect classified documents upto the “SECRET” level, and has designated use of AES-192 and AES-256 to protect classified documents up to the “TOP-SECRET” level[7].

AES has proven to be a reliable security cipher, as the only practical attacks against AES are side-channel attacks based on weak implementation or key management of AES-encryption based products. A research paper in 2011 published by Bogdanov, Khovratovich and Rechberger was successful in recovering AES keys using a technique called a “Biclique attack”. However because of its high computational complexity, even this attack doesn’t threaten the practical use of AES. Therefore, due to its reliability, efficiency and trustworthiness, AES was chosen as the encryption standard for this project. In present day cryptography, AES is widely adopted and supported in both hardware and software. Additionally, AES has built-in flexibility of key length, which allows a degree of ‘future-proofing’ against progress in the ability to perform exhaustive key searches.

However, just as for DES, the AES security is assured only if it is correctly implemented and good key management is employed.

3.1.3 Key Generation

Input: ID of owner and user

Output: Secret key

Base64 algorithm is designed to encode any binary data, stream of bytes, into a stream of 64-printable characters. In which, the binary data is transformed to ASCII text, which can be transported in email without problems. On the recipient's end, the data is decoded and the original file is rebuilt.

The Base64 encoding process is to:

Step 1: Divide the input bytes stream into blocks of 3 bytes.

Step 2: Divide 24 bits of each 3-byte block into 4 groups of 6 bits.

Step 3: Map each group of 6 bits to 1 printable character, based on the 6-bit value using the Base64 character set map.

Step 4: If the last 3-byte block has only 1 byte of input data, pad 2 bytes of zero (\x0000). After encoding it as a normal block, override the last 2 characters with 2 equal signs (==), so the decoding process knows 2 bytes of zero were padded.

Step 5: If the last 3-byte block has only 2 bytes of input data, pad 1 byte of zero (\x00). After encoding it as a normal block, override the last 1 character with 1 equal signs (=), so the decoding process knows 1 byte of zero was padded.

Step 6: Carriage return (\r) and new line (\n) are inserted into the output character stream. They will be ignored by the decoding process.

3.1.4 Revocation

Revocation is the act of recall or annulment. It is the reversal of an act, the recalling of a grant or privilege, or the making void of some deed previously existing.

As the data user is no longer part of the organization,he/she should not be able to access or download the files uploaded by data owner. Data owner can thus revocate the files i.e. A new copy of a existing file which is downloaded by user is created in the cloud, new key is generated, cipher is updated and then the old copy is deleted from the cloud.

3.2 System requirements

3.2.1 Hardware Specifications

- Processor : Any Processor above 500MHz
- RAM : 2GB
- Hard Disk : 20GB
- Input Device : Standard Keyboard and Mouse**.**

3.2.2 Software Specifications

- Operating System : Windows XP/7/8/10
- Language : Java, HTML, JSP
- IDE : Eclipse Oxygen
- Database : MySQL
- Development Kit : JDK 1.6

CHAPTER IV: PROJECT CONCEPTS & DEVELOPMENT

4.1 System Concepts

Concepts:

Data User/Owner:
The data user / owner can upload their file to the Cloud Server. They are able to encrypt & upload the data file and being able to set the access privileges for each file. They can also request access to files uploaded by other users.

Cloud Server:
The cloud is used for data storage for the owners that can share their encrypted data with responsibly authorized users who download those encrypted files and then decrypt them.

Key Distribution Center:
The key distribution center is used for storing our verification parameters and is responsible for secret key generation for specific files and their users. It is also responsible for maintaining the correct keys at all times.

Secured Credentials:
Application users are required to register into the system first. After registering they will be able to access and perform whatever their role-privileges allow them to do. The two roles here are the data user and the data administrator.

Encrypt / Decrypt Module:
When the user uploads the file, the encryption system will be activated and will encrypt the uploaded file into ciphertext format before sending it to the cloud. We use AES algorithm for encryption/decryption and the owners will convert normal files into encrypted documents with a dynamically generated secret key. The key can be sent to users for allowing them to decrypt and download the file in plaintext.

4.2 Project Development:

The project is a web application which implements Java, JSP(Java Server Pages) and SQL. The list of Java classes, servlets and JSP web pages involved in this project's construction are pictured in the figure below:

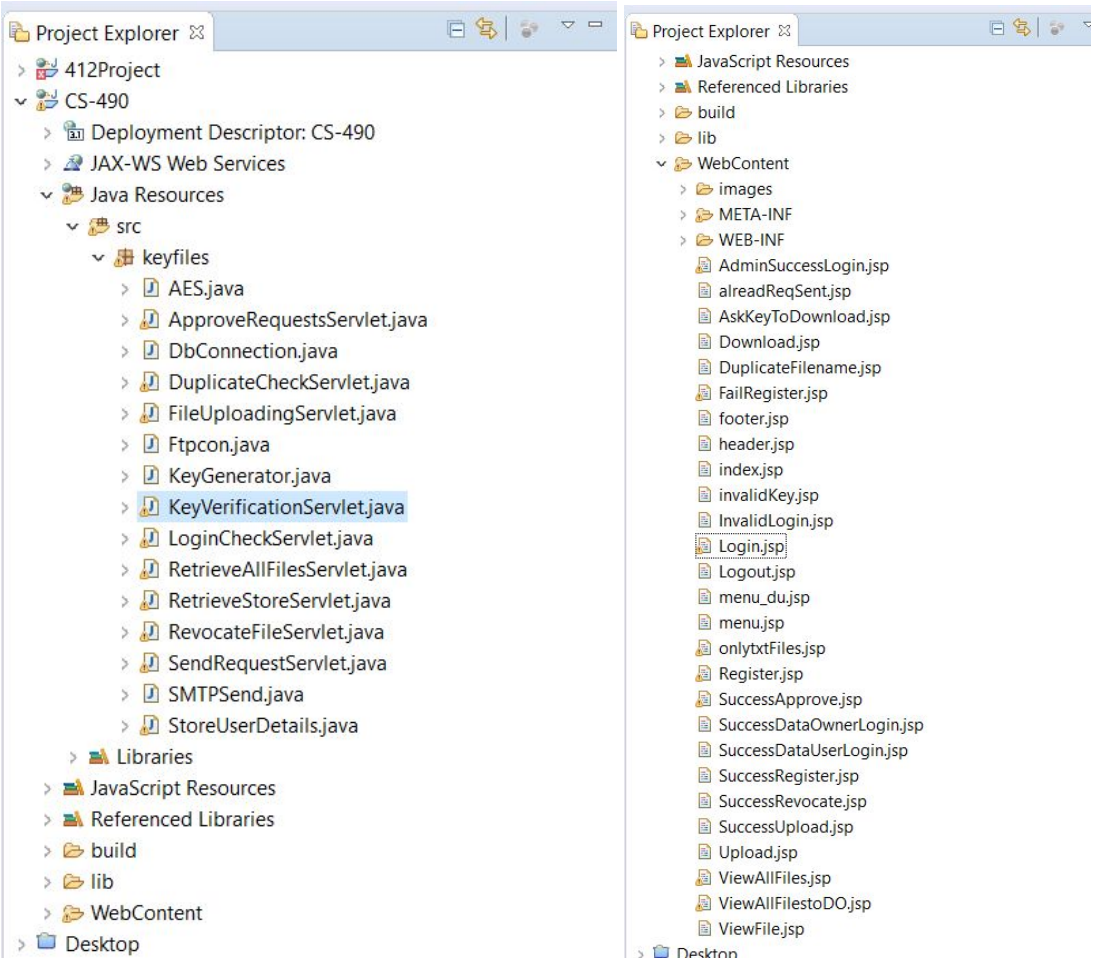


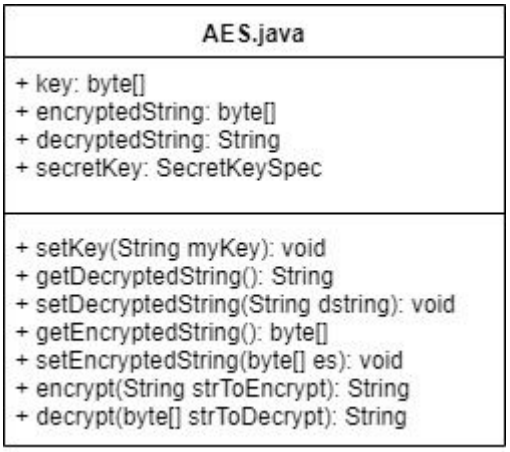
Fig 3.2.1: Java Resources(left) & Webcontent(right)

The following classes listed below perform the most work important regarding the project. The classes and their contributions to the project are explained below.

- AES.java
- KeyGenerator.java
- KeyVerificationServlet.java
- FileUploadingServlet.java
- RevocateFileServlet.java

Main Encryption Algorithm: AES (Implemented in AES.java)

This project deals with protection of consumer data, there is a need to decide on the proper method to provide this security. Since we are building a system that will not require the use of multiple keys, symmetric encryption algorithms were considered. AES was selected for this project due to its security (better than DES), versatility(block cipher) and also due to its fairly inexpensive requirement for computing power. The mode chosen for the AES algorithm to implement was the Electronic Code Book (ECB). UML[10] diagram is provided below.



The encryption algorithm is implemented in the class AES.java as follows:

Important imported libraries:

1. java.security.MessageDigest;
2. javax.crypto.Cipher;
3. javax.crypto.spec.SecretKeySpec;
4. org.apache.commons.codec.binary.Base64;

- **Function:** public static void setKey(String myKey)
    1. A new byte array object "key" is used to stored the bytes of the "myKey" object. The encoding used is UTF-8.
    2. A MessageDigest object is instantiated to hash and digest the values of the "key" object.
    3. Since AES uses 128-bit blocks, the values of the "key" object are copied and are padded to a length of 16 bytes. (Since 16 bytes = 128 bits).
    4. Lastly, the secret key is created using the SecretKeySpec library and the method specified is "AES". (Example: *secretkey* = new SecretKeySpec(*key*, "AES");).

- **Function:** public static String encrypt(String strToEncrypt)
    1. A cipher object is initiated using the electronic code book mode.
    2. The cipher is initiated using cipher.ENCRYPT_MODE and the "secretkey" prepared in the class before.
    3. The Cipher initiates its process in encrypt mode after getting the encoding (UTF-8) of the "strToEncrypt" bytes. The bytes are then converted from binary to ASCII using the Base64 library.
    4. The new string is then set as the encrypted string using the setEncryptedString method.

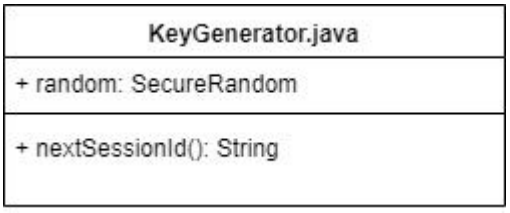
- **Function:** public static String decrypt(byte[] strToDecrypt):

This function takes in an array of encrypted bytes, decrypts and returns a string.

- A cipher object is initiated using the electronic code book mode.
- The cipher is initiated using cipher.DECRYPT_MODE and the secret key prepared in the function before.
- The Cipher initiates its process in decrypt mode after getting the encoding (UTF-8) of the "strToDecrypt" bytes. The bytes are then converted to binary and decrypted using Base64 and the initiated cipher object.
- The bytes are then converted to a string object and new string is then set as the encrypted string using the setDecryptedString method.

Key Generation: KeyGenerator.java

This class is used to generate the secret key for the user. Here, a simulated sessionID is used as the basis for the secret key.



Function: public static String nextSessionId()

- A random BigInteger(32, *random*), uniformly distributed over the range 0 to (2^{32} - 1), inclusive, is generated. We are using this function to simulate a sessionID.
- The SecureRandom library is used to generate a SecureRandom object "*random*" to ensure that a cryptographically strong number is generated.

Key Verification: KeyVerificationServlet.java

This servlet is used to check the key the user entered, and performs an action based on the result.

- User enters a key when prompted.
- The parameter "key" of the HttpServletRequest object "request" is saved to a new String object. This object will be used to compare the key value to the correct key stored in the database.
- The Connection class is used to obtain a connection to the preferred database.
- If the key entered equals the correct key in the database, then the user is redirected to "Download.jsp", the page where they can download and view the file.
- If the key entered is incorrect, the user is redirected to an error page titled "invalidKey.jsp", which notifies the user that the key they entered is incorrect and to try again.

File Upload: FileUploadingServlet.java

This class uploads a user file after encrypting it using the KeyGenerator.java and AES.java class. This class also checks whether the file being uploaded is a duplicate file, and also verifies the type of file being uploaded i.e. .txt files and performs error handling as well. This class uses a important java library resources which are listed below.

1. Duplicate File/File Type Check
2. Secure Key Generation & Encryption
3. Upload to cloud

Important imported library resources:

1. java.io.File
2. java.io.FileInputStream (Used to read data from file)
3. java.io.FileOutputStream (Used to write data out to file)
4. java.util.Hashtable (Implements a hash table to map keys to values)
5. javazoom.upload.MultipartFormDataRequest (Used to parse and handle HTTP request for uploading data).
6. javazoom.upload.UploadBean

A. Duplicate File/File Type Check:

- The destination paths for the original and encrypted files are specified first. This path can be anywhere on local system or web/storage server.
- The user-uploaded file is then enumerated into sets and hashtable made.
- A Database query is set up to check the file uploaded using its name. The table used for this check is the “store” table.
- Duplication Check: The filename is then used to check the store for any duplicate file names. If a file with the same name exists, the user is then redirected to DuplicateFilename.jsp where they are prompted to upload a file with a new name.
- File Type Check: The file type of the file is checked here. Since this system only supports the upload and sharing of .txt files, we will check the file type of the uloaded file by checking the path of the original file. If the path ends with “.txt”, we know that the file is a .txt file. If it is not, then the user is redirected to an error page “onlytxtfiles.jsp”, where they are notified that only .txt files are allowed for upload.

B. Secure Key Generation & Encryption:

- A FileInputStream object is used to read in the file from the original path.
- A FileOutputStream object is used to write the encrypted data to a file on the encrypted path.
- The FileInputStream object reads in the characters of the uploaded file and stores then in a String object “input”.
- **KeyGenerator.java** generates a key.

- AES.java sets the key using **setKey** function, and encrypts the uploaded file using the function **encrypt(input)**.
- The encrypted data is written out using the path specified by the FileOutputStream object. The data is being stored in the file on the encrypted file destination path.

C. Upload To Cloud:

- A new File object "f" is created using the file stored on the encrypted destination path.
- A connection object for the File transfer protocol is instantiated. This connection is made to a cloud-based data hosting website DriveHQ. The FTP connection to the DriveHQ is performed in the Ftpcon.java class(located in project resources).
- The encrypted file "f" is uploaded using the upload method specified in Ftpcon.java.

File Revocation: RevocateFileServlet.java

This servlet performs the File Revocation i.e. the deletion of the encrypted file and a new encrypted file being uploaded with different keys. This servlet follows some of the same steps used by the FileUploadingServlet.java, along with a few changes. It performs two additional steps, which are deleting the file from the cloud and updating the keys in the database.

- Once the Revocate link on the webapp is clicked, the file stored at the location for encrypted files is decrypted and copied. This step can also be simulated in a different way by using the file at the path location for the original, unencrypted files, and re-reading and encrypting that file. However this requires the original file must be present at the path location.
- The **Secure Key Generation** step takes place.
- The file with the old encryption is deleted.
- The **Upload to Cloud** step takes place.
- A new database connection is opened and the cipher keys are updated in the "store" table in the database.
- The statement is executed and the server response is dispatched.

4.3 Project flow

Design consists of two modules:

1. Data Owner/User
2. Administrator

1. Data Owner/User:

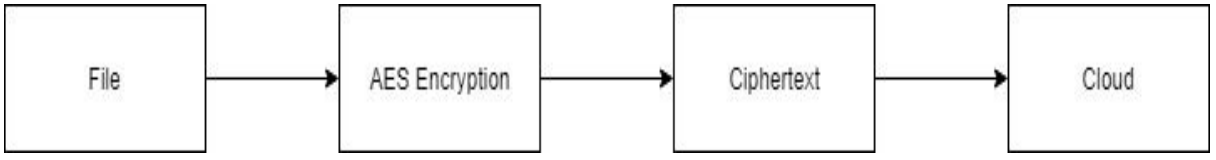
In this module, we develop the Data Owner/User module. The data owner module is developed such that the new users will signup initially and then login for authentication. The data owner module provides the option of uploading the file to the Cloud Server. The process of file uploading to the cloud Server is undergone with Identity-based encryption format.

The data owner will check the progress status of the file upload by him/her. Data owner/user is provided with the features of Revocation and Ciphertext update to the file, as well as the option to upload and view all the files that have been uploaded to the cloud server. The user can also send a Request for File Access to the Administrator for any particular file that the user wants to access but cannot.

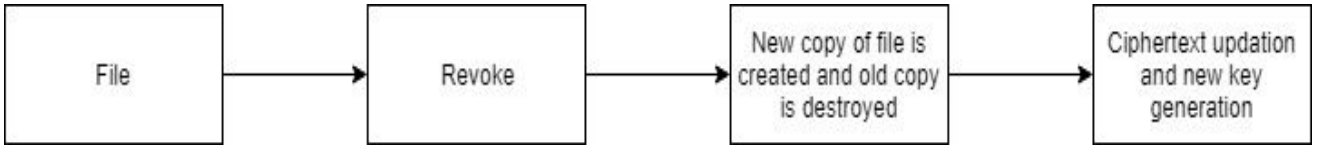
After getting decrypt key from the Administrator, he/she can access the File. The data owner/user is also enabled to download or view the File. After completion of the process, the user logs out of the session.

Data owner/user can do the following:

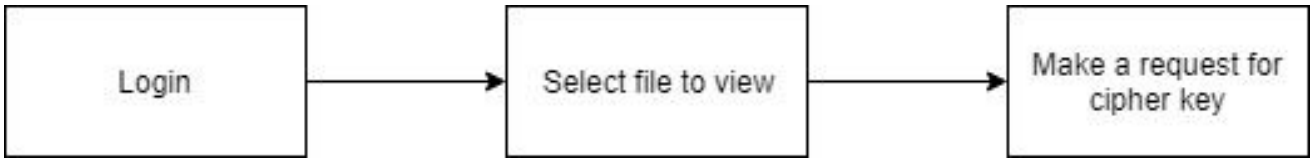
a.**Upload a file:** The data owner can register to maintain their file into cloud server and allow access to data users to access the file. Here the data in files will be encrypted using AES(Advanced Encryption Standard) and using FTP(File Transfer Protocol) they will be stored in the cloud.



b. Revocate: As the data user is no longer part of the organization, he/she should not be able to access or download the files uploaded by data owner. Data owner can thus revocate the files i.e. a new copy of a existing file which is downloaded by user is created in the cloud, new key is generated, cipher is updated and then the old copy is deleted from the cloud.



C. View & Request File: The data owner/user can view the files uploaded and file names. The request to the Administrator for the secret key can be made here. Once it has been approved, the cipher key will be received through the user's email, after which they can use it to view the file.



2. Administrator:

Administrator will login on the Administrator page. He/she will check the pending requests of any of the above person. After accepting the request from the above person, he/she will generate the secret key for encryption and decryption. After the complete process, the Administrator will logout the session.

Administrator can view the requests made by data user and respond to them i.e sending a cipher key through email.

4.4 System Design Diagrams:

4.4.1 Use Case Diagram

Use case diagram is used to capture the dynamic aspect of a system.The purposes of use case diagram is it is used to gather requirements of a system and to get an outside view of a system. It is used to identify external and internal factors influencing the system and show the interaction among the requirements.

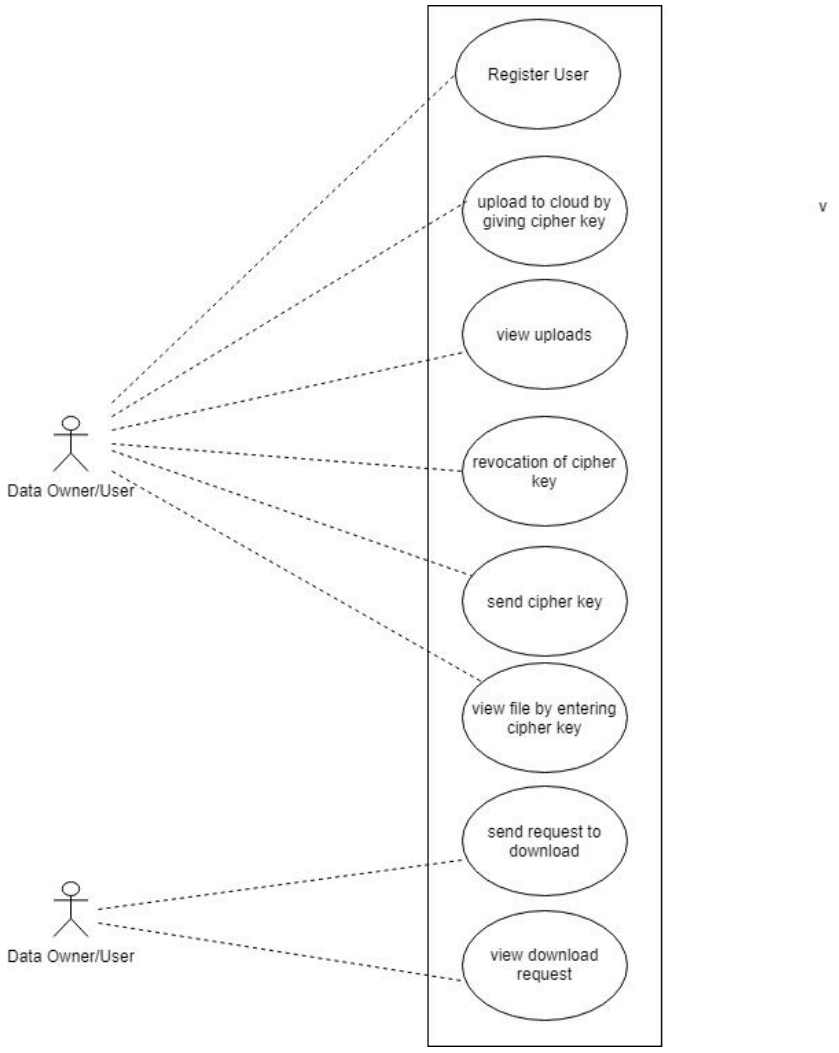


Fig 4.4.1 Use case diagram

4.4.2 Activity Diagram

Activity diagrams are mainly used as a flow chart consists of activities performed by the system. The purpose of activity diagram is to describe the sequence from one activity to another. It is also used to describe the parallel, branched and concurrent flow of the system.

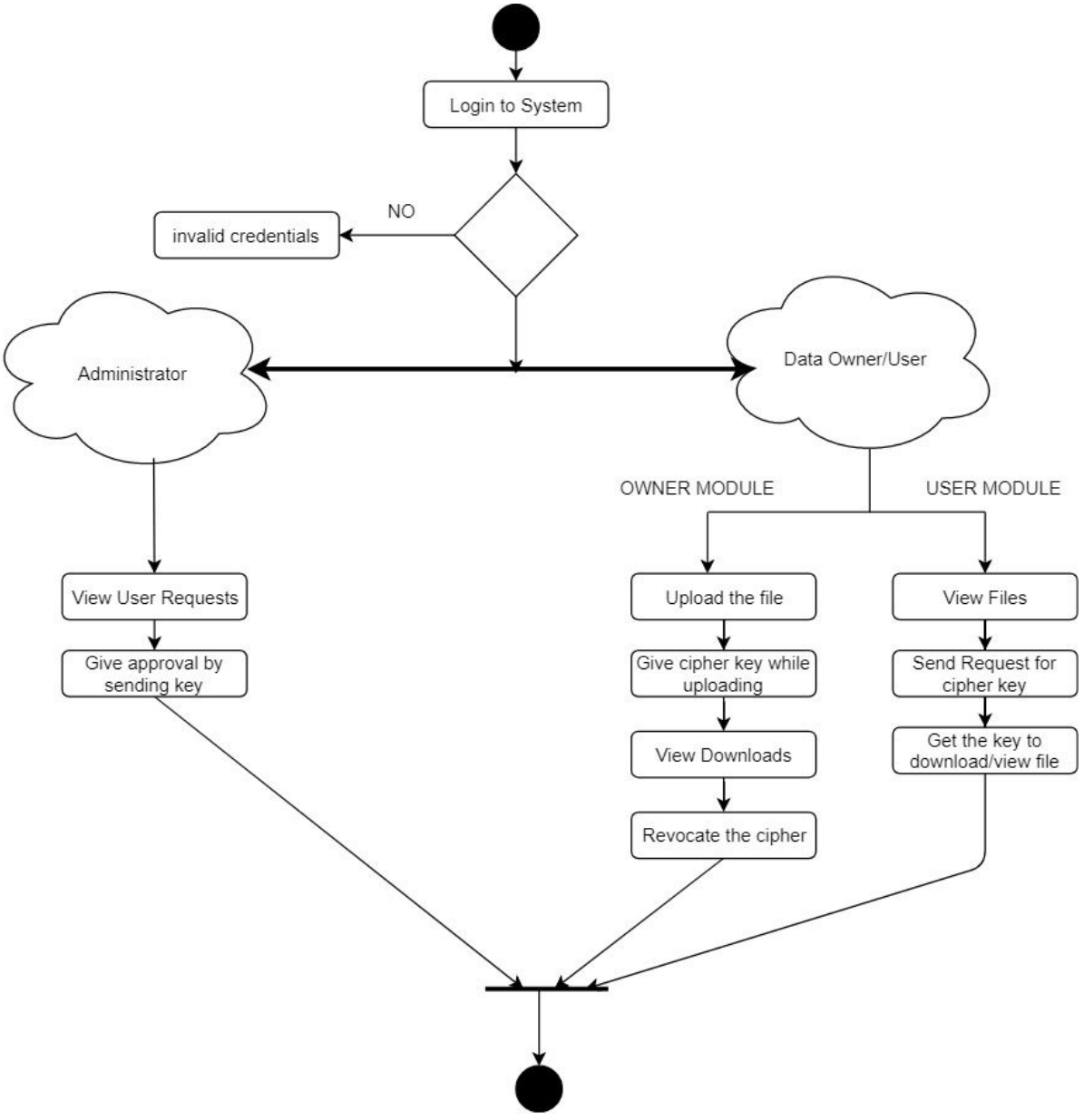


Fig 4.4.2 Activity diagram

4.4.3 Sequence Diagram

Sequence diagram emphasizes on time sequence of messages. The purposes of sequence diagram is to capture dynamic behaviour of a system i.e, to describe the message flow in the system and describe structural organization of the objects and interaction among objects.

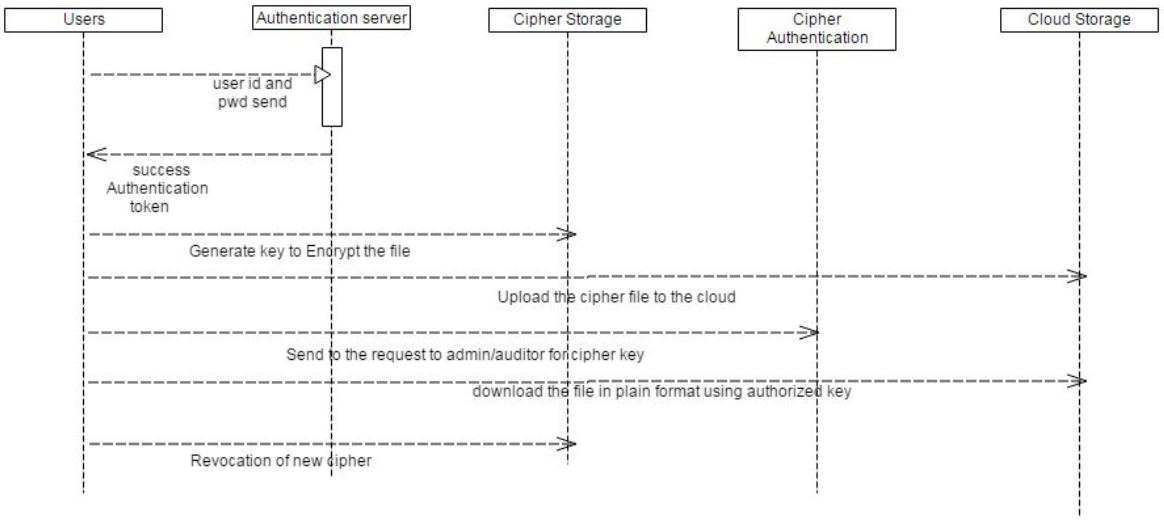


Fig 4.4.3 Sequence diagram

4.4.4 Deployment Diagram

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed. So deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships. Usage of deployment diagrams can be described as to model the hardware topology of a system, to model embedded system, to model hardware details for a client/server system, to model hardware details of a distributed application, forward and reverse engineering.

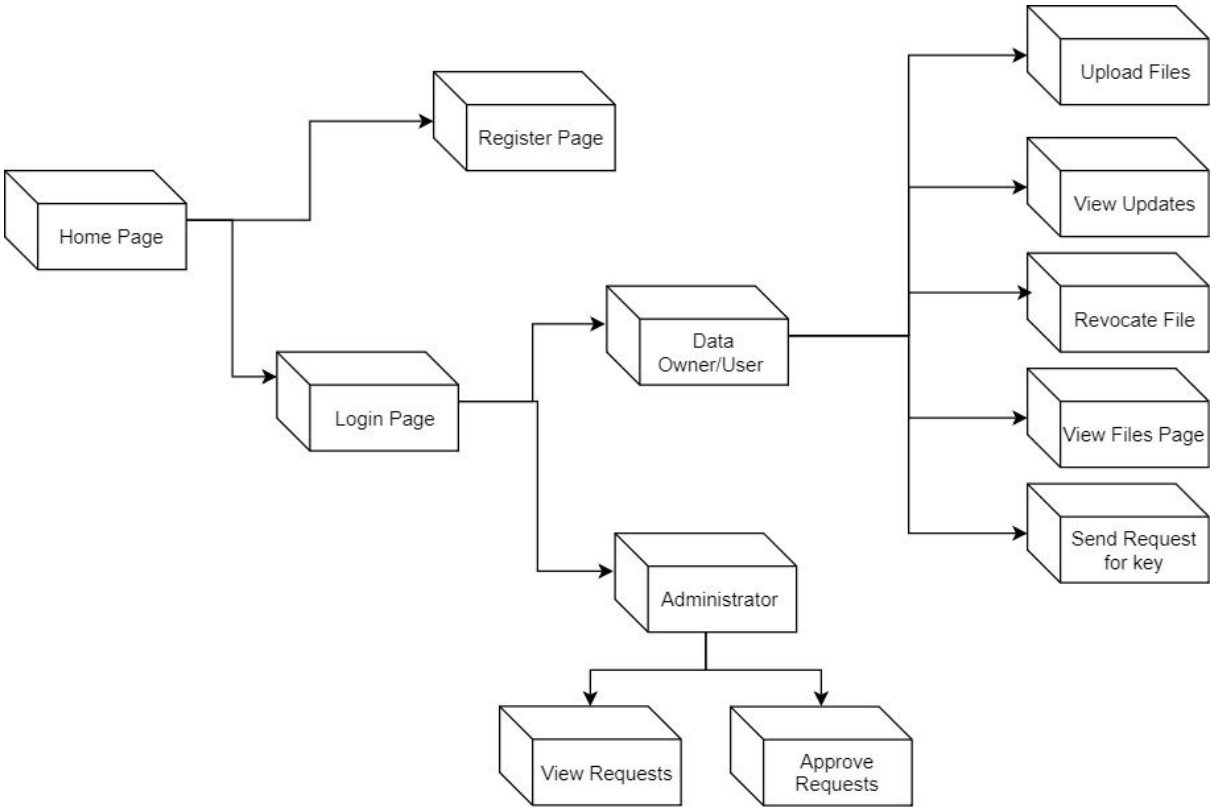


Fig 4.4.4 Deployment diagram

CHAPTER V: THREAT MODEL, TESTING & RESULTS

5.1 Threat Model:

To understand the complexities of the system, we will first be constructing a threat matrix to assess the security functionalities of the proposed system. Threat modeling is a procedure used for optimizing network security by identifying objectives and vulnerabilities and then devising countermeasures to prevent the effects of threats to the system.

5.1.1 Roles:

- Data Owner/User
- Data Administrator

5.1.2 External System Considerations:

These are the external system considerations that we are making before we define the security model of the system. These considerations aim to provide a framework upon which to develop this system.

- Secure private network between web server and database server.
- MySQL database.
- Database server will be the existing data server.
- All servers behind firewalls.
- All communications over TLS.
- Site should be accessible over any network i.e. private or public.

5.1.3 Use Case Scenarios:

- System only allows for .txt files to be uploaded.
- User can search for files uploaded in the database.
- User can encrypt and upload files.
- User can revocate files uploaded by them.
- User can request Admin for cipher key for a specific file they wish to access.
- Admin receives request for cipher key.
- Admin can approve requests based on user.

5.1.4 Assets:

Assets are the most important parts of the system that are most likely to be attacked for their value and system information. We decide that the following are the most valuable assets of the system:

- Users
- User credentials

- Data/ Data Content
- Database system
- Webapp Database access

5.1.5 Threat Matrix:

The following matrix is an analysis of the threats faced by the system and the actions taken to protect it.

Threat	Asset	Action
Exploiting SQL injection flaws	User, Database	Perform data validation at every entry point i.e. Login page and redirect to Login Page. Username only allows alphanumeric characters and Password has a required length & complexity.
Unauthorized access through URL/URL Data Visibility	User, Data	No sensitive data is visible through the URL due to the use of POST method. No direct access to data.
Unauthorized user	User	Implements authorization checks i.e. checks if either Data User or Admin. If neither, then prompts user to register with the system.
Browser Cache may contain contents of data	Data	Pages with data sensitive content implement anti-caching HTTP headers. Ex: ViewFile.jsp SSL used for other communications.
Visibility of Data/Data Interception	Data	Data is encrypted before being stored on the cloud Database. Any attack would only result in acquiring the

		ciphertext. Use SSL for all communications.
User negligence/User may not have logged off on a shared computer	User, Data, Database	Timeout function has been implemented. User is automatically logged out and redirected to Login page after 30 seconds of inactivity.

5.1.6 Abuse Case Example:

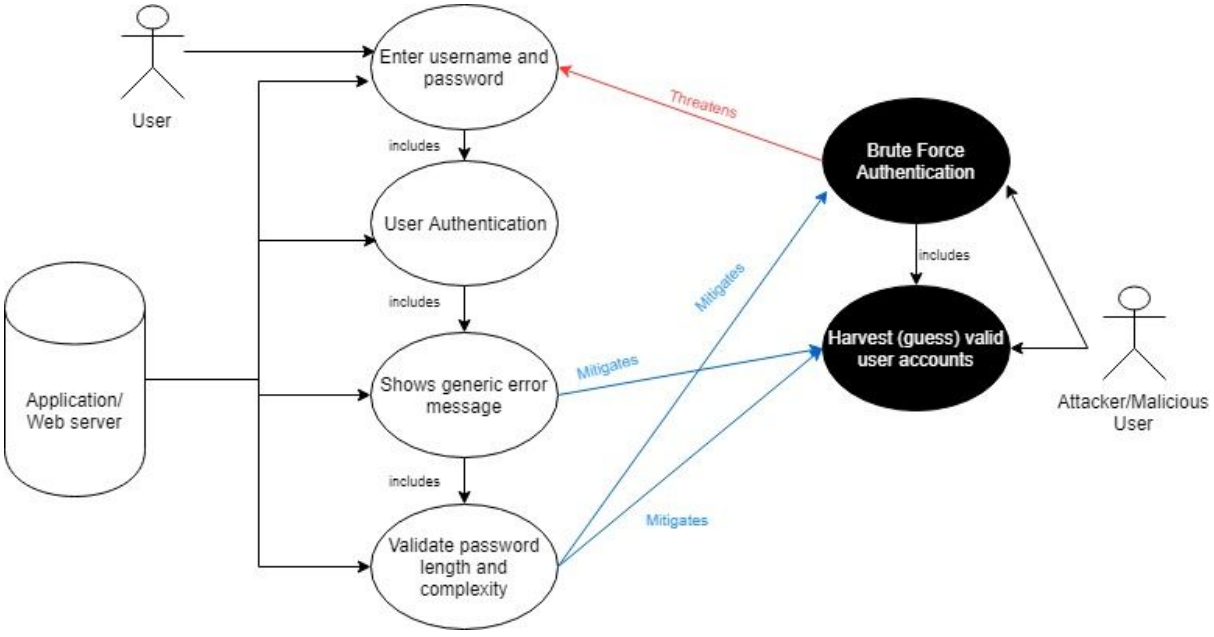


Fig. 5.1.7 User authentication abuse

5.2 Types of Testing;

5.2.1 Unit Testing:

Unit testing is a level of software testing which involves the design of test cases in which individual units of the source code are tested and validated. All internal code and decision branches should be validated. It is meant to evaluate individual software units of the application after the completion of the unit but before it's integration.

Unit tests perform tests at the component level and test a specific application, system configuration and/or a business process. The tests rely on the knowledge of the construction

of the system. Unit tests ensure that each unit of the application performs as accurately to the documented specifications and contains clearly defined inputs and expected results.

5.2.2 Integration Testing:

Integration testing is the analysis of two or more integrated software components on a single platform, to detect failures from any interface defect. It is specifically aimed at exposing the

problems arising from a combination of the components. The purpose of this test is to ensure the components of the software application interact without error, and also measures accuracy and consistency.

5.2.3 Functional Testing:

Functional tests provide a systematic demonstration of the functions tested are working as specified by the business and technical requirements, system documentation and user manuals. It is based on the following items:

- Valid Input: identified classes of valid input must be accepted
- Invalid Input: identified classes of invalid input must be rejected
- Functions: identified functions must be exercised
- Output: identified classes of application outputs must be exercised
- System/Procedures: interfacing systems/procedures must be invoked

5.2.4 System Testing:

System testing ensures that the entire integrated software system meets requirements. It evaluates the entire system configuration inputs to assure known and predictable results.

An example of system testing is the configuration-oriented system integration test. System testing is based on the process of descriptions and flows, emphasizing pre-driven process links and integration points.

5.3 Testing Criteria:

5.3.1 Unit Testing:

Unit testing is usually driven as part of the combined code and unit test phase of the software life cycle, although it is not unusual for putting in unit-testing to be conducted as two distinct phases.

5.3.2 Test Strategy and approach:

Field testing will be performed manually, and functional tests will be written in detail.

5.3.3 Test objectives:

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

5.3.4 Features to be tested:

- Verify that the entries are in the correct format.
- No duplicate entries should be allowed.
- All links should take the user to the correct page.

5.3.5 Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

5.3.6 Acceptance Testing:

User acceptance testing is a critical phase of any project and requires significant participation by the end-user. It also ensures that the system meets all functional requirements.

5.4 Test Cases & Results:

Test Cases	Expected Output	Actual Output	Test Status YES/NO
Login to the Data Owner/User webpage.	Data owner/user must login to the web page.	Owner/User profile is open and owner performs actions.	**YES**
Uploading Files	File should be encrypted and uploaded.	Files are correctly uploaded; both original and revocated copy are encrypted.	**YES**
User View File	File should be displayed for downloading.	User able to view files for downloading.	**YES**
Download Files	File should be downloaded with key for the current user only.	Original file is downloaded with appropriate key for data user	**YES**
File Storage	Files should be stored with encrypted key.	Verified in cloud. Files are stored with encrypted key.	**YES**
User Request for Key	User should request for appropriate key.	User able to get appropriate key and access the data.	**YES**
Revocation of Files	The original file uploaded to the cloud must be decrypted, copied, re-encrypted and uploaded along with different key.	The original copy is deleted and the encrypted copy is uploaded to the cloud. The secret key is also changed.	**YES**
Logout	Data owner/user should be able to log out	Data owner/user logs out to successfully.	**YES**

5.5 ANALYSIS & RESULTS:

Home Page: This view shows the homepage.

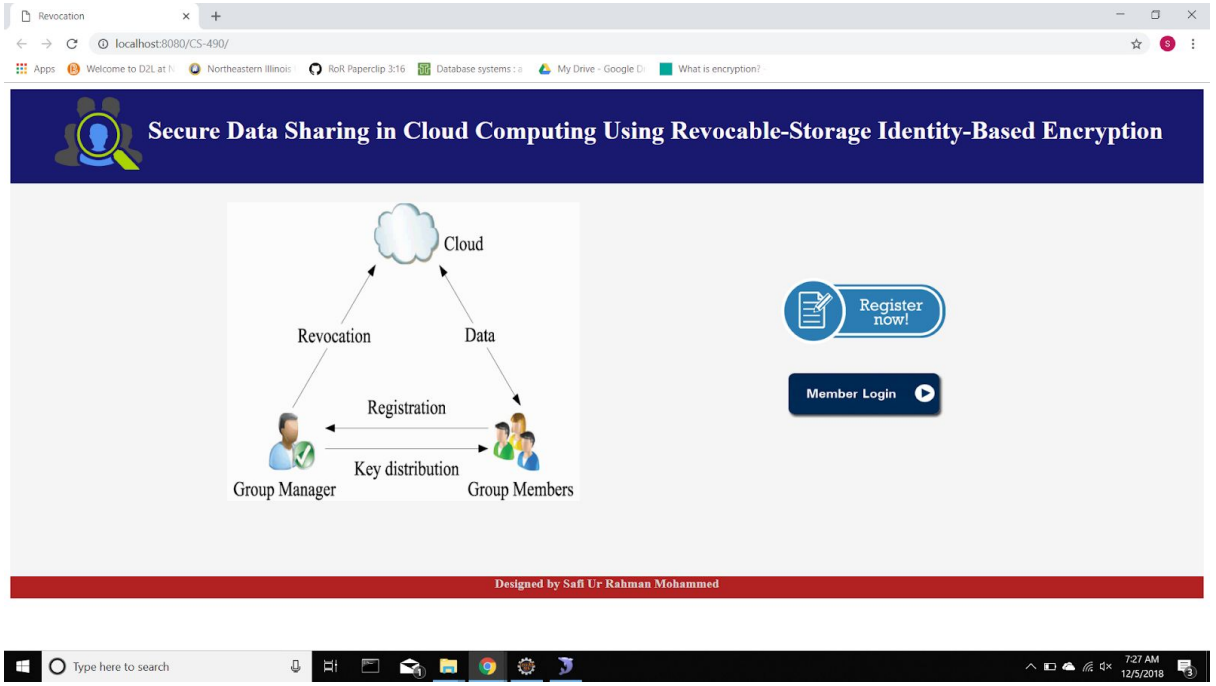


Fig 5.1 Home Page

Register page: New users can sign-up using this page.

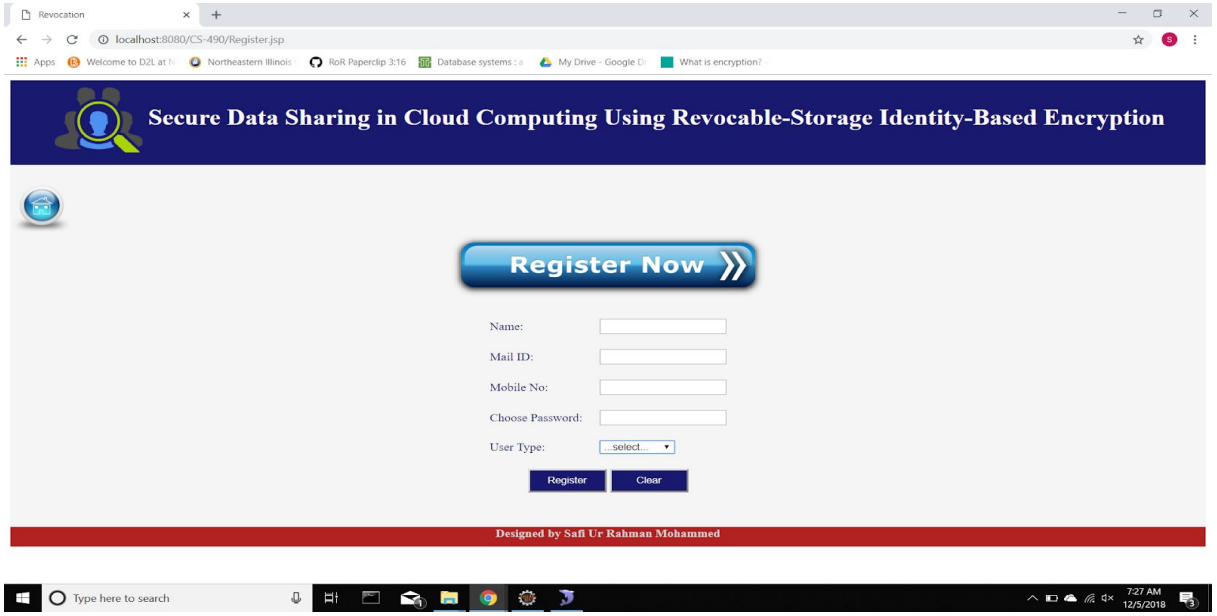


Fig 5.2 Register Page

Data Owner Logged-in: The profile page for logged-in users.

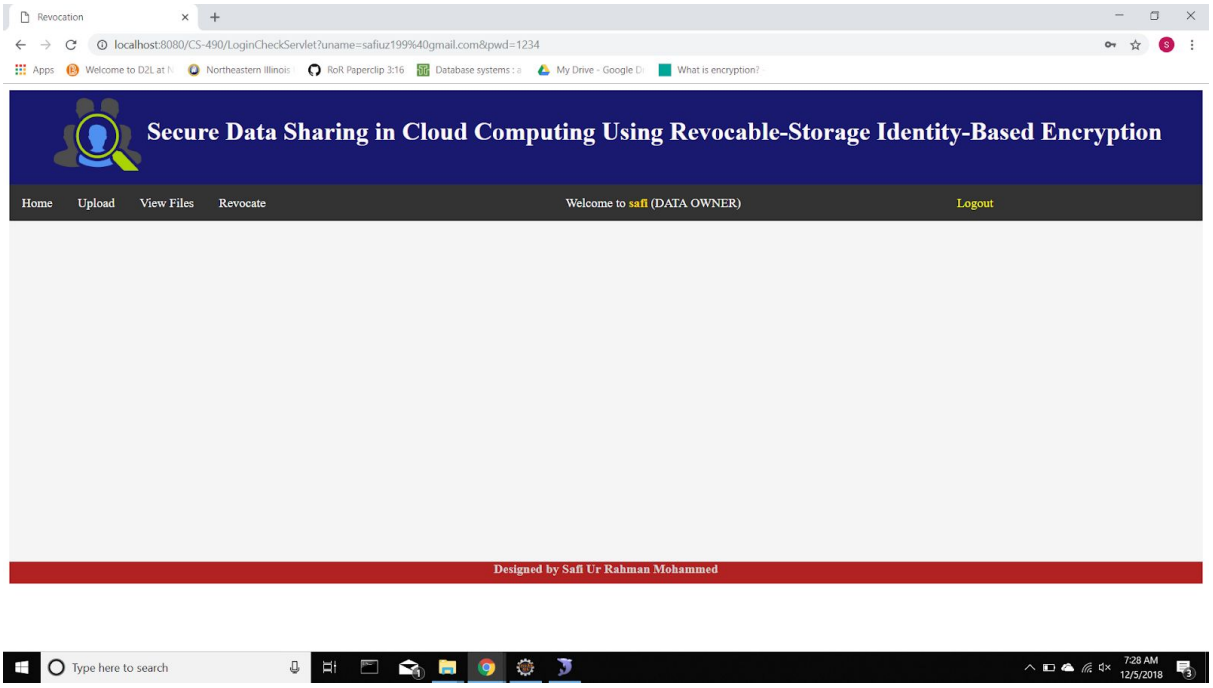


Fig 5.3 Data Owner Logged in

Upload File: Files to be shared can be uploaded here.

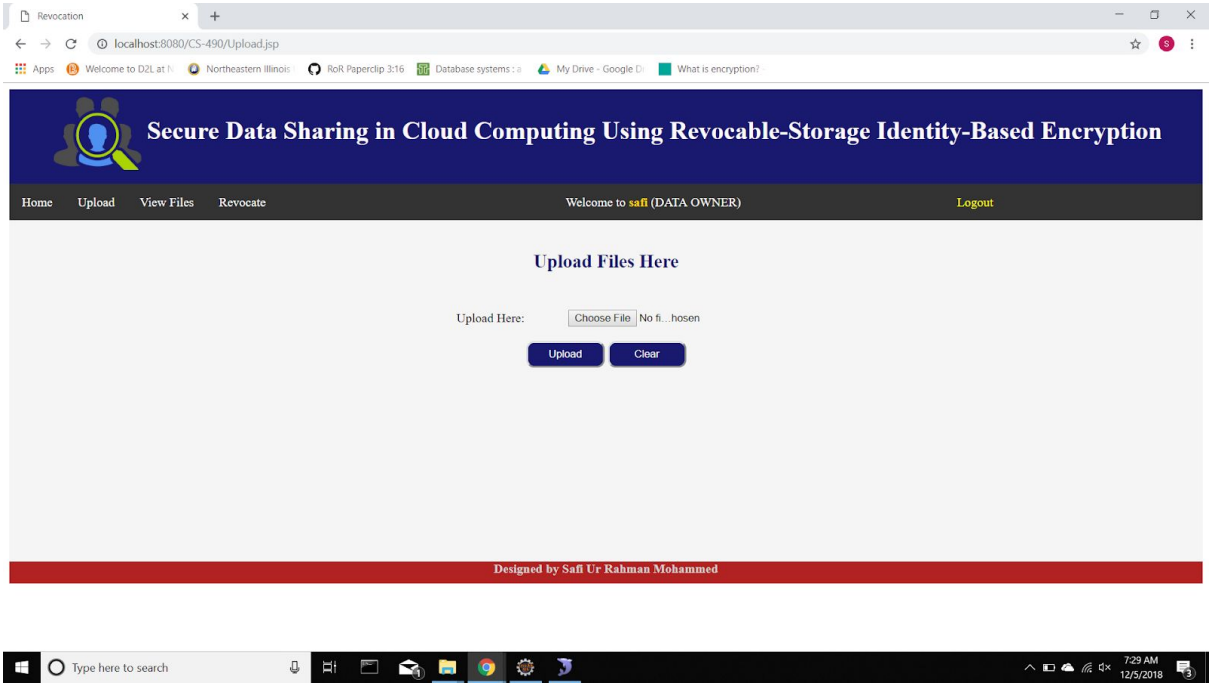


Fig 5.4 Upload File

Upload Successful: File uploaded successfully.

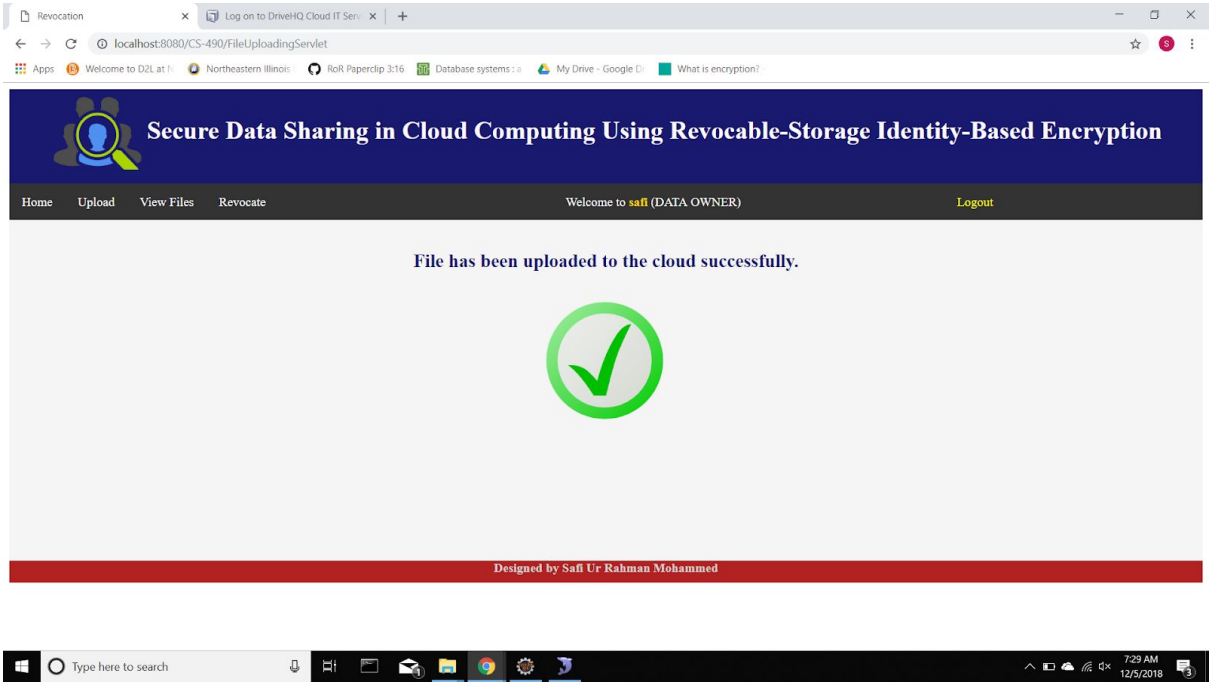


Fig 5.5 File uploaded successfully

Ciphertext of file in the cloud: This screenshot shows the file after being uploaded to the cloud service. The plaintext has been encrypted into ciphertext and uploaded. Attackers trying to access the file will only access ciphertext.

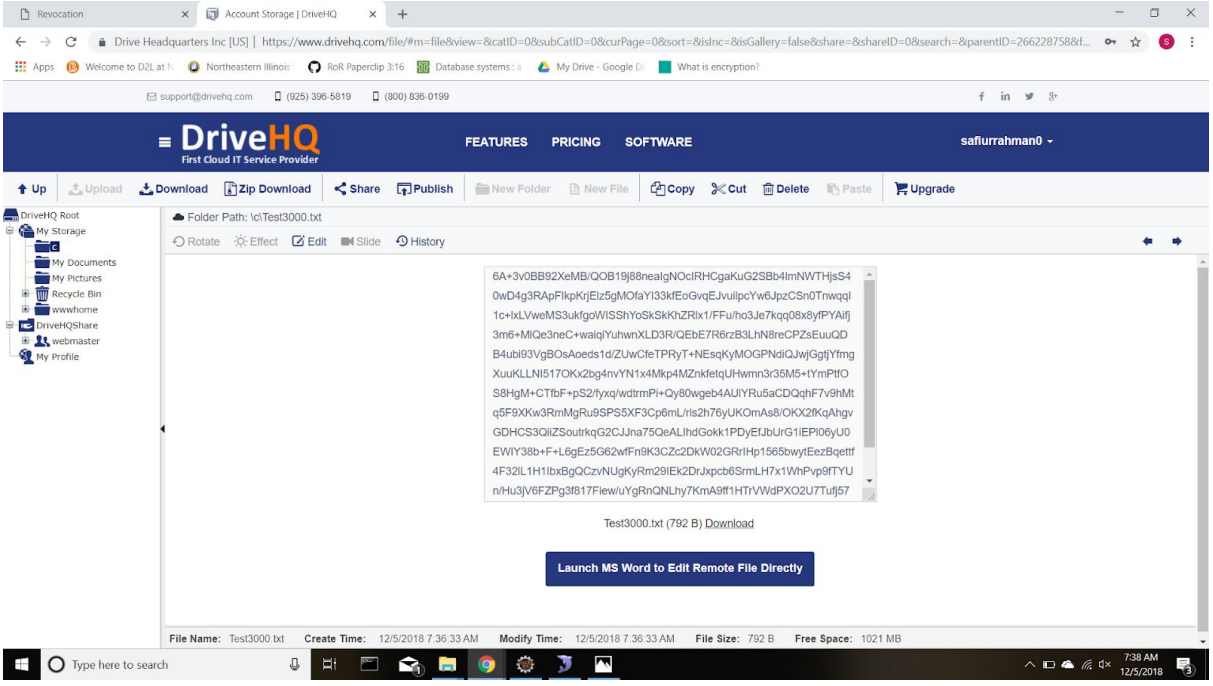


Fig 5.6 Ciphertext of file in cloud

View Files: All files shared in the cloud by other users. Files can be viewed after requesting and using the secret key for decryption after approval by the Data Admin.

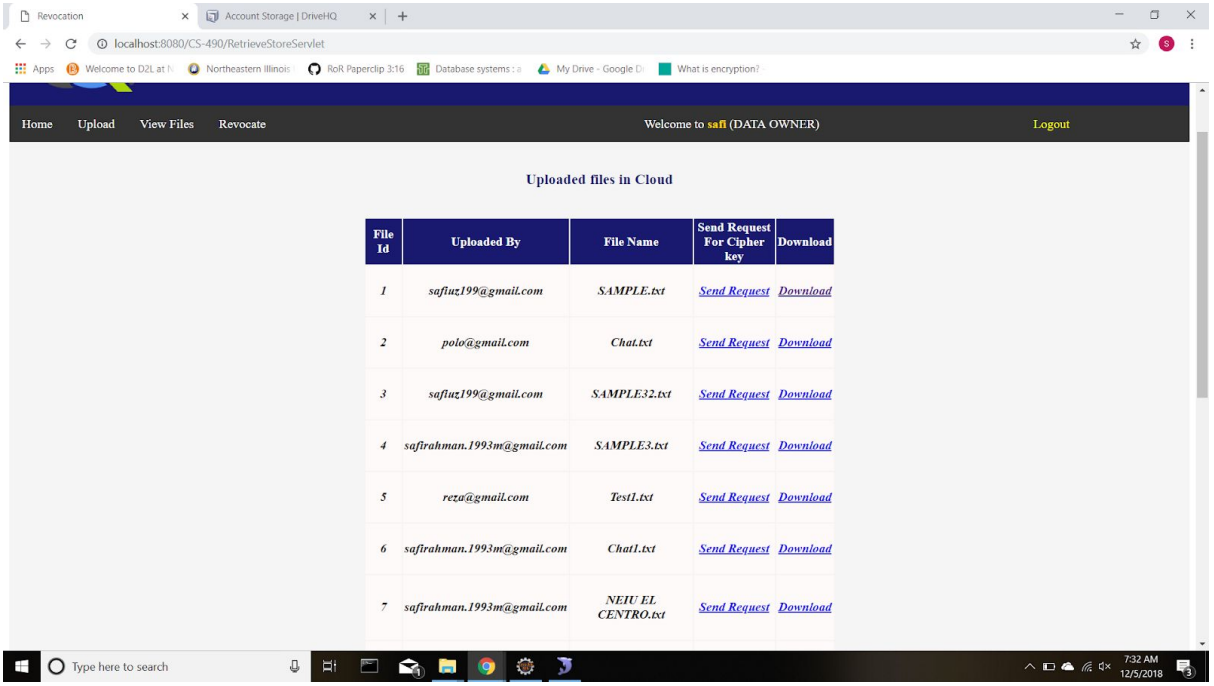


Fig 5.7 View all user uploaded files

Revocate files page: If the user wishes to change the access and ciphertext, they can revocate the file here.

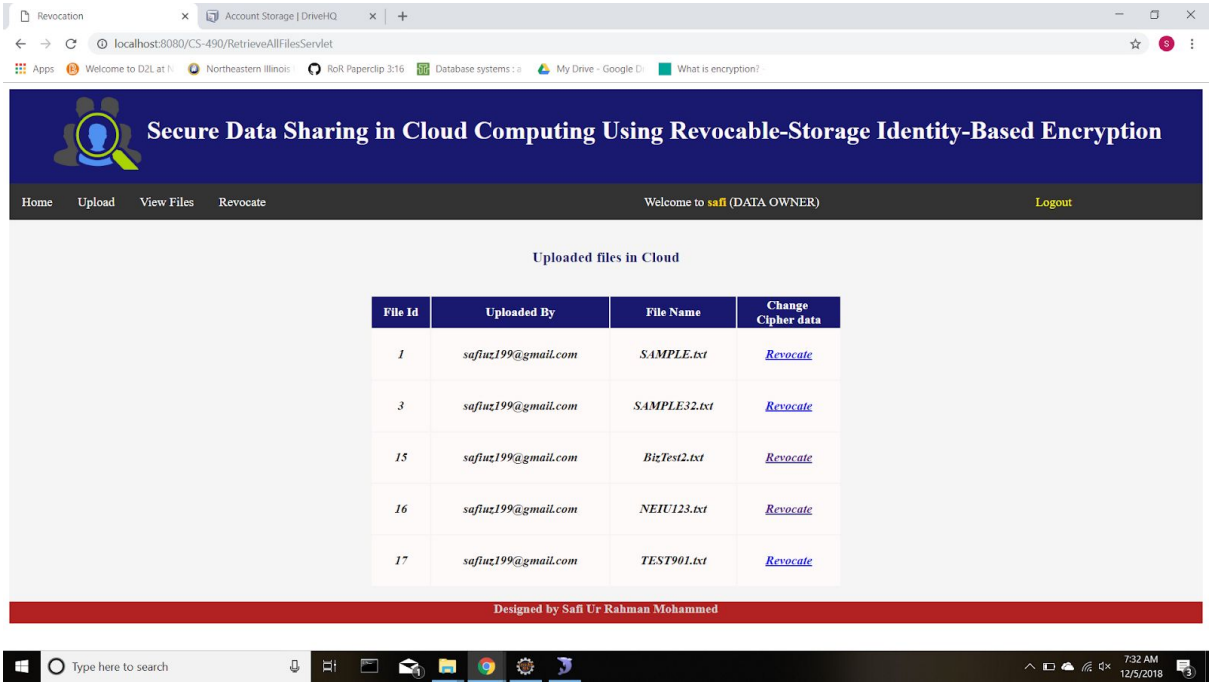


Fig 5.8 Revocate file page

Key Verification page: The secret key is entered and the data is decrypted using the correct key.

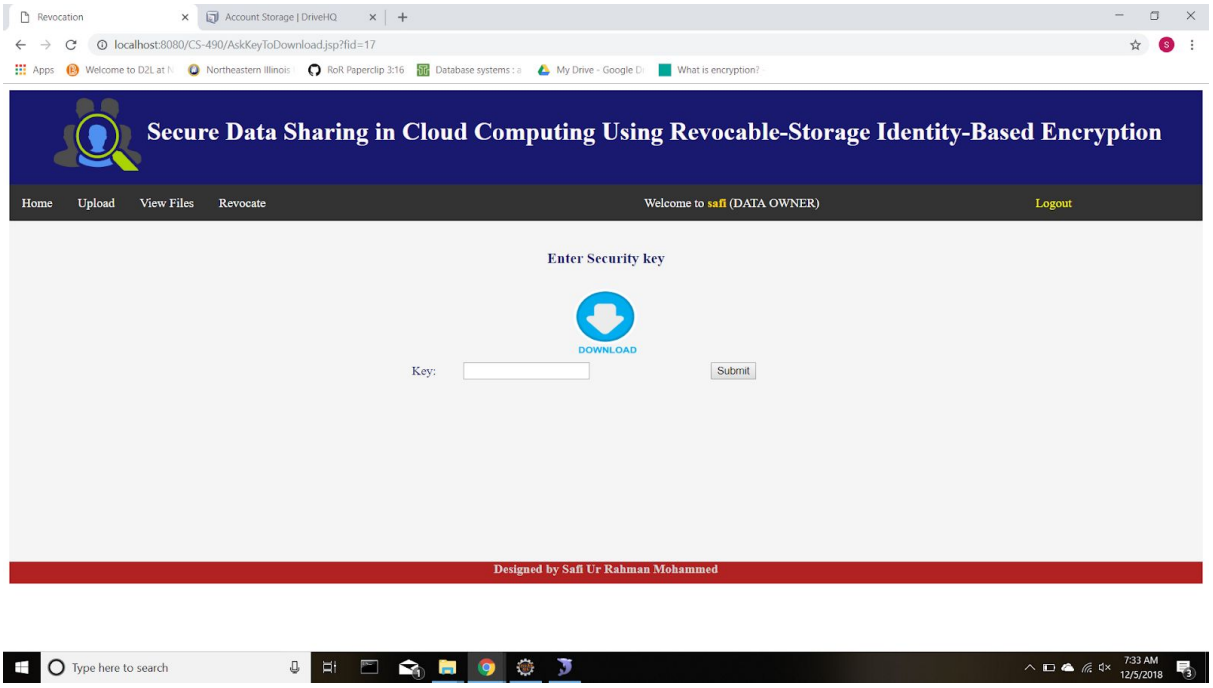


Fig 5.9 Download File

Content of file is displayed: The plaintext is displayed here following a successful key verification and decryption.

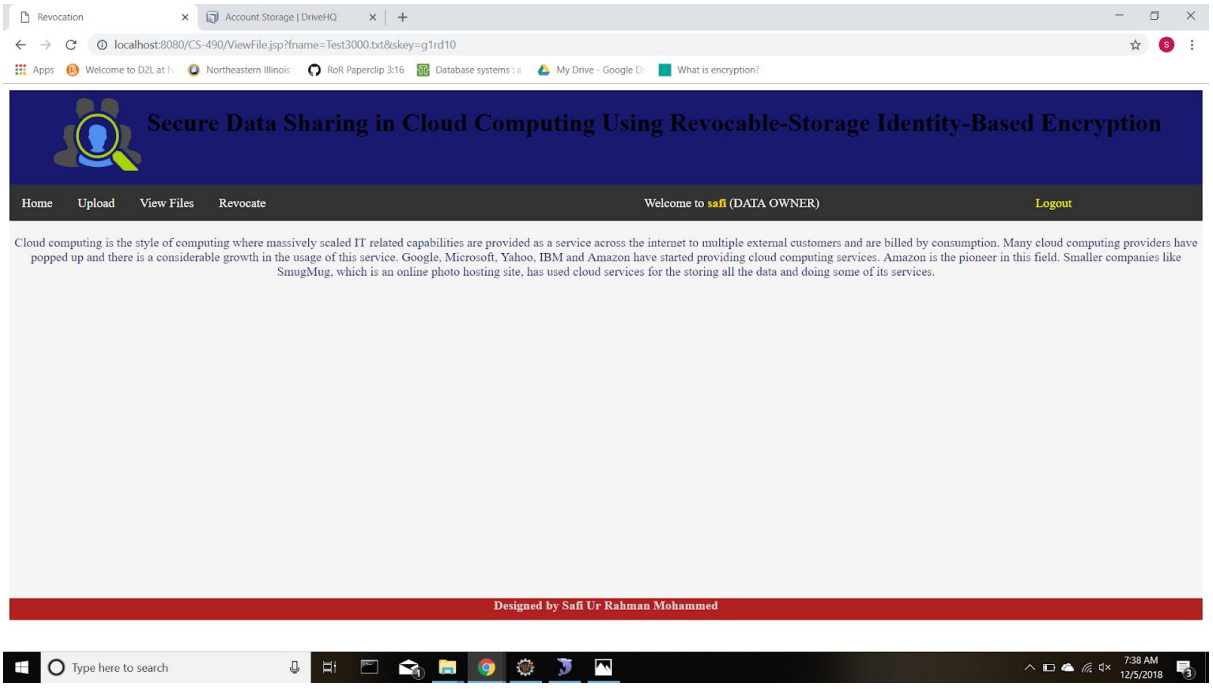


Fig 5.10 Content of file displayed (after decryption)

Administrator Page: Data Administrator Homepage

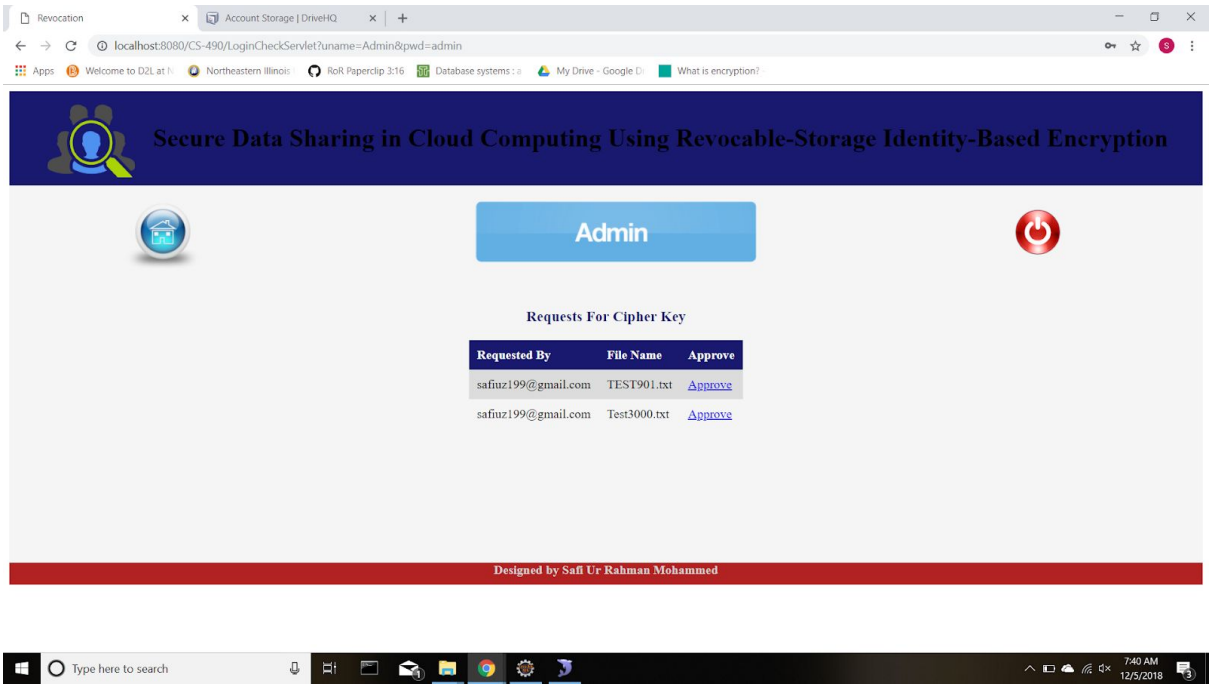


Fig 5.11 Admin Page

Database Tables: a) Store: Used to store User ID, filename and secret key.

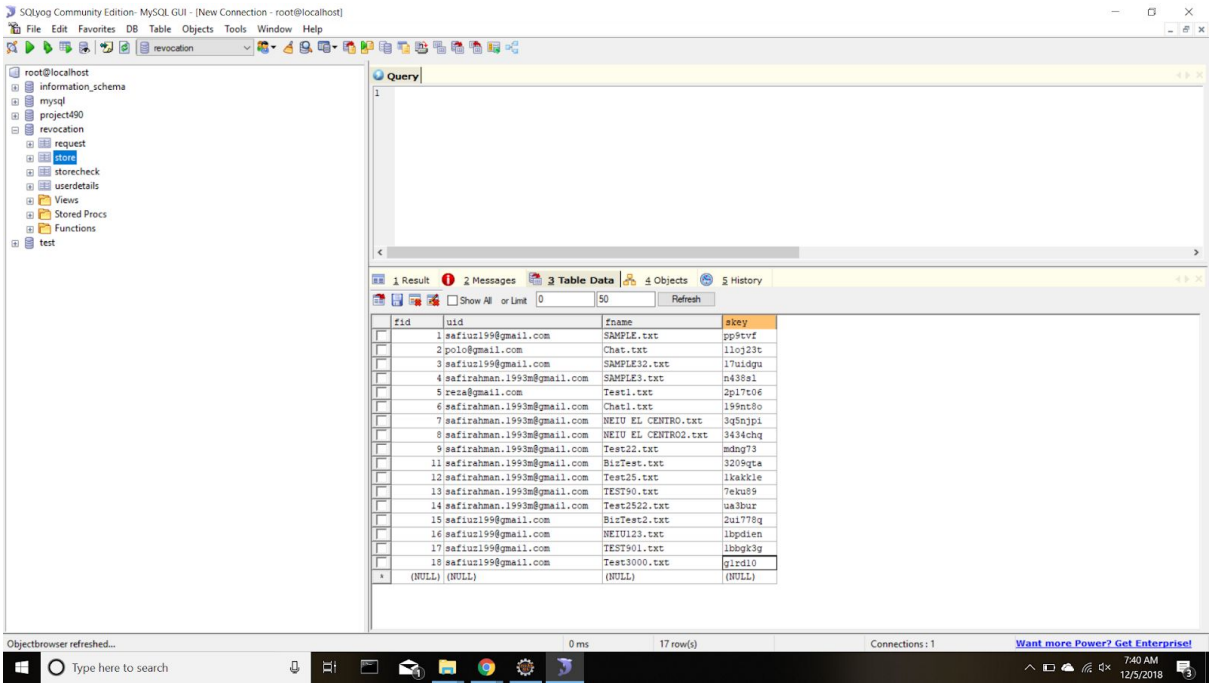


Fig 5.12 Store table

b) Request table: Keeps track of all unapproved and approved file requests.

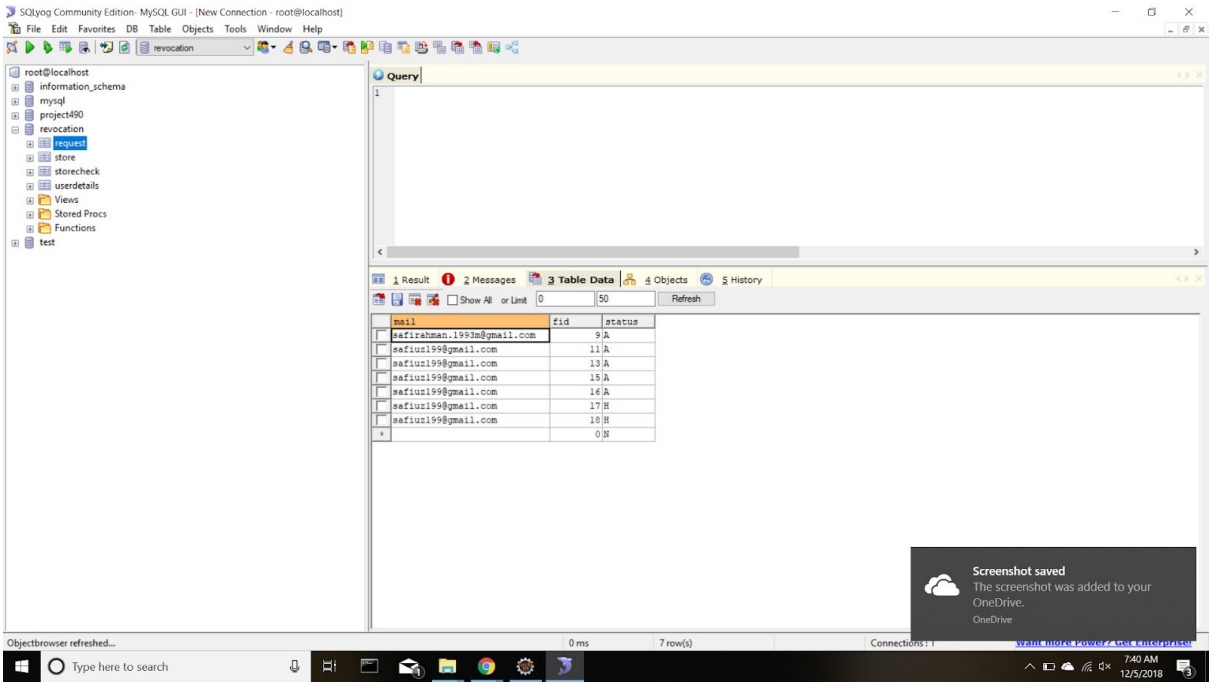


Fig 5.13 Request Table

c) User Details: Used to store User details upon registration. Also used for verifying user Login credentials.

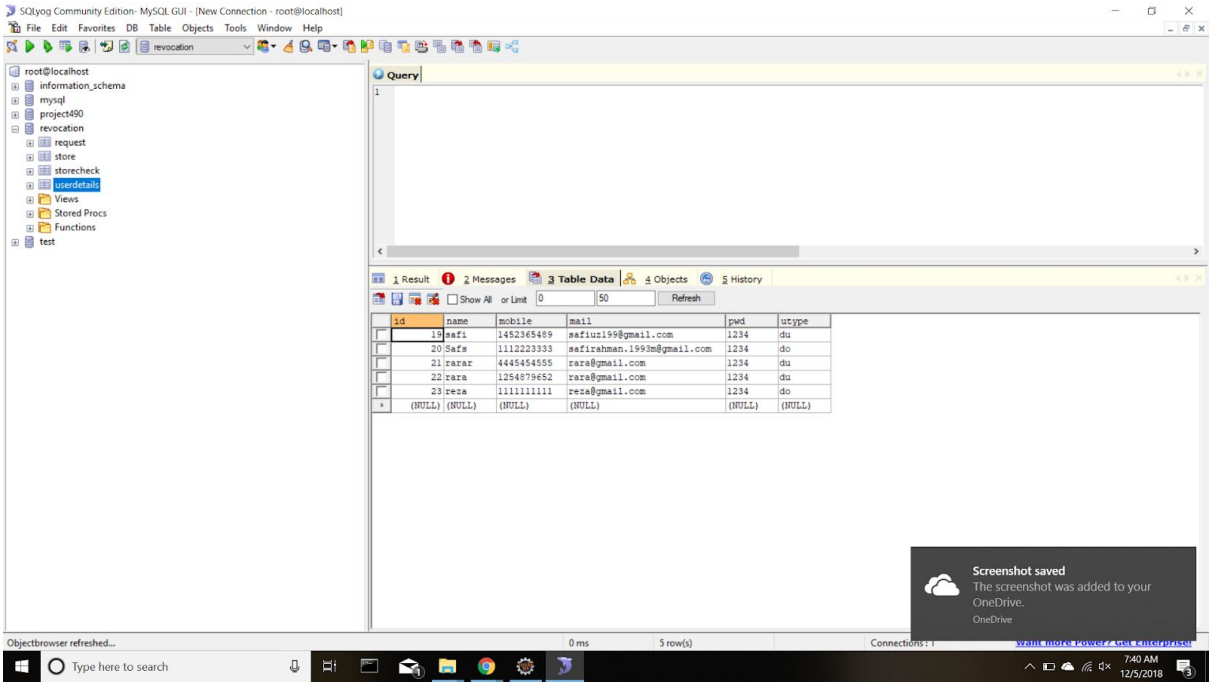


Fig 5.14 User Details

CONCLUSION AND FUTURE SCOPE

As more people migrate to cloud data storage services, the question of how to best secure our online information will inevitably have to be addressed. In recent years there have been massive breaches of confidential data from companies by Facebook. Cloud security policies dictate that the data has to be encrypted and protected from other users. But that does not make the data secure from other accessors such as the government or advertising agencies. The cloud providers may sell the data to a third-party or release their data to the government if mandated by the government. To this end, in this project, we proposed a notion called RS-IBE, which supports identity revocation and ciphertext update simultaneously such that a revoked user is prevented from accessing previously shared data, as well as subsequently shared data. Furthermore, a concrete construction of RS-IBE is presented.

Our proposed scheme makes it possible to achieve the right of privacy with no undesirable interference. By doing the identity check, we are able to make our data inaccessible to any other person than the data owner themselves.

The current project has a few limitations, and it can be further improved by making the system able to encrypt files of formats other than .txt. As this project was performed on a local system with limited computing power, file formats other than .txt would require high computational power. Optimizing the encryption and revocation algorithms will not only speed up the encryption and decryption processes, but will also ensure that other file types are supported as well. Another enhancement which we can make is to further automate the system to revocate the file upon download. This will ensure that the data is safe and encrypted at all time periods. Further use and enhancement of this project could include using the encryption environment to support social media platforms, as in users wishing to share data with one another could use this encryption-system to personally guarantee confidentiality of their data while still making use of the social media platform.

REFERENCES

[1] C. Wang, Q. Wang, K. Ren, and W. Lou, “Ensuring Data Storage Security in Cloud Computing,” Proc. 17th Int’l Workshop Quality of Service (IWQoS ’09), pp. 1-9, July 2009.

[2] Wang Cong, Wang Qian, RenKui, Cao Ning and Lou Wenjing , "Toward Secure and Dependable Storage Services in Cloud Computing," Services Computing, IEEE Transactions on , vol.5, no.2, pp.220-232, April-June 2012.

[3] Hsiao-Ying Lin; Tzeng, W.-G.; "A Secure Erasure CodeBased Cloud Storage System with Secure Data Forwarding," Parallel and Distributed Systems, IEEE Transactions on , vol.23, no.6, pp.995-1003, June 2012.

[4] Zhiguo Wan; Jun'e Liu; Deng, R.H.; "HASBE: A Hierarchical Attribute-Based Solution for Flexible and Scalable Access Control in Cloud Computing," Information Forensics and Security, IEEE Transactions on , vol.7, no.2, pp.743-754, April 2012.

[5] A. Shamir, “Identity-based cryptosystems and signature schemes,” in Advances in cryptology. Springer, 1985, pp. 47–53.

[6] D. Boneh and M. Franklin, “Identity-based encryption from the weil pairing,” SIAM Journal on Computing, vol. 32, no. 3, pp. 586– 615, 2003.

[7] “Advanced Encryption Standard(AES)”, https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[8] “Symmetric Key Algorithm”, https://en.wikipedia.org/wiki/Symmetric-key_algorithm

[9] “Symmetric vs. Asymmetric Encryption”, https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences

[10] “Unified Modeling Language”, https://en.wikipedia.org/wiki/Unified_Modeling_Language