A Project Report

On

Secure File Storage On Cloud Using Hybrid Cryptography

Submitted in partial fulfilment of the requirement of

University of Mumbai

For the Degree of

Bachelor of Engineering

in

COMPUTER ENGINEERING

Submitted by

Shruti Kanatt
Amey Jadhav
Prachi Talwar

Supervised by

Mrs. Rakhi Kalantri

Department of Computer Engineering

Fr. Conceicao Rodrigues Institute of Technology
Sector 9A, Vashi, Navi Mumbai - 400703

UNIVERSITY OF MUMBAI

2019-2020

APPROVAL SHEET

This is to certify that the project entitled

“Secure File Storage On Cloud Using Hybrid Cryptography”

Submitted by

Shruti Kanatt	**(101622)**
Amey Jadhav	**(101665)**
Prachi Talwar	**(101676)**

Supervisors : ____________________

Project Coordinator : ____________________

Examiners : 1. ____________________

2. ____________________

Head of Department : ____________________

Date :

Place :

Declaration

We declare that this written submission for B.E. Declaration entitled "**Secure File Storage On Cloud Using Hybrid Cryptography**" represent our ideas in our own words and where others' ideas or words have been included. We have adequately cited and referenced the original sources. We also declared that we have adhere to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any ideas / data / fact / source in our submission. We understand that any violation of the above will cause for disciplinary action by institute and also evoke penal action from the sources which have thus not been properly cited or from whom paper permission have not been taken when needed.

Project Group Members:

1. Shruti Kanatt, 101622

2. Amey Jadhav, 101665

3. Prachi Talwar, 101676

Abstract

Cloud computing is a term that has gained widespread use over the last few years. With the exponential increase in data use that has accompanied society's transition into the digital 21st century, it is becoming more and more difficult for individuals and organizations to keep all of their vital information, programs, and systems up and running on in-house computer servers. The solution to this problem is cloud computing and storage. One big advantage that cloud storage provides is data security, and thus the aim of our project is to increase the security of cloud storage using hybrid cryptography. Our project consists of a system that allows users to safely upload and download files from the cloud, and additionally, to show the comparative study of the existing system, comprising of traditional encryption algorithms (RSA and AES), versus the hybrid cryptography algorithm (which uses AES and blowfish algorithms).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Simply put, cloud computing is the delivery of computing services — including servers, storage, databases, networking, software, analytics, and intelligence—over the Internet ("the cloud") to offer faster innovation, flexible resources, and economies of scale. Cloud storage is a model of computer data storage in which the digital data is stored in logical pools. The physical storage spans multiple servers (sometimes in multiple locations), and the physical environment is typically owned and managed by a hosting company. These cloud storage providers are responsible for keeping the data available and accessible, and the physical environment protected and running. People and organizations buy or lease storage capacity from the providers to store user, organization, or application data. Cloud storage services may be accessed through a collocated cloud computing service, a web service application programming interface (API) or by applications that utilize the API, such as cloud desktop storage, a cloud storage gateway or Web-based content management systems.

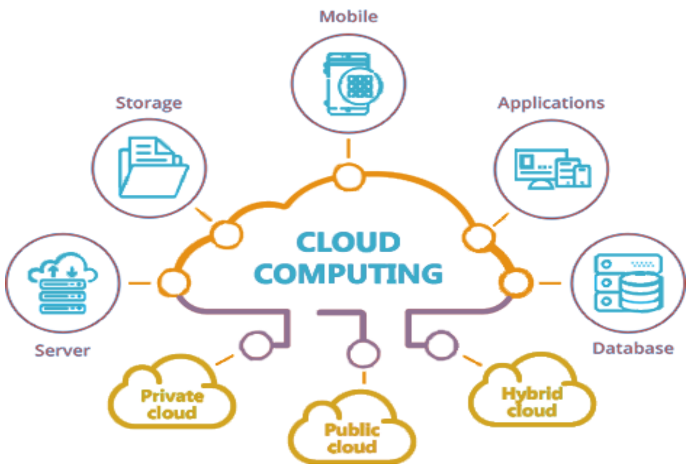


Figure 1.1: Cloud Computing

1.2 Motivation

Cloud storage is a cloud computing model in which data is stored on remote servers accessed from the internet, or ”cloud.” It is maintained, operated and managed by a cloud storage service provider on a storage servers that are built on virtualization techniques. Cloud storage is also known as utility storage – a term subject to differentiation based on actual implementation and service delivery. The advantages of cloud storage are as follows:
1. Total Cost of Ownership: With cloud storage, there is no hardware to purchase, storage to provision, or capital being used for ”someday” scenarios. You can add or remove capacity on demand, quickly change performance and retention characteristics.
2. Time to Deployment: When development teams are ready to execute, infrastructure should never slow them down. Cloud storage allows IT to quickly deliver the exact amount of storage needed, right when it’s needed. This allows IT to focus on solving complex application problems instead of having to manage storage systems.
3. Information Management: Centralizing storage in the cloud creates a tremendous leverage point for new use cases. By using cloud storage lifecycle management policies, you can perform powerful information management tasks including automated tiering or locking down data in support of compliance requirements.
4. Improved Security: One of the major concerns of every business, regardless of size and industry, is the security of its data. Data breaches and other cybercrimes can devastate a company’s revenue, customer loyalty and brand positioning. Cloud offers many advanced security features that guarantee that data is securely stored and handled. Cloud storage providers implement baseline protections for their platforms and the data they process. From there, most enterprises supplement these protections with added security measures of their own to bolster cloud data protection and tighten access to sensitive information in the cloud.

1.3 Aim and Objective

While keeping up with increasing popularity of could computing environments, the security issues introduced through adaptation of this technology are also increasing. Though Cloud Computing offers many benefits, it is vulnerable to attacks. Attackers are consistently trying to find loopholes to attack the cloud computing environment. The traditional security mechanisms which are used are reconsidered because of these cloud computing deployments. Hence the aim of this project is to increase efficient encryption management to increase the reliability and the rate of security in cloud computing environments.

Due to the advancement of technology in our society digital images and videos are playing a major role than just plain and simple text, thus demanding a serious protection of user privacy. In order to make everything secure, the encryption of audio and video is very important as it helps to minimize malicious attacks from unauthorized parties. The recent advancement in science and technology, mainly in the computer and communication industry, allows potentially a huge market for distributing digital multimedia content through the internet. However, the rapid increase of digital document, multimedia processing tools and the availability of internet access throughout the world have given birth to a perfect environment for copyright fraud and uncontrollable distribution of multimedia content. One of the major challenges currently is the protection of intellectual content in multimedia networks. Hence the objective of this project is to implement a double stage encryption algorithm for the security of multimedia contents against a negligent third party and side channel attack. The proposed randomly generated key algorithm produces a unique symmetric key every time that lets the data be encrypted successfully.

1.4 Report Outline

This report highlights the theoretical concept of the project, including the background information and objective. This report also explains the outline of the working of the system along with the knowledge of the theory of the working of the different encryption algorithms.

Chapter 2

Study of the Secure File Storage On Cloud Using Hybrid Cryptography System

2.1 Cloud Computing

Cloud Computing is the practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer. Cloud computing is the trending technology that uses the network to provide service to the user. The consumers need no concern where the hardware and software or the application is operating, they only need to have a simple device that can simply operate with the cloud.The consumers pay much less for the cloud and have no maintenance liabilities. Cloud computing allows the business user or individual user to use the application through internet without installing in their system. For example: Gmail, Facebook, YouTube, drop box.The main advantage of cloud computing is low cost, increased storage and flexibility.The major risk in cloud computing is security and privacy (i.e. by putting the valuable data on someone else's server in an unknown location).

2.2 Cloud Computing Services

The cloud is the combination of three potential services.The three key cloud services are termed as: Infrastructure-as-a- Service (IaaS), Platform-as-a-Service (PaaS) and Softwareas- a- Service (SaaS).In IaaS, the cloud provider supplies Virtual Machines (VMs) and storage on which consumers can build and run applications.Virtual Machines (VMs) are a set of virtualized infrastructural components.The PaaS offers third party services like operating system, programming environments and web servers, whereas SaaS offers different application software. The four popular cloud deployment models for the consumers are: private cloud , public cloud ,hybrid cloud and community cloud. Public cloud environments are accessible by multiple renters, whereas the private clouds are decorated with virtual resources for particular organization only. Hybrid cloud is the composition of public and private cloud and community cloud is dedicated to several groups.

2.3 Cyber Attack on Cloud

The cyber attack causes various serious harm to cloud users.The main aim of cyber attacks on cloud computing to gain access to user data and to cloud service.The user will store sensitive information in the cloud.The cloud service provider wants to take necessary step to protect data in the cloud.The provider must ensure that their infrastructure is secure and that their clients' data and applications are protected, while the user must take measures to fortify their application and use strong passwords and authentication measures. A number of security threats are associated with cloud data services: not only traditional security threats, such as network eavesdropping, illegal invasion, and denial of service attacks, but also specific cloud computing threats, such as side channel attacks, virtualization vulnerabilities, and abuse of cloud services. The following security requirements limit the threats: Data confidentiality is the property that data contents are not made available or disclosed to illegal users. Outsourced data is stored in a cloud and out of the owners' direct control. Only authorized users can access the sensitive data while others should not gain any information of the data. Access controllability means that a data owner can perform the selective restriction of access to their data outsourced to the cloud. Legal users can be authorized by the owner to access the data, while others can not access it without permission. Data integrity demands maintaining and assuring the accuracy and completeness of data. A data owner always expects that her or his data in a cloud can be stored correctly and trustworthily. It means that the data should not be illegally tampered, improperly modified, deliberately deleted, or maliciously fabricated.

2.4 Cloud Storage Security

2.4.1 Cryptography

In computer science, cryptography refers to secure information and communication techniques derived from mathematical concepts and a set

of rule-based calculations called algorithms to transform messages in ways that are hard to decipher.
Cryptography is the process of writing the secret information in human unreadable secret format. Encrypt the plaintext into ciphertext by using the secret key which cannot be readable by an unauthorized person and transfer the cipher text between the parties on an insecure channel. After the data is received at the receiver side the cipher text is decrypted using the valid secret key and retrieves the original message. Without the knowledge of a secret key, the attacker cannot retrieve the secret message.

2.4.2 Symmetric/Private Key Cryptography

This is the simplest kind of encryption that involves only one secret key to cipher and decipher information. Symmetrical encryption is an old and best-known technique. It uses a secret key that can either be a number, a word or a string of random letters. It is blended with the plain text of a message to change the content in a particular way. The sender and the recipient should know the secret key that is used to encrypt and decrypt all the messages. Blowfish, AES, RC4, DES, RC5, and RC6 are examples of symmetric encryption. The most widely used symmetric algorithm is AES-128, AES-192, and AES-256.

2.4.3 Asymmetric/Public Key Cryptography

Asymmetrical encryption is also known as public key cryptography, which is a relatively new method, compared to symmetric encryption. Asymmetric encryption uses two keys to encrypt a plain text. Secret keys are exchanged over the Internet or a large network. It ensures that malicious persons do not misuse the keys. It is important to note that anyone with a secret key can decrypt the message and this is why asymmetric encryption uses two related keys to boosting security. A public key is made freely available to anyone who might want to send you a message. The second private key is kept a secret so that you can only know. Asymmetric encryption is mostly used in day-to-day communication channels, especially over the Internet. Popular asymmetric key encryption algorithms include EIGamal, RSA, DSA, ELLIPTIC CURVE, PKCS.

2.5 AES Algorithm

The Advanced Encryption Standard (AES) is a symmetric-key block cipher algorithm. The AES has three fixed 128-bit block ciphers with cryptographic key sizes of 128, 192 and 256 bits. Key size is unlimited, whereas the block size maximum is 256 bits. The AES design is based on a substitution-permutation network (SPN) and does not use the Data Encryption Standard (DES) Feistel network.

1. Key Expansions:

   Round keys are derived from the cipher key using AES key schedule. AES requires a separate 128-bit round key block for each round plus one more.

2. Initial Round:

   Add Round Key—each byte of the state is combined with a block of the round key using bitwise xor.

3. Rounds:

   (a) Sub Bytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.

   (b) Shift Rows—a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.

   (c) Mix Columns—a mixing operation which operates on the columns of the state, combining the four bytes in each column.

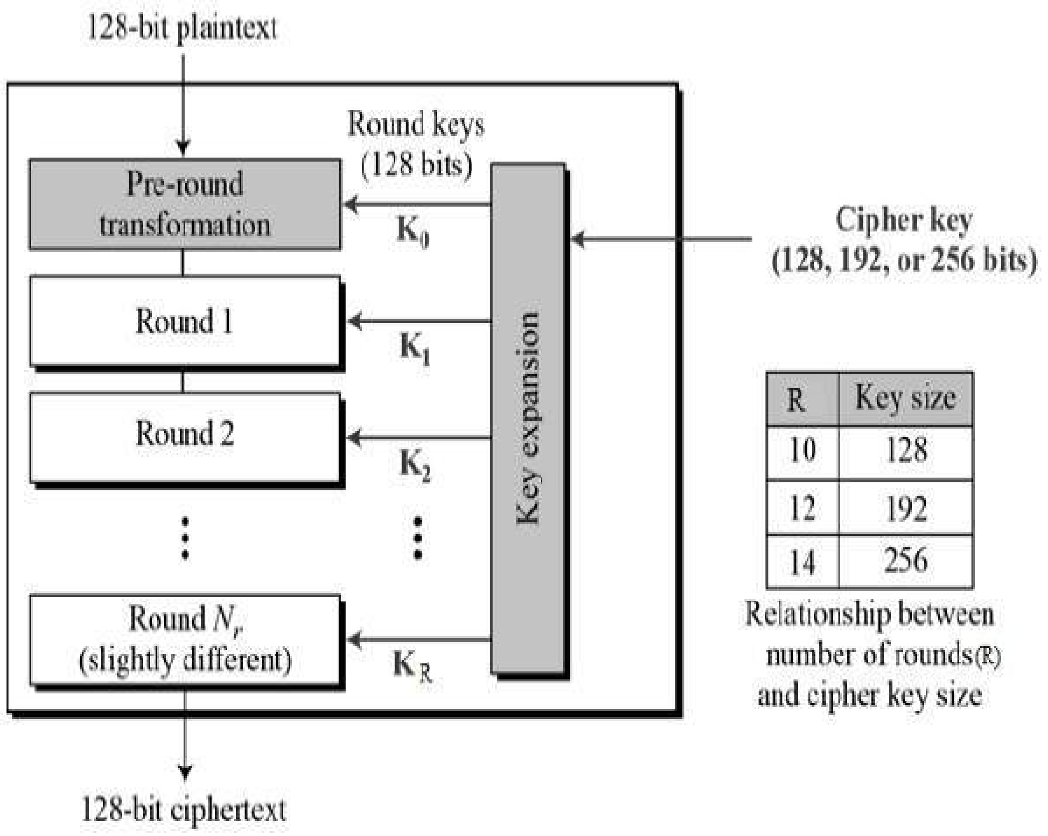
   (d) Add RoundKey

4. Final Round (no Mix Columns):

   (a) Sub Bytes

   (b) Shift Rows

   (c) Add RoundKey



R	Key size
10	128
12	192
14	256

Figure 2.1: AES Algorithm

2.6 Blowfish Algorithm

Blowfish is a symmetric block encryption algorithm designed in consideration with:

- Fast: It encrypts data on large 32-bit microprocessors at a rate of 26 clock cycles per byte.
- Compact: It can run in less than 5K of memory.
- Simple: It uses addition, XOR, lookup table with 32-bit operands.
- Secure: The key length is variable, it can be in the range of 32 448 bits: default 128 bits key length.
- It is suitable for applications where the key does not change often, like communication link or an automatic file encryptor.
- Unpatented and royalty-free.

Description of Algorithm:
Blowfish symmetric block cipher algorithm encrypts block data of 64-bits at a time.it will follows the feistel network and this algorithm is divided into two parts.

1. Key-expansion
2. Data Encryption

Key-expansion:

It will convert a key of at most 448 bits into several sub key arrays totaling 4168 bytes. Blowfish uses large number of sub keys.
These keys are generated earlier to any data encryption or decryption.
The p-array consists of 18, 32-bit sub keys:
P1,P2,. . . . .,P18
Four 32-bit S-Boxes consists of 256 entries each:
S1,0, S1,1,. S1,255
S2,0, S2,1,. S2,255
S3,0, S3,1,. S3,255
S4,0, S4,1,.............S4,255

Generating the Sub keys:

The sub keys are calculated using the Blowfish algorithm:

1. Initialize first the P-array and then the four S-boxes, in order, with a fixed string. This string consists of the hexadecimal digits of pi (less the initial 3): P1 = 0x243f6a88, P2 = 0x85a308d3, P3 = 0x13198a2e, P4 = 0x03707344, etc. 2. XOR P1 with the first 32 bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P14). Repeatedly cycle through the key bits until the entire P-array has been XO Red with key bits. (For every short key, there is at least one equivalent longer key; for example, if A is a 64-bit key, then AA, AAA, etc., are equivalent keys.) 3. Encrypt the all-zero string with the Blowfish algorithm, using the sub keys described in steps (1) and (2). 4. Replace P1 and P2 with the output of step (3). 5. Encrypt the output of step (3) using the Blowfish algorithm with the modified sub keys. 6. Replace P3 and P4 with the output of step (5). 7. Continue the process, replacing all entries of the P array, and then all four S-boxes in order, with the output of the continuously changing Blowfish algorithm.

In total, 521 iterations are required to generate all required sub keys. Applications can store the sub keys rather than execute this derivation process multiple times.

Data Encryption:

It is having a function to iterate 16 times of network. Each round consists of key-dependent permutation and a key and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookup tables for each round.

<u>Algorithm: Blowfish Encryption</u>

```
Divide x into two 32-bit halves: xL, xR

For i = 1 to 16:
      xL = XL XOR Pi
       xR = F(XL) XOR xR
      Swap XL and xR
      Swap XL and xR (Undo the last swap.)
      xR = xR XOR P17
xL = xL XOR P18
Recombine xL and xR
```

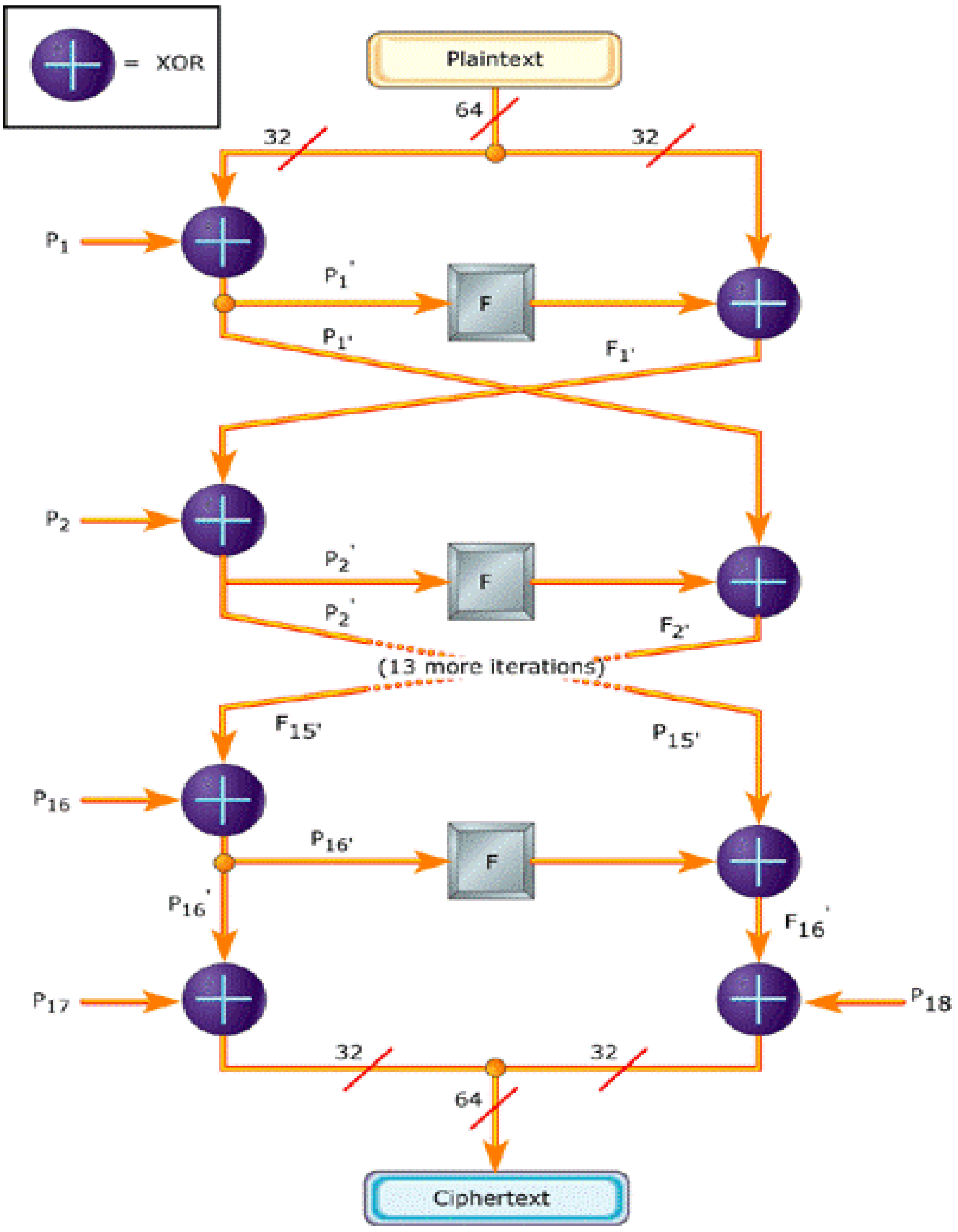


Figure 2.2: Blowfish Algorithm

2.7 RSA Algorithm

The Rivest-Shamir-Adleman (RSA) algorithm is one of the most popular and secure public-key encryption methods. The algorithm capitalizes on the fact that there is no efficient way to factor very large (100-200 digit) numbers.

Using an encryption key (e,n), the algorithm is as follows:

1. Represent the message as an integer between 0 and (n-1). Large messages can be broken up into a number of blocks. Each block would then be represented by an integer in the same range.
2. Encrypt the message by raising it to the eth power modulo n. The result is a ciphertext message C.
3. To decrypt ciphertext message C, raise it to another power d modulo n

The encryption key (e,n) is made public. The decryption key (d,n) is kept private by the user. How to Determine Appropriate Values for e, d, and n:

1. Choose two very large (100+ digit) prime numbers. Denote these numbers as p and q.
2. Set n equal to p * q.
3. Choose any large integer, d, such that GCD(d, ((p-1) * (q-1))) = 1
4. Find e such that e * d = 1 (mod ((p-1) * (q-1)))

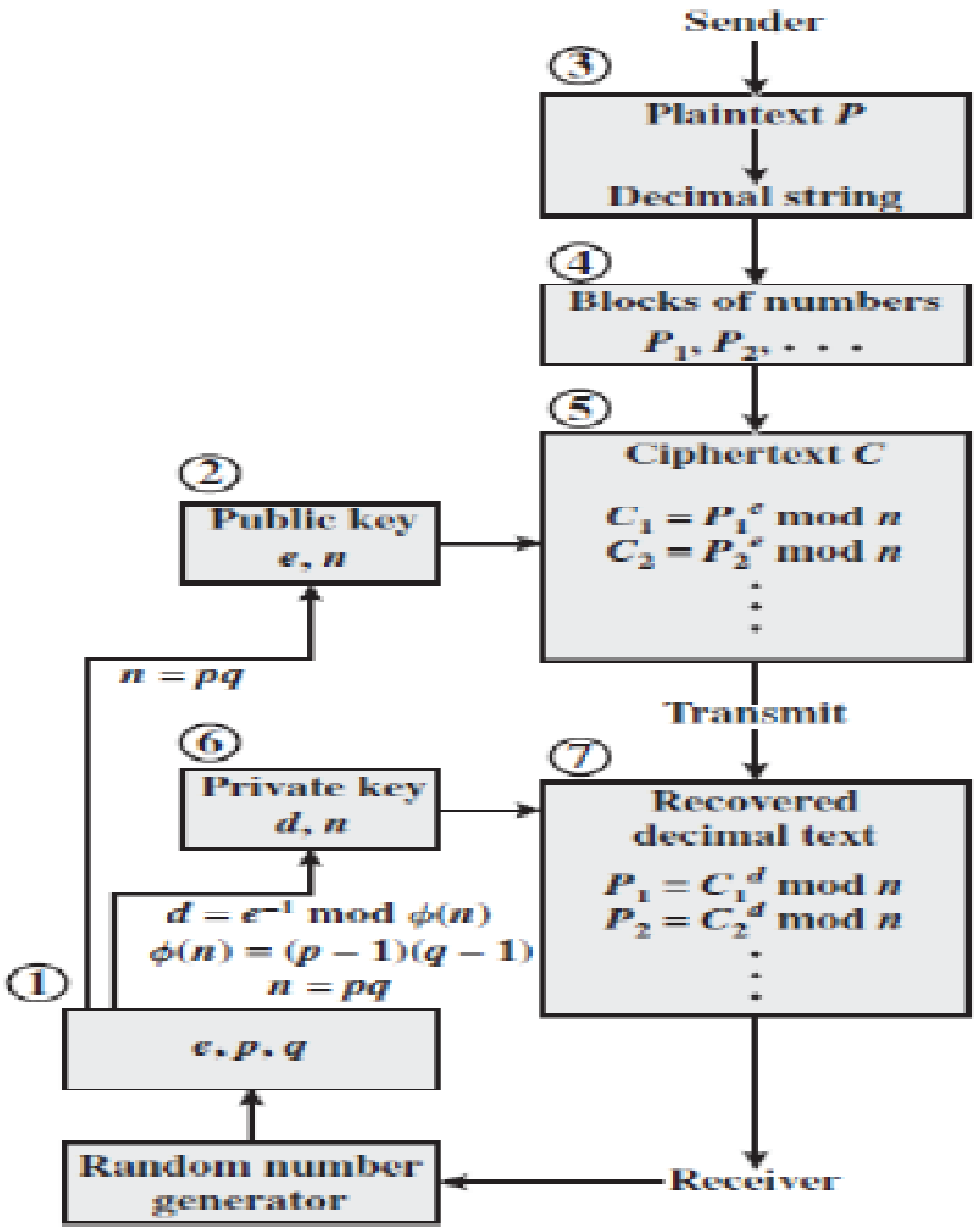


Figure 2.3: RSA Algorithm

2.8 Related Work

The following table describes in brief the methodologies used and limitations of a few related existing systems.

Table 2.1: Existing Systems

Title	**Methodology**	**Limitations**
Secure storage and access of data in Cloud computing	ECC (Elliptic Curve Cryptography) algorithm. Performs authentication, key generation, encryption and decryption	Uses single key for encryption and decryption hence providing less security
Use of Digital Signature with Diffie Hellman key exchange and AES encryption algorithm to enhance Data Security in Cloud Computing	Use of Diffie Hellman for key exchange. Authentication prov ided by Digital Signature scheme and lastly files encrypted using AES	Time consuming procedure as three different steps using different techniques are performed
RSA Encryption and Digital Signature	Use of RSA algorithm in combination with MD5 Digest to ensure data security on cloud	RSA algorithm only provides key encryption and along with MD5 it provides single text encryption and not multiple text encryption
Survey Paper on Cloud Storage Security	Using EFS, NTFS with cache for securing data files by using automatic cryptographic systems inbuilt in EFS	As cryptographic systems are inbuilt in EFS, modifications for providing better security measures is difficult to implement
Secure File Storage and File Sharing	Separate servers are used for input, storage and output functions. Providing better security by keeping separate modules	As three different servers are used there can be connectivity issues as well as synchronizatio n problems

Chapter 3

Proposed System

3.1 Problem Statement

Cloud storage is a very important aspect of cloud computing. Thus security of cloud storage is a very pressing matter, and double encryption using hybrid cryptography is an efficient way of doing so. This is achieved using AES algorithm with Blowfish algorithm to improve cloud security.

3.2 Scope & Proposed Solution

In the proposed scheme, the data owner is responsible for generating updating information and sending them to the cloud server. Thus, the data owner needs to store the encrypted file. In this system hybrid encryption technique is applied on the data file using AES and Blowfish algorithm to securely store file data in cloud. There is no server admin in between user and owner, user view file and request for downloading this file, this request is directly sent to the owner of the file; if the owner accepts this user file then the can download it, but if the owner discards this request, the user can't download this file. Data in form of text, image and video proposes different challenges in encryption and decryption.

Hybrid Cryptography concept is used for securing storage system of cloud. Two different approaches are used to show the difference between less secure and more secure systems. The first approach uses RSA and AES algorithms; RSA is used for key encryption and AES is used for text or data encryption. In the second or we can say more secured approach, AES and Blowfish algorithms are used. In this approach, these two algorithms provide double encryption over data and key which provides higher security compared to the first one.

Additionally, share feature is also provided to enable users to share files amongst themselves. The owner of the file who wishes to share the file sends an intimation to the user with whom they want to share the file. When the latter receives this intimation, they send a request to the former. They now authorise the share, and so the file is now available to the latter as well for access.

Furthermore, duplication detection is also available in the proposed system. Each file is given a unique hash value. When a file is chosen to be uploaded onto the cloud, it is first checked if one with the same hash value is already present in the cloud. If ye, then duplication is detected. Hence, this helps save storage space and removes redundancy.

Chapter 4

Design of the Secure File Storage On Cloud Using Hybrid Cryptography System

4.1 Requirement Engineering

4.1.1 Requirement Elicitation

Our system requires a portal with cloud access for uploading files to cloud and downloading them. To operate the portal a laptop with good network connection is required. We require a cloud API which will facilitate the storage of files. The files are encrypted and then uploaded on cloud and then the user can download the decrypted file by entering the decryption key. The user needs a registered email-id for receiving the decryption key for encrypting the file. Once decrypted, the user can download the file.

4.1.2 Software Lifecycle Model

Software Development Life Cycle, SDLC for short, is a well-defined, structured sequence of stages in software engineering to develop the intended software product. SDLC provides a series of steps to be followed to design and develop a software product efficiently. The development of the project for this semester was spanned over a duration of 6 months. In order to effectively design this system, the Waterfall model was practiced. Our goal of creating a comprehensive security system consisting of six general phases was made possible using this model since t requirements were very well documented, clear and fixed, product definition was stable, technology is understood and is dynamic, there were no ambiguous requirements, and ample resources with required expertise were available to support the product.

4.1.3 Requirement Analysis

Requirements Analysis is the process of defining the expectations of the users for an application that is to be built or modified. Requirements analysis involves all the tasks that are conducted to identify the needs of different stakeholders. Therefore requirements analysis means to analyze, document, validate and manage software or system requirements. Thus requirement analysis for our project was done at the beginning of last semester. No new requirements were added for this semester.

4.1.3.1 Activity Diagrams

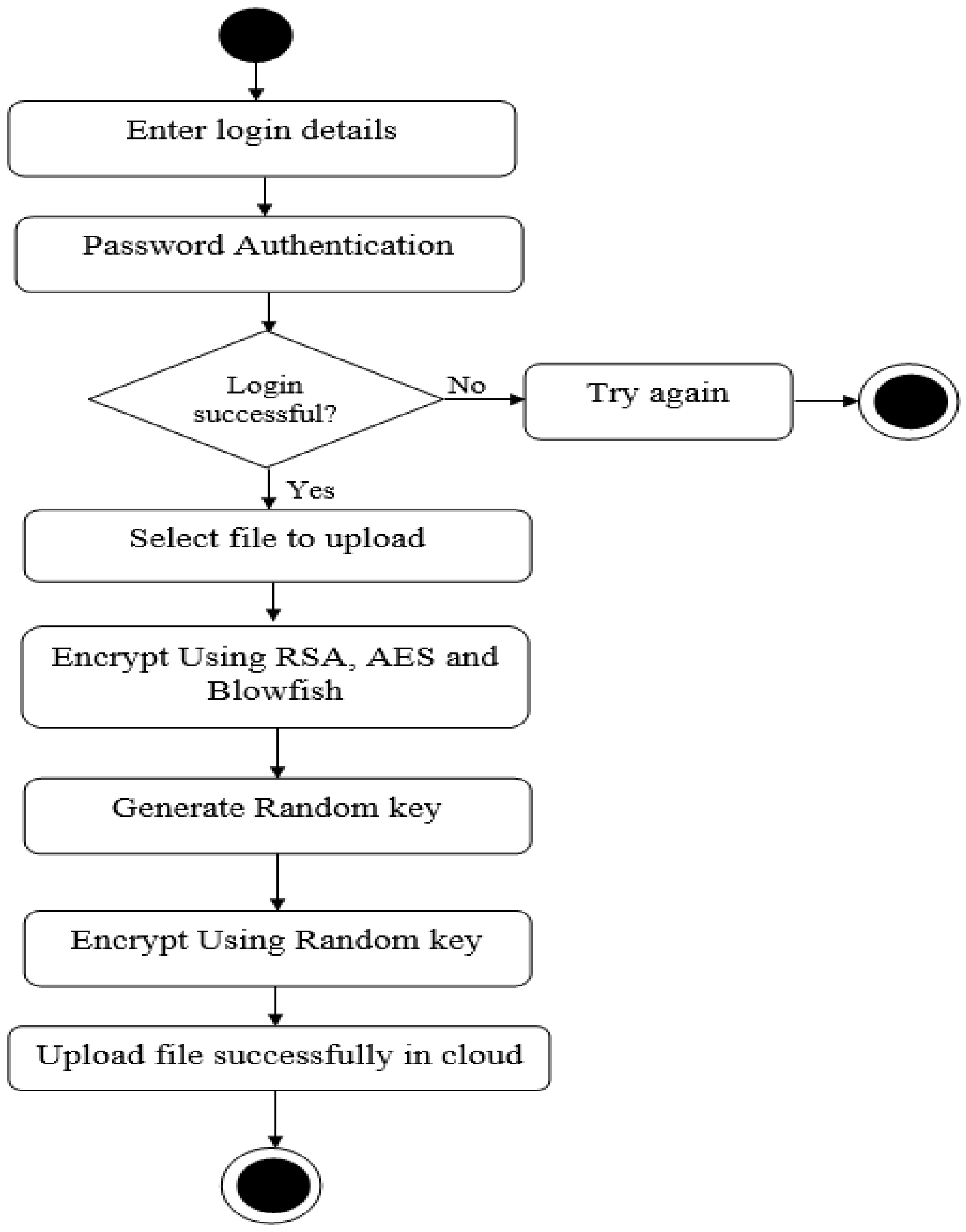


Figure 4.1: Activity Diagram for Upload

The user logs in to the system, and if the login is successful then the user can select a file to be uploaded, which will be encrypted using either AES and RSA combination or AES and Blowfish hybrid algorithm combination. This is done using a randomly generated key. The file is thus successfully uploaded onto the cloud.

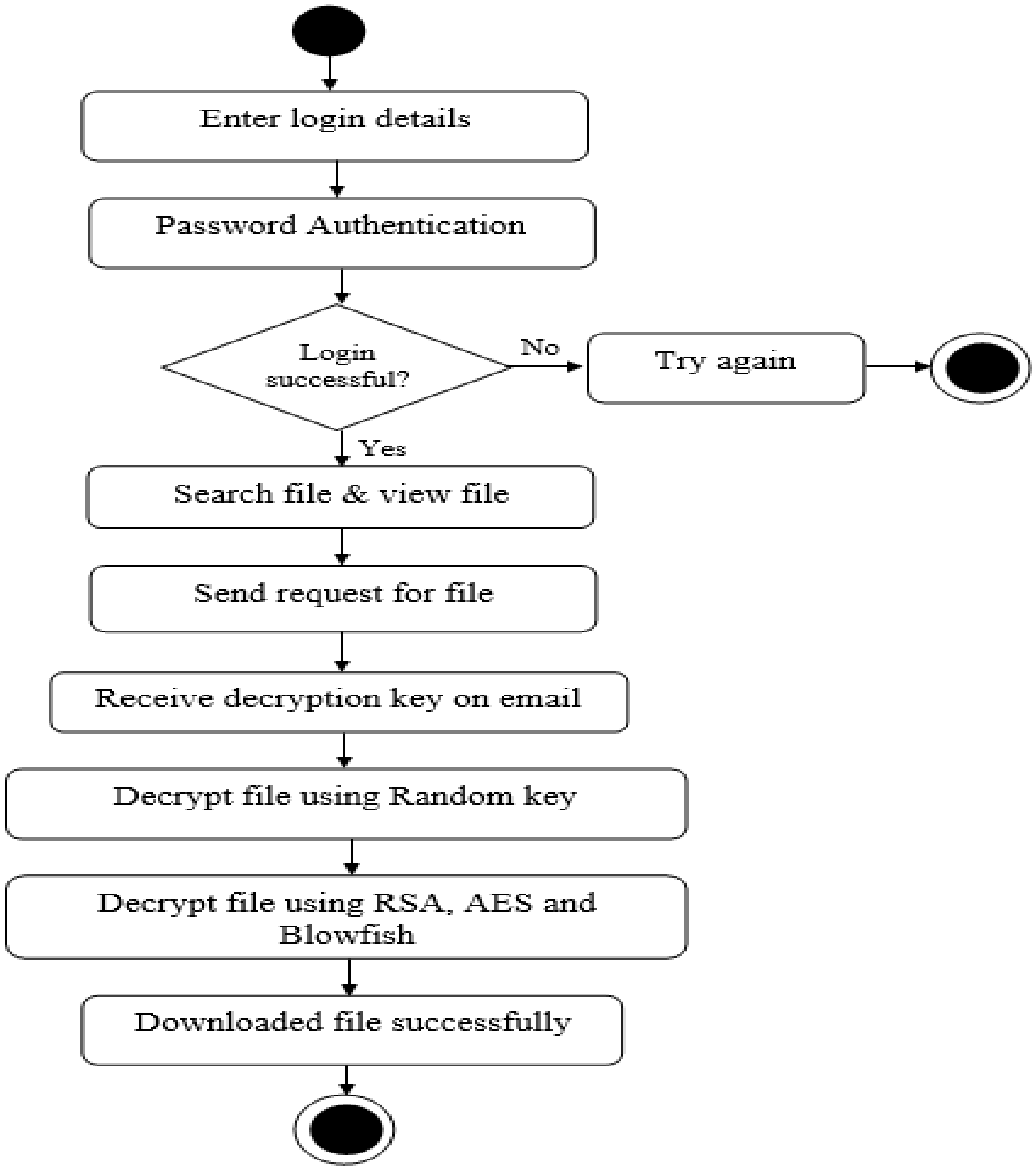

Enter login details
Password Authentication
Login successful?
No
Try again
Yes
Search file & view file
Send request for file
Receive decryption key on email
Decrypt file using Random key
Decrypt file using RSA, AES and Blowfish
Downloaded file successfully

Figure 4.2: Activity Diagram for Download

The user logs in to the system, and if the login is successful then the user can view their uploaded files. Upon selecting the file to be downloaded, the decryption key is sent to the registered email address. Using this key, the file is decrypted using AES and RSA combination or AES and Blowfish hybrid algorithm combination. The original file is thus downloaded.

4.1.3.2 Use Case Diagram

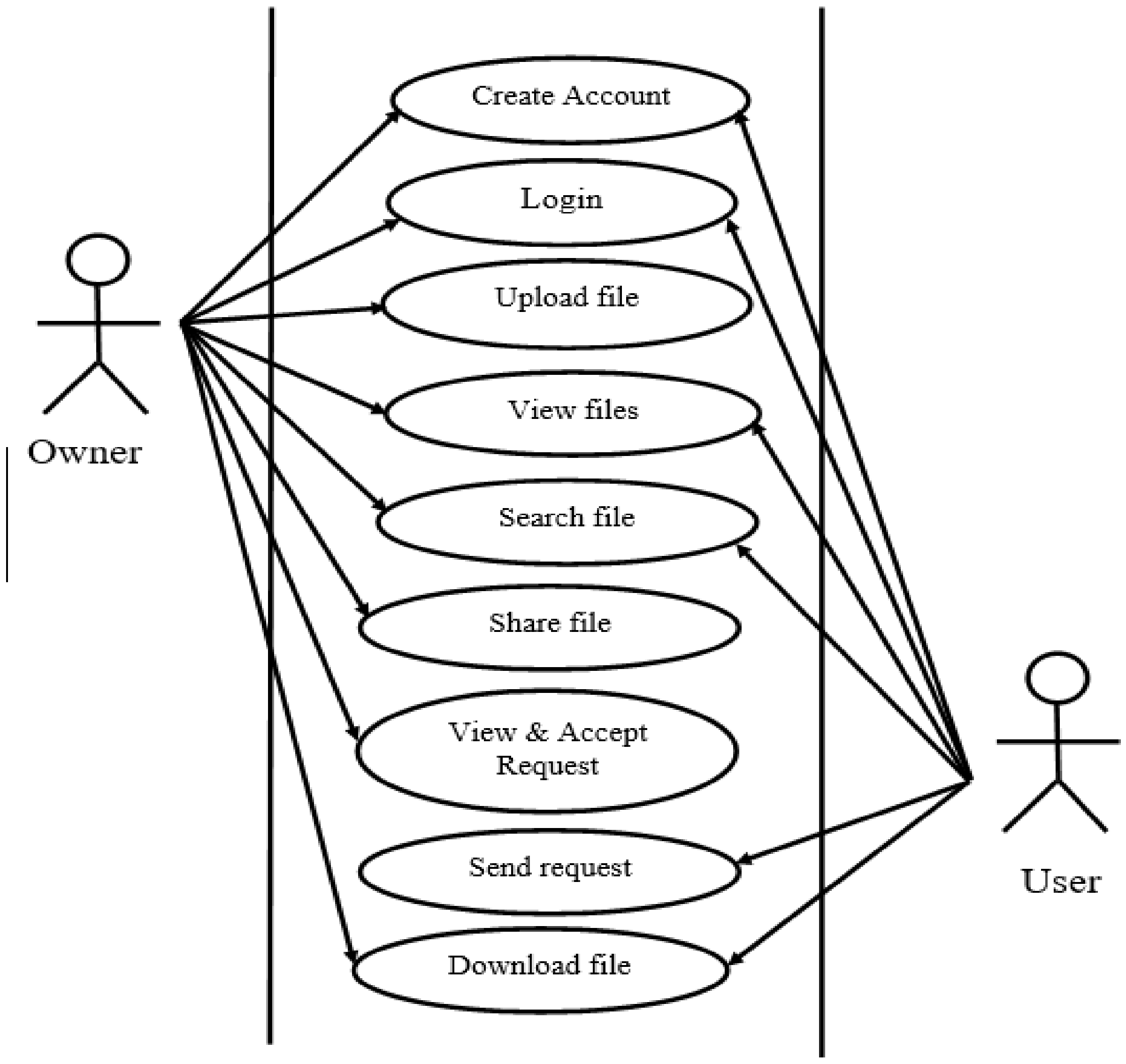


Figure 4.3: Use Case Diagram

The user can perform the following functions: create an account, login to their account, upload files, view their uploaded files, share their files with other users, view and accept share requests, and download files from the cloud.

4.1.3.3 Cost Analysis

Table 4.1: Cost Analysis

WBS Items	**#Units /Hrs.**	**Cost/ Unit/Hr.**	**Subtotals**	**WBS Level 1 Totals**	**% of Total**
1. Project Management Project Team Members	160	Rs 250	Rs 8,000	Rs 8,000	44.4%
2. Hardware Hardware Devices	3	Rs 100	Rs 300	Rs 300	1.7%
3. Software Software Development			Rs 5,000	Rs 5,000	27.8%
4. Testing (10% of total hardware and software costs)			Rs 530	Rs 530	2.9%
5. Support Project Team Members	9	Rs 20	Rs 180	Rs 180	1.0%
6. Online Processing			Rs 1,000	Rs 1,000	5.5%
7. Reserves (20% of total estimate)			Rs 3,002	Rs 3,002	16.7%
Total Project Estimate				Rs 18,012	

4.1.3.4 Hardware and Software Requirements

- Hardware Requirements:
  1. Processor: Pentium 4 or faster
  2. RAM: 512 MB or more
  3. Hard disk: 16 GB or more
- Software Requirements:
  1. JAVA JDK 7
  2. Apache tomcat Server
  3. Eclipse
  4. Windows Operating System
  5. MySQL

4.2 System Architecture

4.2.1 UI/UX diagram

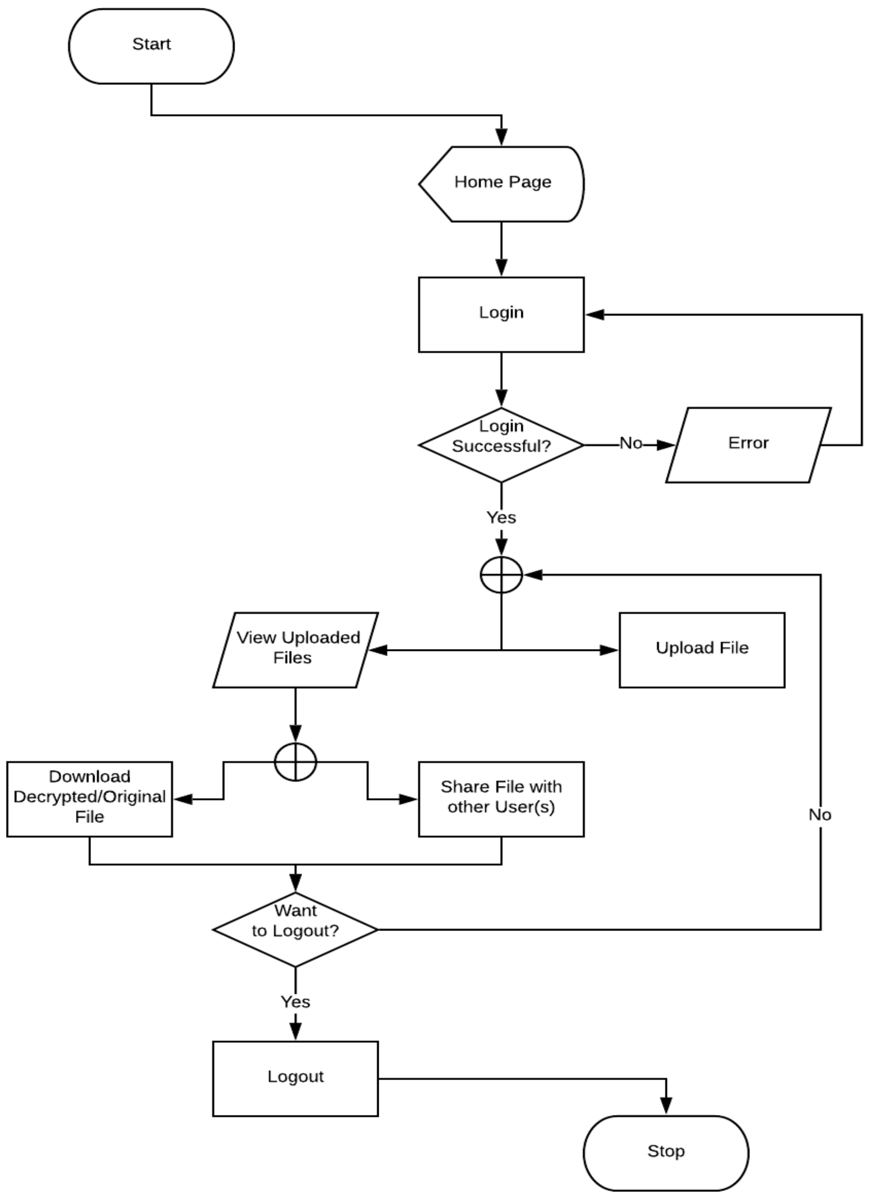


Figure 4.4: Flowchart

The user is taken to the home page on opening the portal, from where they can go to the login page. Upon successful login, user can either upload files onto the cloud or view already uploaded files. From here, the user can either download the decrypted or original file or share files with other users. Finally, the user can logout out of their account on the portal.

4.2.2 Block Diagram

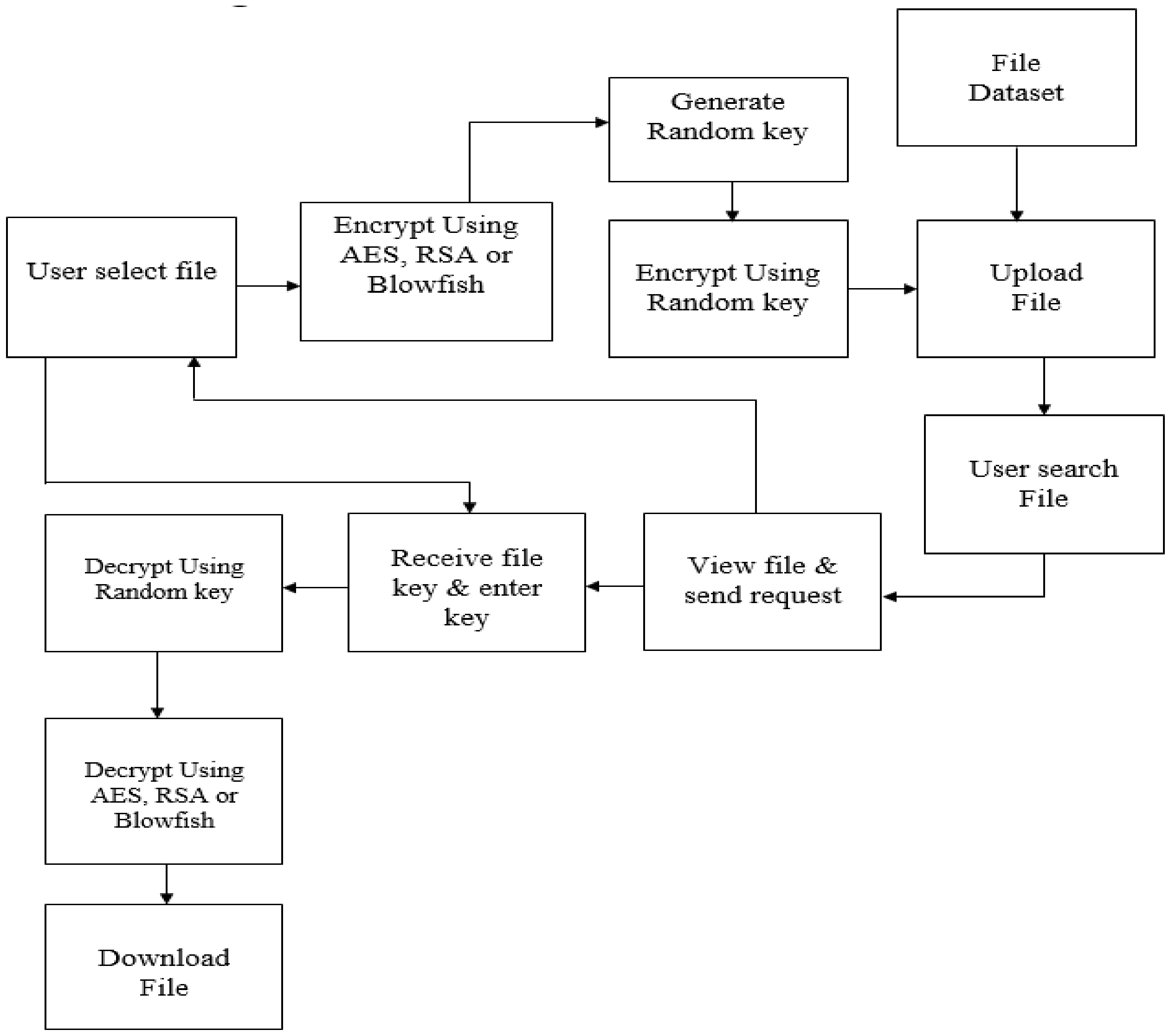


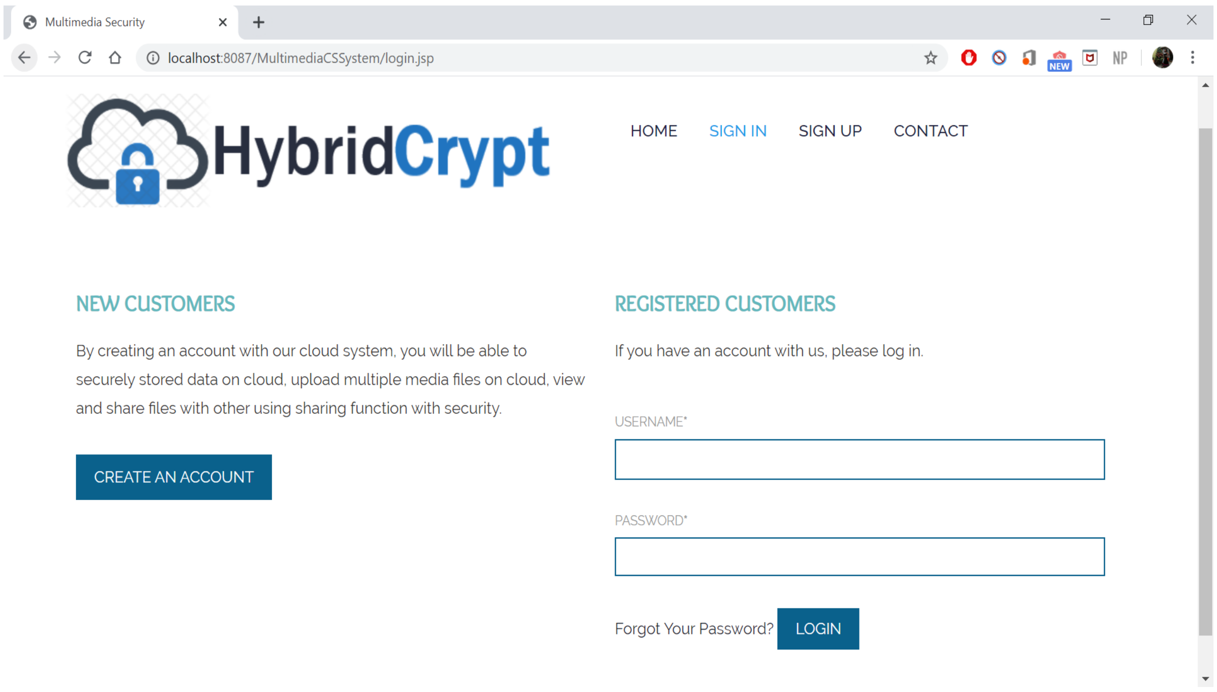
Figure 4.5: Block Diagram

The block diagram shows in schematic form the general arrangement of the parts or components of the system. The user logs in to the system, and if the login is successful then the user can select a file to be uploaded, which will be encrypted using either AES and RSA combination or AES and Blowfish hybrid algorithm combination. This is done using a randomly generated key. The file is thus successfully uploaded onto the cloud. The user can also view their uploaded files. Upon selecting the file to be downloaded, the decryption key is sent to the registered email address. Using this key, the file is decrypted using AES and RSA combination or AES and Blowfish hybrid algorithm combination. The original file is thus downloaded.

Chapter 5

Result and Discussion

5.1 Screenshots of the System



NEW CUSTOMERS

By creating an account with our cloud system, you will be able to securely stored data on cloud, upload multiple media files on cloud, view and share files with other using sharing function with security.

REGISTERED CUSTOMERS

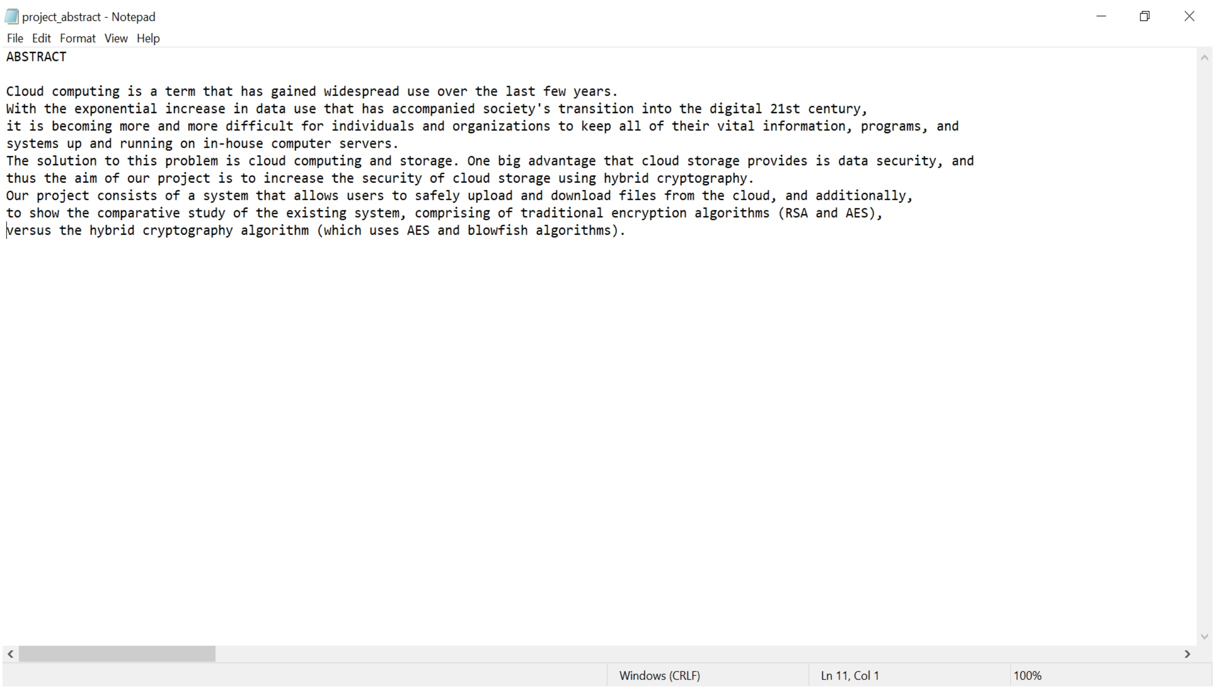
If you have an account with us, please log in.

USERNAME*

PASSWORD*

Forgot Your Password?

Figure 5.1: Login Page



```
ABSTRACT

Cloud computing is a term that has gained widespread use over the last few years.
With the exponential increase in data use that has accompanied society's transition into the digital 21st century,
it is becoming more and more difficult for individuals and organizations to keep all of their vital information, programs, and
systems up and running on in-house computer servers.
The solution to this problem is cloud computing and storage. One big advantage that cloud storage provides is data security, and
thus the aim of our project is to increase the security of cloud storage using hybrid cryptography.
Our project consists of a system that allows users to safely upload and download files from the cloud, and additionally,
to show the comparative study of the existing system, comprising of traditional encryption algorithms (RSA and AES),
versus the hybrid cryptography algorithm (which uses AES and blowfish algorithms).
```

Figure 5.2: Original File

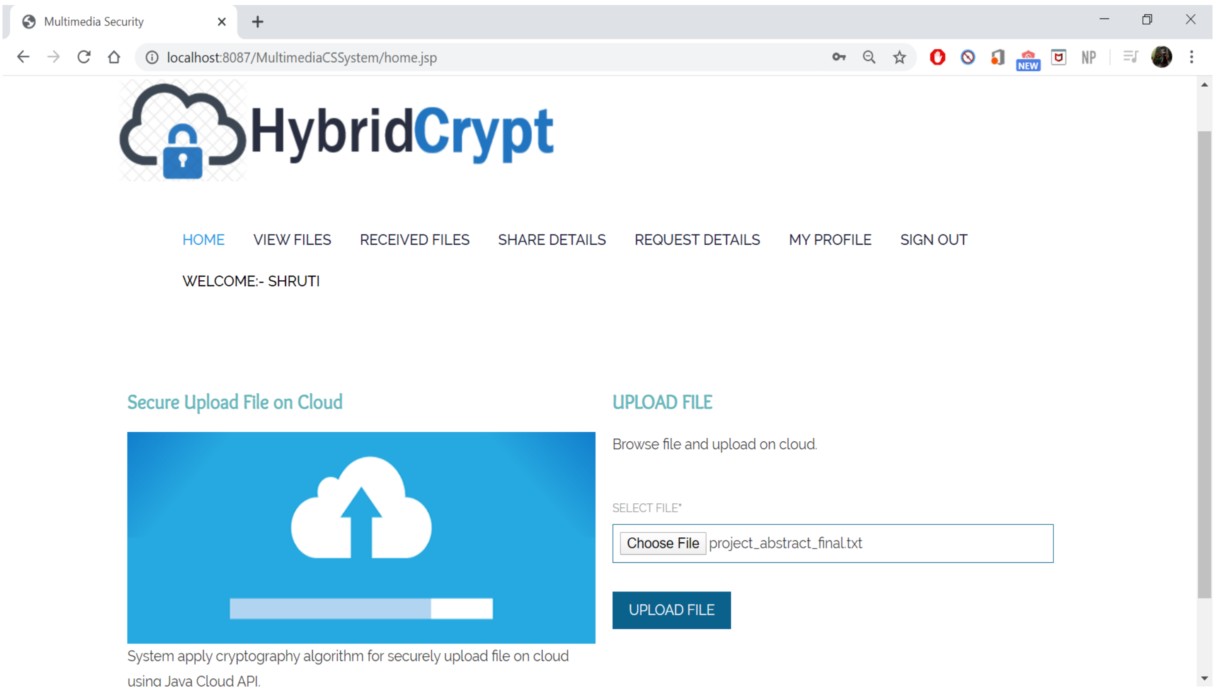
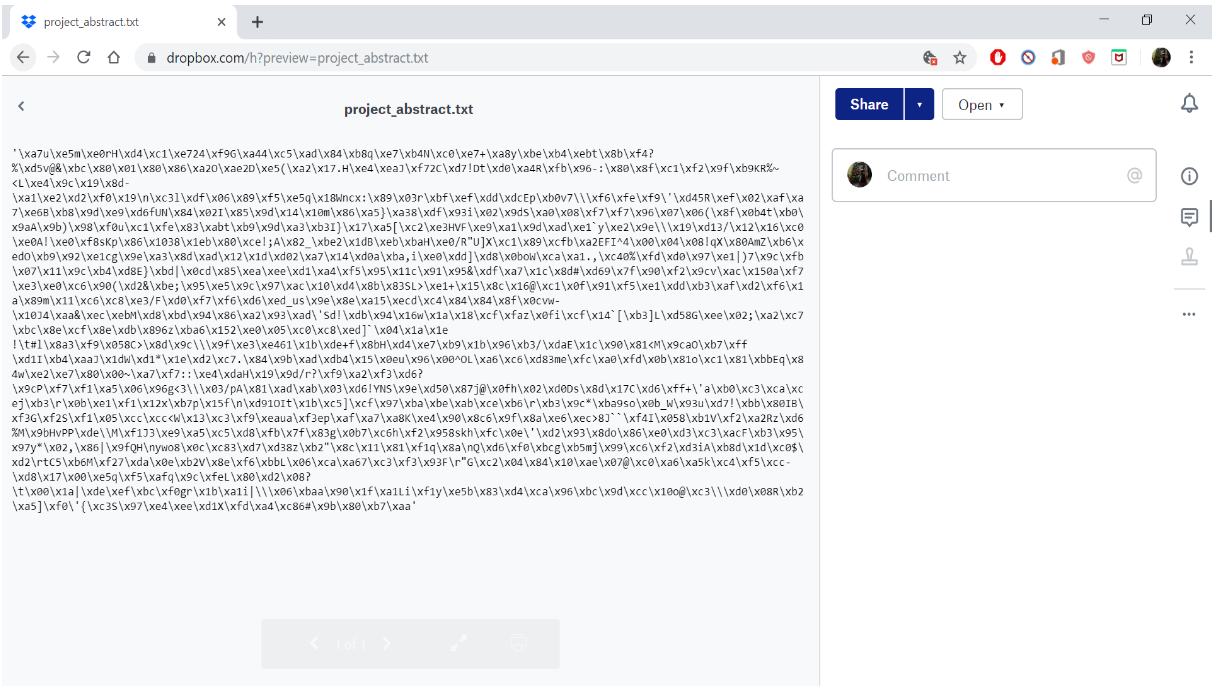
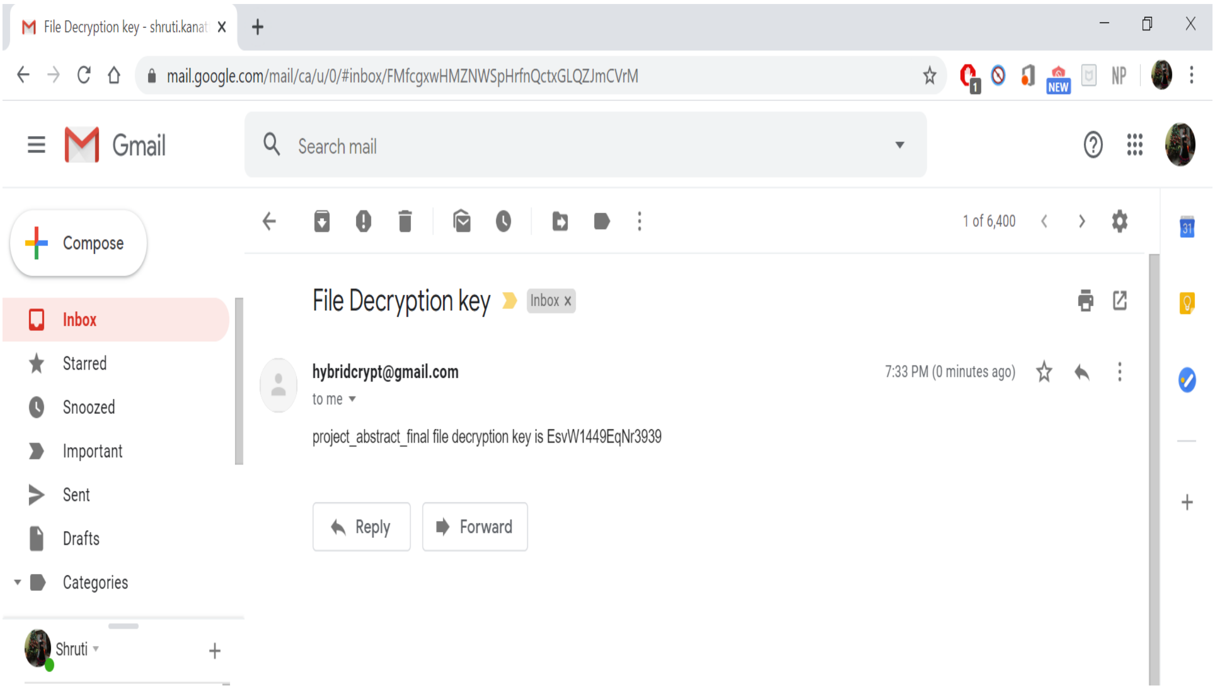

Multimedia Security
localhost:8087/MultimediaCSSystem/home.jsp
HybridCrypt
HOME
VIEW FILES
RECEIVED FILES
SHARE DETAILS
REQUEST DETAILS
MY PROFILE
SIGN OUT
WELCOME:- SHRUTI
Secure Upload File on Cloud
UPLOAD FILE
Browse file and upload on cloud.
SELECT FILE*
Choose File
project_abstract_final.txt
UPLOAD FILE
System apply cryptography algorithm for securely upload file on cloud using Java Cloud API.

Figure 5.3: Upload File



```
'\xa7u\xe5m\xe0rH\xd4\xc1\xe724\xf9G\xa44\xc5\xad\x84\xb8q\xe7\xb4N\xc0\xe7+\xa8y\xbe\xb4\xebt\x8b\xf4?
%\xd5v@&\xbc\x80\x01\x80\x86\xa2O\xae2D\xe5(\xa2\x17.H\xe4\xeaJ\xf72C\xd7!Dt\xd0\xa4R\xfb\x96-:\x80\x8f\xc1\xf2\x9f\xb9KR%~
<L\xe4\x9c\x19\x8d-
\xa1\xe2\xd2\xf0\x19\n\xc3l\xdf\x06\x89\xf5\xe5q\x18Wncx:\x89\x03r\xbf\xef\xdd\xdcEp\xb0v7\\\xf6\xfe\xf9\'\xd45R\xef\x02\xaf\xa
7\xe6B\xb8\x9d\xe9\xd6fUN\x84\x02I\x85\x9d\x14\x10m\x86\xa5}\xa38\xdf\x93i\x02\x9dS\xa0\x08\xf7\xf7\x96\x07\x06(\x8f\x0b4t\xb0\
x9aA\x9b)\x98\xf0u\xc1\xfe\x83\xabt\xb9\x9d\xa3\xb3I}\x17\xa5[\xc2\xe3HVF\xe9\xa1\x9d\xad\xe1`y\xe2\x9e\\\x19\xd13/\x12\x16\xc0
\xe0A!\xe0\xf8sKp\x86\x1038\x1eb\x80\xce!;A\x82_\xbe2\x1dB\xeb\xbaH\xe0/R"U]X\xc1\x89\xcfb\xa2EFI^4\x00\x04\x08!qX\x80AmZ\xb6\x
edO\xb9\x92\xe1cg\x9e\xa3\x8d\xad\x12\x1d\xd02\xa7\x14\xd0a\xba,i\xe0\xdd]\xd8\x0boW\xca\xa1.,\xc40%\xfd\xd0\x97\xe1|)7\x9c\xfb
\x07\x11\x9c\xb4\xd8E}\xbd|\x0cd\x85\xea\xee\xd1\xa4\xf5\x95\x11c\x91\x95&\xdf\xa7\x1c\x8d#\xd69\x7f\x90\xf2\x9cv\xac\x150a\xf7
\xe3\xe0\xc6\x90(\xd2&\xbe;\x95\xe5\x9c\x97\xac\x10\xd4\x8b\x83SL>\xe1+\x15\x8c\x16@\xc1\x0f\x91\xf5\xe1\xdd\xb3\xaf\xd2\xf6\x1
a\x89m\x11\xc6\xc8\xe3/F\xd0\xf7\xf6\xd6\xed_us\x9e\x8e\xa15\xecd\xc4\x84\x84\x8f\x0cvw-
\x10J4\xaa&\xec\xebM\xd8\xbd\x94\x86\xa2\x93\xad\'Sd!\xdb\x94\x16w\x1a\x18\xcf\xfaz\x0fi\xcf\x14`[\xb3]L\xd58G\xee\x02;\xa2\xc7
\xbc\x8e\xcf\x8e\xdb\x896z\xba6\x152\xe0\x05\xc0\xc8\xed]`\x04\x1a\x1e
!\t#l\x8a3\xf9\x058C>\x8d\x9c\\\x9f\xe3\xe461\x1b\xde+f\x8bH\xd4\xe7\xb9\x1b\x96\xb3/\xdaE\x1c\x90\x81<M\x9caO\xb7\xff
\xd1I\xb4\xaaJ\x1dW\xd1*\x1e\xd2\xc7.\x84\x9b\xad\xdb4\x15\x0eu\x96\x00^OL\xa6\xc6\xd83me\xfc\xa0\xfd\x0b\x81o\xc1\x81\xbbEq\x8
4w\xe2\xe7\x80\x00~\xa7\xf7::\xe4\xdaH\x19\x9d/r?\xf9\xa2\xf3\xd6?
\x9cP\xf7\xf1\xa5\x06\x96g<3\\\x03/pA\x81\xad\xab\x03\xd6!YNS\x9e\xd50\x87j@\x0fh\x02\xd0Ds\x8d\x17C\xd6\xff+\'a\xb0\xc3\xca\xc
ej\xb3\r\x0b\xe1\xf1\x12x\xb7p\x15f\n\xd91OIt\x1b\xc5]\xcf\x97\xba\xbe\xab\xce\xb6\r\xb3\x9c*\xba9so\x0b_W\x93u\xd7!\xbb\x80IB\
xf3G\xf2S\xf1\x05\xcc\xcc<W\x13\xc3\xf9\xeaua\xf3ep\xaf\xa7\xa8K\xe4\x90\x8c6\x9f\x8a\xe6\xec>8J``\xf4I\x058\xb1V\xf2\xa2Rz\xd6
%M\x9bHvPP\xde\\M\xf1J3\xe9\xa5\xc5\xd8\xfb\x7f\x83g\x0b7\xc6h\xf2\x958skh\xfc\x0e\'\xd2\x93\x8do\x86\xe0\xd3\xc3\xacF\xb3\x95\
x97y*\x02,\x86|\x9fQH\nywo8\x0c\xc83\xd7\xd38z\xb2"\x8c\x11\x81\xf1q\x8a\nQ\xd6\xf0\xbcg\xb5mj\x99\xc6\xf2\xd3iA\xb8d\x1d\xc0$\
xd2\rtC5\xb6M\xf27\xda\x0e\xb2V\x8e\xf6\xbbL\x06\xca\xa67\xc3\xf3\x93F\r"G\xc2\x04\x84\x10\xae\x07@\xc0\xa6\xa5k\xc4\xf5\xcc-
\xd8\x17\x00\xe5q\xf5\xafq\x9c\xfeL\x80\xd2\x08?
\t\x00\x1a|\xde\xef\xbc\xf0gr\x1b\xa1i|\\\x06\xbaa\x90\x1f\xa1Li\xf1y\xe5b\x83\xd4\xca\x96\xbc\x9d\xcc\x10o@\xc3\\\xd0\x08R\xb2
\xa5]\xf0\'{\xc3S\x97\xe4\xee\xd1X\xfd\xa4\xc86#\x9b\x80\xb7\xaa'
```

Figure 5.4: Encrypted File



File Decryption key

hybridcrypt@gmail.com

to me

7:33 PM (0 minutes ago)

project_abstract_final file decryption key is EsvW1449EqNr3939

Figure 5.5: Receive Key on Registered E-mail ID

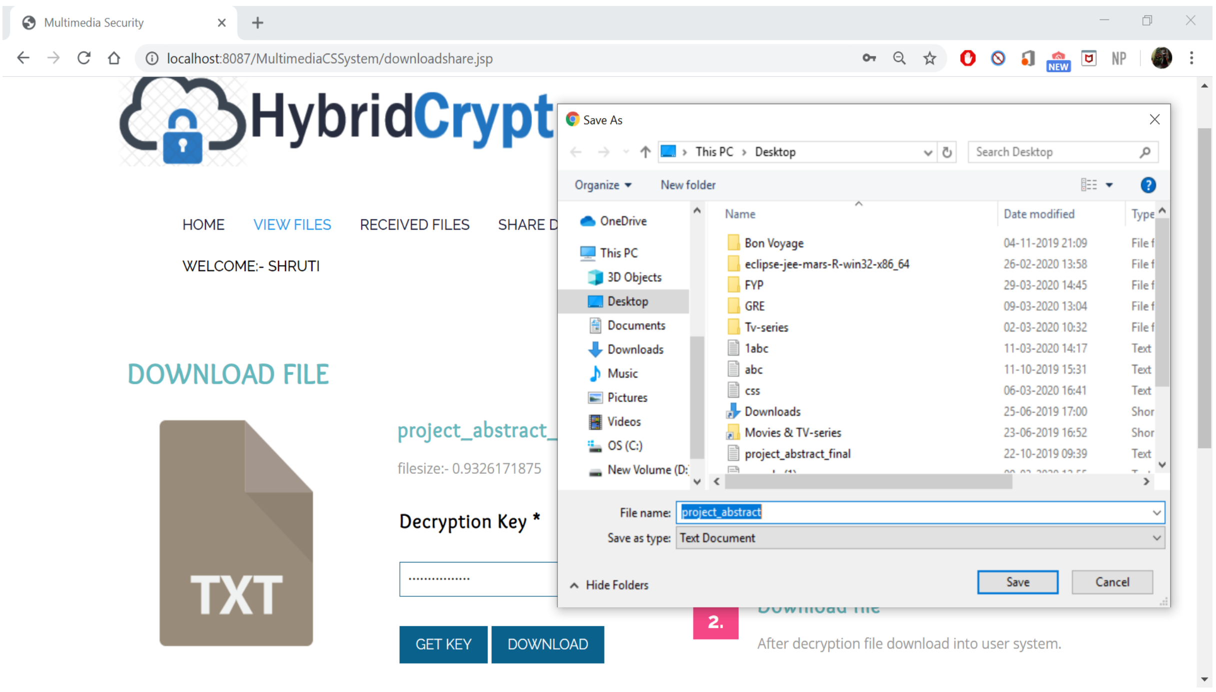


Figure 5.6: Download File

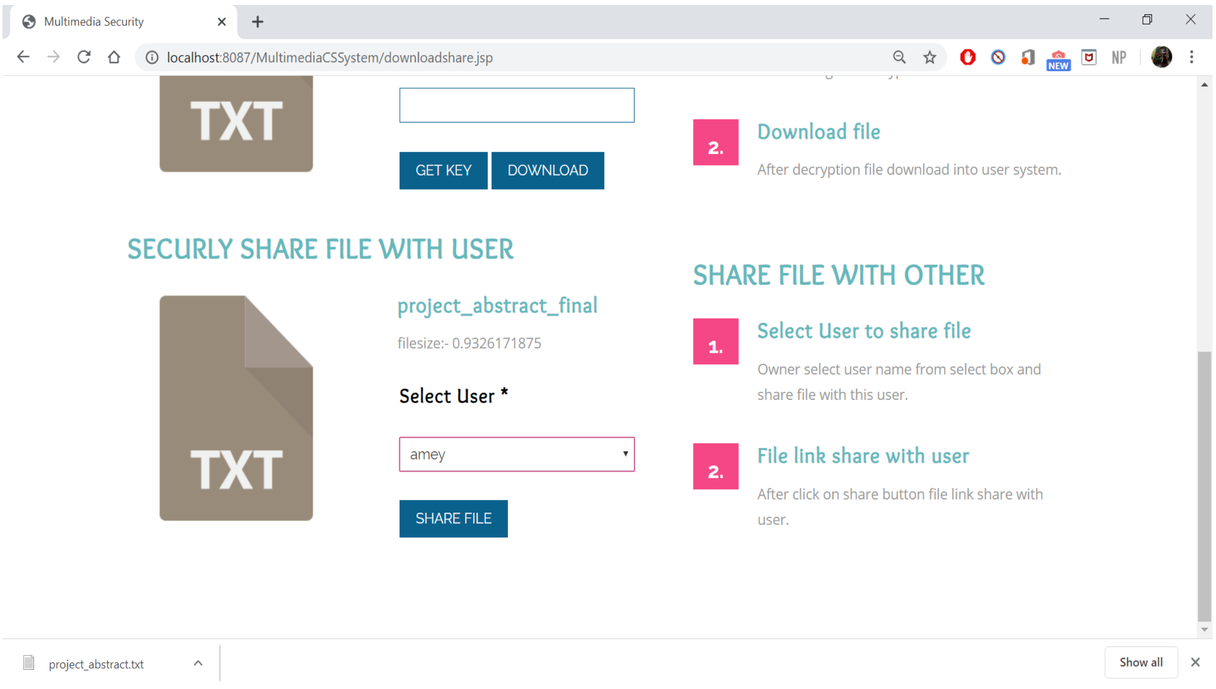
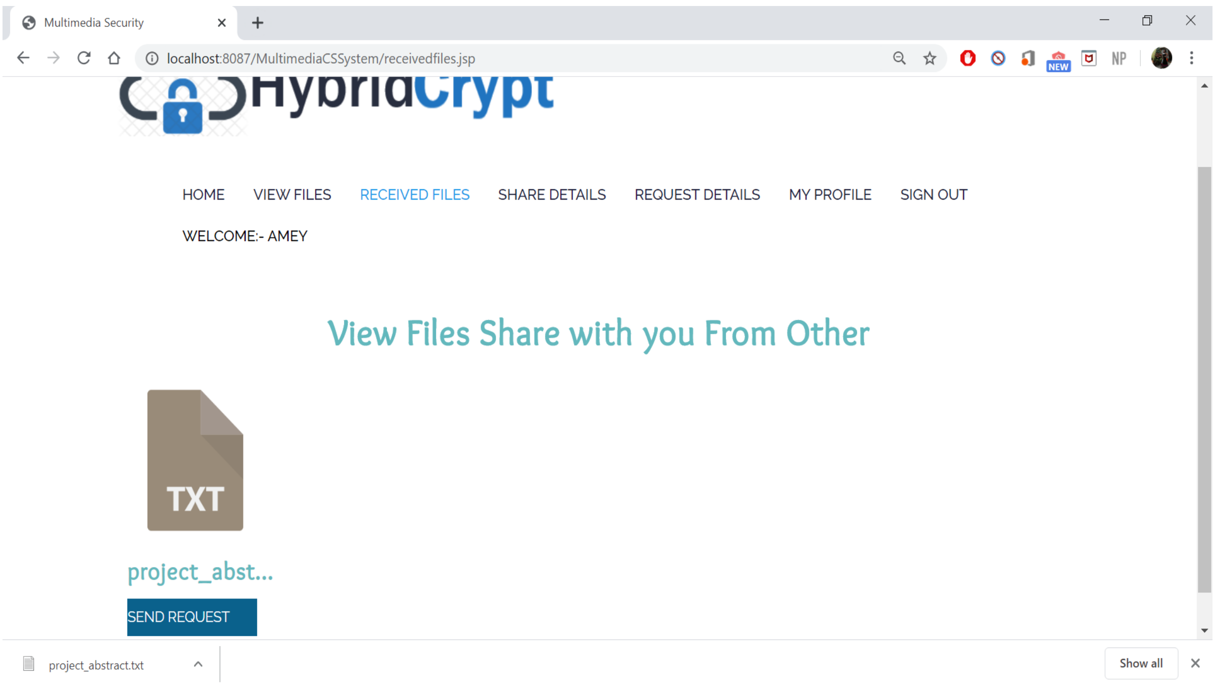

Multimedia Security
localhost:8087/MultimediaCSSystem/downloadshare.jsp
TXT
GET KEY
DOWNLOAD
2.
Download file
After decryption file download into user system.
SECURLY SHARE FILE WITH USER
project_abstract_final
filesize:- 0.9326171875
Select User *
amey
SHARE FILE
SHARE FILE WITH OTHER
1.
Select User to share file
Owner select user name from select box and share file with this user.
2.
File link share with user
After click on share button file link share with user.
project_abstract.txt
Show all

Figure 5.7: User1 Chooses User2 to Share File With

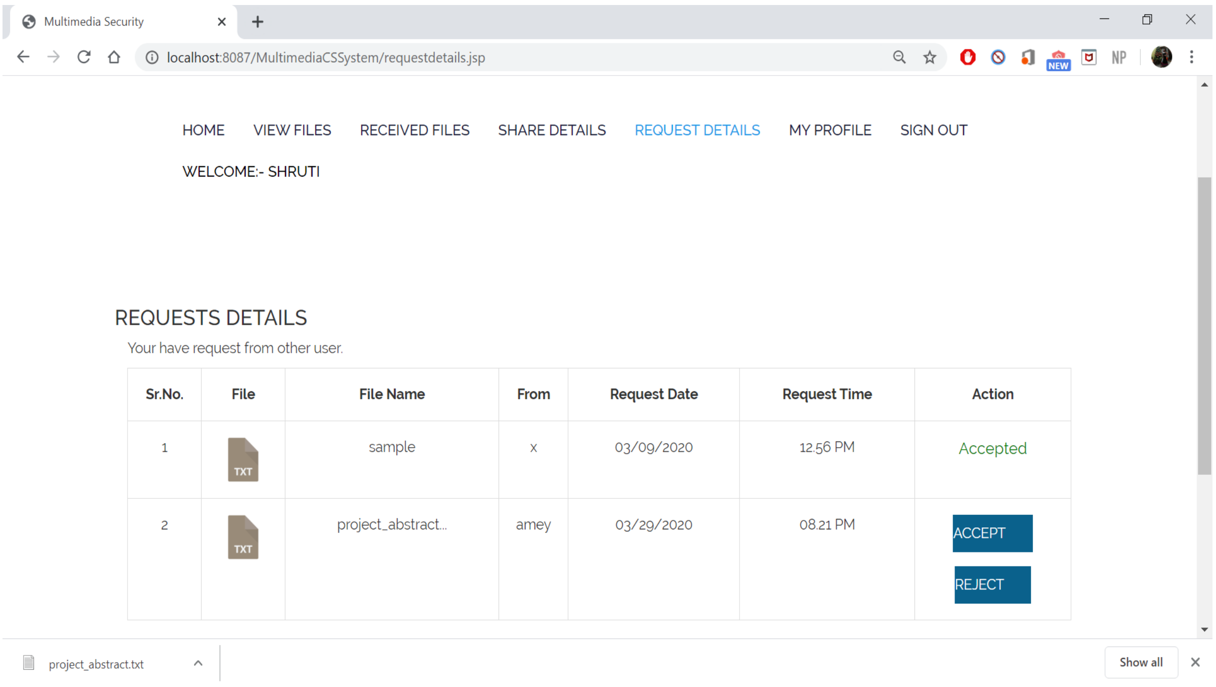


WELCOME:- AMEY

View Files Share with you From Other

project_abst...

Figure 5.8: User2 Receives Intimation and Sends Request to User1



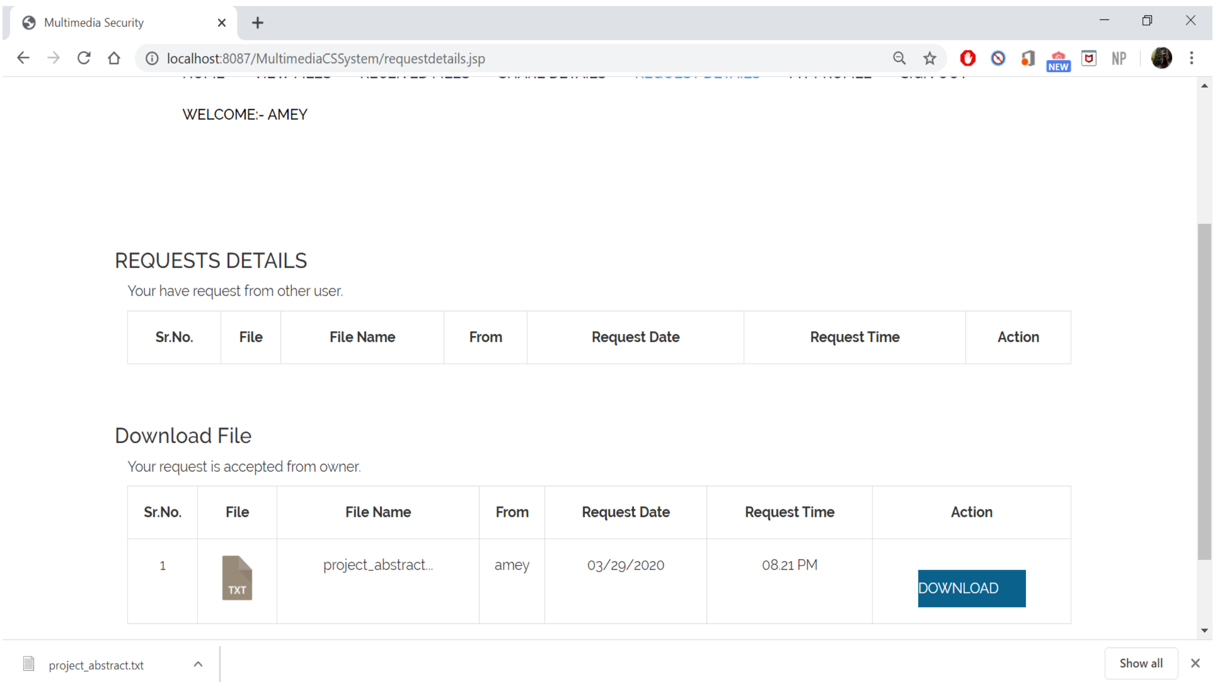
WELCOME:- SHRUTI

REQUESTS DETAILS

Your have request from other user.

Sr.No.	File	File Name	From	Request Date	Request Time	Action
1	TXT	sample	x	03/09/2020	12.56 PM	Accepted
2	TXT	project_abstract...	amey	03/29/2020	08.21 PM	ACCEPT REJECT

Figure 5.9: User1 Accepts Request



WELCOME:- AMEY

REQUESTS DETAILS

Your have request from other user.

Sr.No.	File	File Name	From	Request Date	Request Time	Action

Download File

Your request is accepted from owner.

Sr.No.	File	File Name	From	Request Date	Request Time	Action
1	TXT	project_abstract...	amey	03/29/2020	08:21 PM	DOWNLOAD

Figure 5.10: User2 Can Now Download the Shared File

5.2 Sample Code

- hash.java:

```
package uploader;

import java.io.*;
import java.net.*;
import java.util.*;
import java.security.*;

public class hash {

    public void Hash() {
        // Construct.
    }

    public static void main(String args[]) {
        String filehash=getHash("sumit");
        System.out.println(filehash);
    }

    public static byte[] createFileHash(String filename, String method) {
        try {
            try {
                InputStream fis = new FileInputStream(filename);

                byte[] buffer = new byte[1024];
                MessageDigest complete = MessageDigest.getInstance(method )
                   red↪ ;
                int numRead = 0;
                while (numRead != -1) {
                   numRead = fis.read(buffer);
                   if (numRead > 0) {
                      complete.update(buffer, 0, numRead);
                   }
                }
                fis.close();
                return complete.digest();
            }
            catch(NoSuchAlgorithmException nsae) {
                return null;
            }
            // Do nothing for it.
        }
        catch(IOException e ) {
            return null;
        }
        // Do nothing for it.

    }

    public static byte[] createHash(String text, String method) {
        try {
            byte[] b = text.getBytes();
            MessageDigest algorithm = MessageDigest.getInstance(method );
            algorithm.reset();
            algorithm.update(b);
            byte messageDigest[] = algorithm.digest();
            return messageDigest;
        }
        catch(NoSuchAlgorithmException nsae) {
            return null;
        }
        // Do nothing for it.

    }

    public static String getFileHash(String filename) {
        try {
            byte[] b = createFileHash(filename, "SHA-1");
            return asHex(b);
        }
        catch(Exception e) {
            return null;
            //Don't do anything else.
        }
    }

    public static String getHash(String text) {
        try {
            byte[] b = createHash(text, "SHA-1");
            return asHex(b);
        }
        catch(Exception e) {
            return null;
            //Don't do anything else.
        }
    }

    public static String asHex(byte[] b) {
        String result = "";
        for (int i=0; i < b.length; i++) {
            result +=
            Integer.toString(( b[i] & 0xff ) + 0x100, 16).substring(1 );
        }
        return result;
    }
}
    }
```

- extnselector.java:

```
package uploader;

import java.io.File;
import java.io.IOException;

import org.apache.commons.io.FilenameUtils;

public class extnslector {



        public String extnslector(String extension) {

        String exten = null;

        String files;
        File folder = new File(path.path+"//PNG");
        File[] listOfFiles = folder.listFiles();

        for (int i = 0; i < listOfFiles.length; i++) {
        if (listOfFiles[i].isFile()) {
        files = listOfFiles[i].getName();
        if (files.endsWith(".png") || files.endsWith(".PNG")) {
        String fileNameWithOutExt = FilenameUtils.removeExtension(files);
        // System.out.println(fileNameWithOutExt);
        if (extension.toLowerCase().endsWith(fileNameWithOutExt.
            red↪ toLowerCase()))
        {
        exten = "PNG//" + fileNameWithOutExt + ".png";
        }
        }
    }
        }
        return exten;
        }

}
```

- EncryptFile.java:

```
package encryption;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;

public class EncryptFile {

KeyGenerator keyGenerator = null;
static SecretKeySpec secretKey = null;
static Cipher cipher = null;
String masterPassword = null;
public String instance = "RSA";

public EncryptFile(String masterPassword) {
this.masterPassword = masterPassword;
this.masterPassword = "My␣Key";
try
    {
    keyGenerator = KeyGenerator.getInstance(instance);
    secretKey = new SecretKeySpec(masterPassword.getBytes(), instance);
    cipher = Cipher.getInstance(instance);
    } catch (NoSuchPaddingException | NoSuchAlgorithmException ex) {
    System.out.println(ex);
        }
    }
    public static String encrypt(String srcPath ,String destpath) {
        File srcFile = new File(srcPath);
        File encryptedFile = new File(destpath);
// File encryptedFile = new File("./encryptfiles/" + rawFile.getName());

    InputStream inStream = null;
    OutputStream outStream = null;
    try
    {
    cipher.init(Cipher.ENCRYPT_MODE, secretKey);
    inStream = new FileInputStream(srcFile);
    outStream = new FileOutputStream(encryptedFile);
    byte[] buffer = new byte[1024];
    int len;
    while ((len = inStream.read(buffer)) > 0) {
        outStream.write(cipher.update(buffer, 0, len));
        outStream.flush();
        }
    BufferedReader br = null;
    outStream.write(cipher.doFinal());
    inStream.close();
    outStream.close();
    String print = "";
    br = new BufferedReader(new FileReader(encryptedFile));
    String str;
    while ((str = br.readLine()) != null) {
            print = print + str + "\n";
        }
    } catch (IllegalBlockSizeException | BadPaddingException |
        red↪ nvalidKeyException ex) {
        System.out.println(ex);
    } catch (FileNotFoundException ex) {
        System.out.println(ex);
    } catch (IOException ex) {
        System.out.println(ex);
    }
        return destpath;
    }

}
```

- fileencryption.java:

```
package encryption;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import javax.crypto.Cipher;
import javax.crypto.CipherOutputStream;
import javax.crypto.spec.SecretKeySpec;

public class file_encryption {
       public static void encryption(String source, String destination,
                      String MYkey) throws IOException {
       //AES
       String I = source;
       String O = destination;

       // FileRecordHandler handler = new FileRecordHandler();
       // handler.FileRecordHandler(user , filename, extension, MYkey ;
              try {

       FileInputStream file = new FileInputStream(I);
       FileOutputStream outfile = new FileOutputStream(O);
       // byte k[] = "CooL2116NiTh5252".getBytes();
       byte k[] = MYkey.getBytes();
       try
       {
       SecretKeySpec key = new SecretKeySpec(k, "AES");
       Cipher enc = Cipher.getInstance("AES");
       enc.init(Cipher.ENCRYPT_MODE, key);
       CipherOutputStream cos = new CipherOutputStream(outfile, enc);
       byte[] buf = new byte[1024];
       int read;
       while ((read = file.read(buf)) != -1) {
       cos.write(buf, 0, read);
       }
       file.close();
       outfile.flush();
       cos.close();
       System.out.print("file␣incrypted");
       outfile.close();

              } catch (Exception e) {
                     System.out.print(e);
              }
       } catch (FileNotFoundException e) {
              e.printStackTrace();
              System.out.print(e);
              }
       }

}
```

- filedownloader.java:

```
package downloader;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import uploader.path;

import Database.DatabaseConnection;

/**
 * Servlet implementation class filedownloder
 */
@WebServlet("/filedownloder")
public class filedownloder extends HttpServlet {
       private static final long serialVersionUID = 1L;

       /**
        * @see HttpServlet#HttpServlet()
        */
       public filedownloder() {
              super();
              // TODO Auto-generated constructor stub
       }

       /**
        * @see HttpServlet#doGet(HttpServletRequest request,
           red↪ HttpServletResponse
        * response)
        */
       protected void doGet(HttpServletRequest request,
                      HttpServletResponse response) throws
                         red↪ ServletException, IOException {

       }

       /**
        * @param path
        * @see HttpServlet#doPost(HttpServletRequest request,
           red↪ HttpServletResponse
        * response)
        */
       protected void doPost(HttpServletRequest request,
          red↪ HttpServletResponse response) throws ServletException,
          red↪ IOException {
              String backuppath=getServletContext().getRealPath("/backup"
                 red↪ );
```

```
PrintWriter out = response.getWriter();
HttpSession session = request.getSession();
String user = (String) session.getAttribute("username");
String file = request.getParameter("filename");
String key = request.getParameter("deckey");
String uploader = request.getParameter("uploder");

String filePath = path.path + uploader;

File pathToCreate = new File(filePath);
while(!(pathToCreate.exists())){
       pathToCreate.mkdir();
}

filePath = path.path + uploader + "//" + file;

DatabaseConnection db = new DatabaseConnection();
db.dbconnection();
String dcryptPath = preProcessing.preProcessing(uploader,
   red↪ key, file, db, user, backuppath);

try {
       if (!(dcryptPath.equals(""))) {
              response.setContentType("application/octet-
                 red↪ stream");
       response.setHeader("Content-Disposition",
              "attachment;filename="+file);

              FileInputStream fileInputStream = new
                 red↪ FileInputStream(dcryptPath);

              int i;
              while ((i=fileInputStream.read()) != -1) {
              out.write(i);
              }
              fileInputStream.close();
              out.close();

              db.closeConnection(db);

              response.setContentType("text/html;charset=
                 red↪ UTF-8");
              response.getWriter().write("success");
       } else {

              response.setContentType("text/html;charset=
                 red↪ UTF-8");
              response.getWriter().write("fail");
       }
} catch (Exception e) {
       // TODO Auto-generated catch block
       e.printStackTrace();
```

```
                }

        }

}
```

5.3 Testing

5.3.1 Unit Testing

Unit Testing: The goal of unit testing to separate each part of the program and test that the individual parts are working correctly and as intended.
Test Objectives: Authorizing the login credentials of user and checking file duplication detection

Table 5.1: Unit Testing

Test Condition	**Input Specification**	**Output Specification**	**Success/Fail**
User logging to system	Login ID and password	User is able to login and access the system	Success
Checking file duplication detection	Upload same file twice	Duplication detected	Success

5.3.2 Integration Testing

Integration Testing: Combine the unit tested module one by one and test the functionality of the combined unit.
Test Objectives: After logging in to the system user is able to upload files and share them

Table 5.2: Integration Testing

Test Condition	**Input Specification**	**Output Specification**	**Success/Fail**
Logging in and then uploading and sharing of files	User logs in with their user ID and password and uploads and shares file	File is uploaded and user is able to share the file	Success

5.3.3 Blackbox Testing

BlackBox Testing: In BlackBox Testing, we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.
Test Objectives: Encryption of uploaded file and download the decrypted file

Table 5.3: BlackBox Testing

Test Condition	**Input Specification**	**Output Specification**	**Success/Fail**
Encryption of uploaded file and download the decrypted file.	Upload of file and decryption key	Uploaded file is encrypted and decrypted file is downloaded on entering key	Success

Chapter 6

Conclusion & Future Scope

This project implements a double stage encryption algorithm that provides the security of multimedia contents in the cloud. The proposed algorithm is crucial in the second stage. The randomly generated key provides more security than the conventional encryption system. The ciphertext is stored in the cloud instead of original multimedia content. The cipher text is undoubtedly hard to recover the original content for random asymmetric key. Wide application of the proposed algorithm protects the information from the side channel attacker to grab the multimedia data from the cloud. Thus, the multimedia content is safe in the cloud.

In the future, a possible improvement is to let the owner issue a valid user a special seed value which the user can then use to generate keys on their own. Our solution can be extended to allow users to perform data updates in addition to data owners. Additionally, it would be interesting to provide security of the adaptive HTTP video in mobile cloud computing.

References

[1] Aruna Verma Punam V. Maitri. Secure file storage in cloud computing using hybrid cryptography algorithm. *IEEE WiSPNET*, 2016.

[2] Byung Gook Lee Anu Kumari Arjun Kumar, Hoonjae Lee. Secure storage and access of data in cloud computing. *ICT Convergence (ICTC)*, 2012.

[3] Amit Chugh Sunita Sharma. Survey paper on cloud storage security. *International Journal of Innovative Research in Computer and Communication Engineering*, 2013.

[4] Su Naizheng Zhang Yuhan Diao Zhe, Wang Qinghong. Secure file storage in cloud computing using hybrid cryptography algorithm. *International Journal of Innovative Research in Computer and Communication Engineering*, 2013.

[5] S. Sree Vivek Bharat S. Rawal. Secure cloud storage and file sharing. *Big Data Security on Cloud (BigDataSecurity)*, 2017.

[6] Dr. Bhadresh P. Patel Vishal R. Pancholi. Enhancement of cloud computing security with secure data storage using aes. *International Journal for Innovative Research in Science Technology*, 2016.

[7] Ya Pan Shuying Li. Secure file storage in cloud computing using hybrid cryptography algorithm. *BTAIJ*, 2014.

[8] Min-Shiang Hwang. Wei-Fu Hsien, Chou-Chen Yang. A survey of public auditing for secure data storage in cloud computing. *International Journal of Network Security*, 2016.

[9] Gail-Joon Ahn. Hassan Takabi, James B.D. Joshi. Security and privacy challenges in cloud computing environments . *Security and Privacy Challenges in Cloud Computing Environments*, 2010.

[10] A. EI Abbadi. D. Agrawal, S. Das. Big data and cloud computing: current state and future opportunities. *14th International Conference on Extending Database Technology, pp. 530-533, Uppsala, Sweden*, 2011.

[11] D. Lekkas. D. Zissis. Addressing cloud computing security issues. *Future Generation computer systems, vol. 28*, 2012.

[12] Z. Cao X. Dong W. Jia Y. Chen L. Wei, H. Zhu and A. V . Vasilakos. Security and privacy for storage and computation in cloud computing. *Information Sciences, vol. 258*, 2014.

[13] P. Afsar. G. Thomas, V. Jose.

[14] S. T. Nguyen C. Rong and M. G. Jaatun. Beyond lightning: A survey on security challenges in cloud computing. *Computers Electrical Engineering, vol. 39*, 2013.

[15] L. M. Kaufman. Data security in the world of cloud computing. *IEEE Security Privacy, vol. 7*, 2009.

[16] L. B. Othmane L. Lilien A. Kim M. Kang R. Ranchal, B. Bhargava and M. Linderman. Protection of identity information in cloud computing without trusted third party. *29th IEEE Symposium on Reliable Distributed Systems*, 2010.

[17] M. D. Ryan. Cloud computing security: The scientific challenge, and a survey of solutions. *International Journal of Network Security*, 2016.

[18] R. Gellman. Privacy in the clouds: Risks to privacy and confidentiality from cloud computing. *Journal of Systems and Software, vol. 86*, 2013.

Acknowledgement

Success of a project like this involving high technical expertise, patience and massive support of guides, is possible when team members work together. We take this opportunity to express our gratitude to those who have been instrumental in the successful completion of this project. We would like to show our appreciation to **Mrs. Rakhi Kalantri** for her tremendous support and help, without her this project would have reached nowhere. We would also like to thank her for being our project coordinator as well and for providing us with regular inputs about documentation and project timeline. A big thanks to our HOD **Dr. Lata Ragha** for all the encouragement given to our team. We would also like to thank our principal, **Dr. S. M. Khot**, and our college, **Fr. C. Rodrigues Institute of Technology, Vashi**, for giving us the opportunity and the environment to learn and grow.

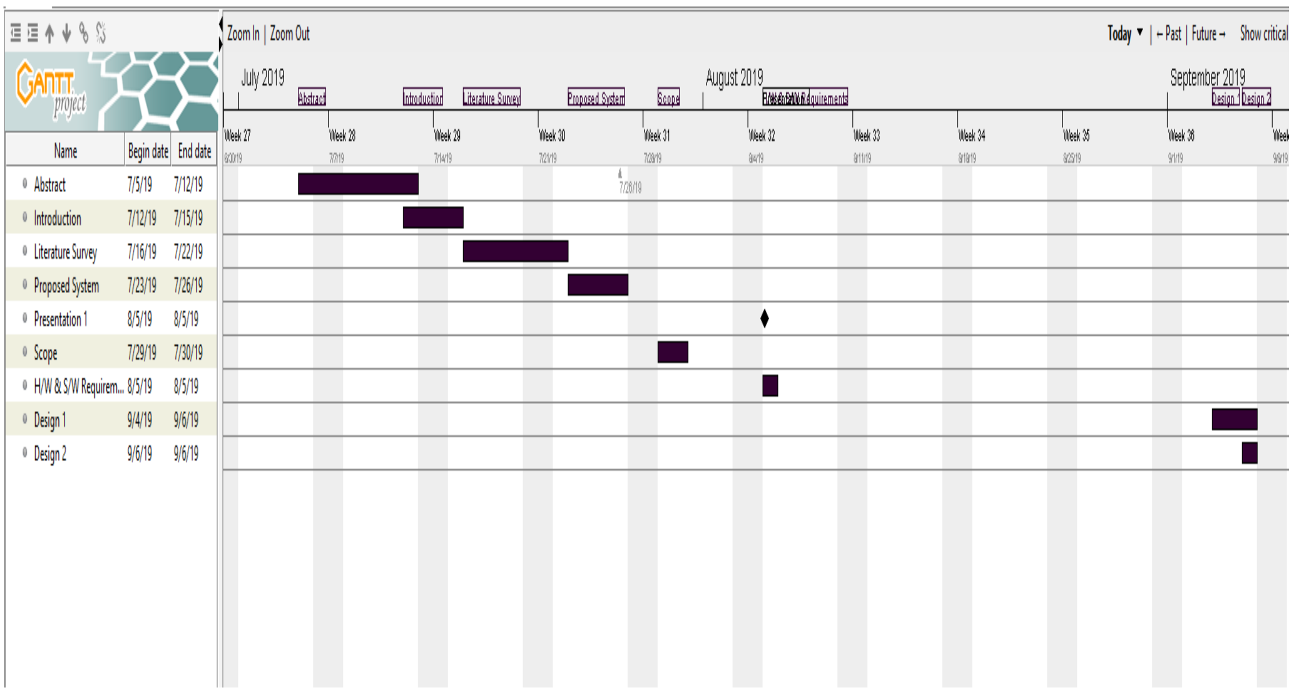
Project Group Members:

1. Shruti Kanatt, 101622

2. Amey Jadhav, 101665

3. Prachi Talwar, 101676

Appendix A : Timeline Chart



Name	Begin date	End date
Abstract	7/5/19	7/12/19
Introduction	7/12/19	7/15/19
Literature Survey	7/16/19	7/22/19
Proposed System	7/23/19	7/26/19
Presentation 1	8/5/19	8/5/19
Scope	7/29/19	7/30/19
H/W & S/W Requirem...	8/5/19	8/5/19
Design 1	9/4/19	9/6/19
Design 2	9/6/19	9/6/19

Figure 6.1: Timeline Chart: Part 1

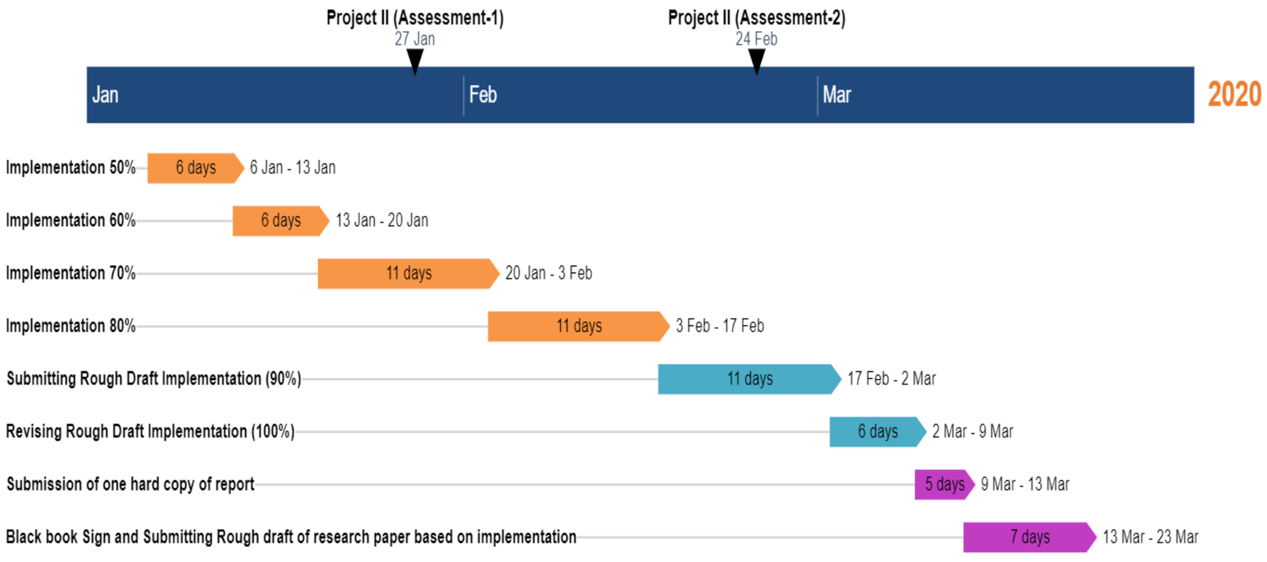


Figure 6.2: Timeline Chart: Part 2

Appendix B : Publication Details

Title: Review of Secure File Storage on Cloud using Hybrid Cryptography

Paper ID: IJERTV9IS020014

Journal: International Journal of Engineering Research Technology (IJERT)

Volume: 9

Issue: 2 (February 2020)

Link: https://www.ijert.org/volume-09-issue-02-february-2020