AN ADVANCED CRYPTOGRAPHY ALGORITHM FOR SECURE FILE ACCESS IN CLOUD

By

L.Vamsi krishna

Project Synopsis

Industry:- Ed-Tech

Primary Azure Technology:- App Service,windows virtual desktop

Other Azure Technologies:- Azure Data Centers

ABSTRACT

Cloud is used in various fields like industry, military, college, etc. for various services and storage of huge amount of data. Data stored in this cloud can be accessed or retrieved on the users request without direct access to the server computer. But the major concern regarding storage of data online that is on the cloud is the Security. This Security concern can be solved using various ways, the most commonly used techniques are cryptography and steganography. But sometimes a single technique or algorithm alone cannot provide high level security. So we have introduces a new security mechanism that uses a combination of multiple cryptographic algorithms of symmetric key. In this proposed system AES (Advanced Encryption Standard) algorithm are used to provide security to data. All the algorithms use 128-bit keys. The technique is used to securely store the key information. Key information will contain the information regarding the encrypted part of the file, the algorithm and the key for the algorithm file during encryption is split into three parts.

Keywords:

Cryptography, Steganography, Security, Cloud Computing.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Introduction

The data storage security in the cloud has been received widespread attention. People on the one hand, hope to be able to use large cloud storage service to alleviate the pressure of the local storage, on the other hand, worry that cloud service providers many confidential information to a third party without authorization, resulting in data information leakage. Data saved in the cloud are required to be encrypted to guarantee a certain security. Cloud computing is generally thought to include the following several levels of service: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). IaaS provides IT Infrastructures as a Service over the Internet. PaaS provides computing platform as a service to support the cloud applications. SaaS allows users to use cloud application without installing and running software on their own computers. Data owners have limited control over IT infrastructures.

This also means that effective data may be revealed and that the deleted data may still be recoverable. However, the file system is responsible for organizing logical orders of data which is stored among the available blocks on the physical medium, and the file system layer which can call for device driver interface allows files to be read, written and created and deleted. These features led people to consider using it to implement secure storage of data filesystem.

1.2 Aim of the project

To secure file storage in cloud computing using cryptography algorithms. Colossus ensures security of the user's data stored on cloud (AWS) by providing a tool that helps to encrypt files using two Algorithm. The user receives the key via email. which will provide Encryption, Storing and Decryption of files then store the encrypted file in the cloud server where the user can access it using the key which is sent to the registered user's email address.

1.3 Project Domain

The goal of the project is to secure the file in cloud using AES algorithm. For data security and privacy protection issue, Cryptography, Steganography methods are used. It provides encryption, storing and decryption of files process takes place for securing the file in cloud.

1.4 Scope of the Project

The main goal is to securely store and access data in cloud that is controlled by the owner of the data. We exploit the technique of cryptography encryption to protect data files in the cloud. Two part of the cloud server improved the performance during storage and accessing of data.

1.5 Methodology

The existing cryptography algorithm in cloud is usually stored their data and this data can be ac cessed anywhere any time, but the data security is the main concern for the organizations in the cloud storage model. In order to offer secure data transmission and communication over the heterogeneous and connected network, encryption models plays an important role. Encryption algorithm used the key for the secure data communication. A key is first agreed upon by the communicating parties and kept secret. Now, the key and the encryption algorithms are utilized for encrypting the message previous to sending it from one party to another. This text obtained, known as cipher text, is received by the other party who then decrypt it taking use of the same key and the decryption algorithm. Here, the key is kept as secret while the encryption and decryption algorithms are the known elements. As we now know, the key is known to both the sender and the receiver in this cryptography method, but this key distribution is of great importance and proves to be a very difficult task. The symmetric key encryption method is used, where only a single key used for the encryption and decryption at the sender and receiver side.

Hybrid encryption is a mode of encryption that merges two or more encryption systems. It incorporates a combination of asymmetric and symmetric encryption to benefit from the strengths of each form of encryption. These strengths are respectively defined as speed and security. Hybrid encryption is considered a highly secure type of encryption as long as the public and private keys are fully secure.A hybrid encryption scheme is one that blends the convenience of an asymmetric encryption scheme with the effectiveness of a symmetric encryption scheme. Hybrid encryption is achieved through data transfer using unique session keys along with symmetrical encryption. Public key encryption is implemented for random symmetric key encryption. The recipient then uses the

public key encryption method to decrypt the symmetric key. Once the symmetric key is recovered, it is then used to decrypt the message.

The steps of hybrid encryption are:

Generate a symmetric key.

The symmetric key needs to be kept a secret.

Encrypt the data using the secret symmetric key.

The person to whom we wish to send a message will share his public key,keep the private key a secret.

Encrypt the symmetric key using the public key of the receiver.

Send the encrypted symmetric key to the receiver.

Send the encrypted message text.

The receiver decrypts the encrypted symmetric key using her private key and gets the symmetric key needed for decryption.

The receiver uses the decrypted symmetric key to decrypt the message, getting the original message.

Chapter 2

LITERATURE REVIEW

In this chapter the literature review is a search and evaluation of the available literature in your given subject or chosen topic area. It documents the state of the art with respect to the subject or topic you are writing about.It synthesises the information in that literature into a summary

2.1 Cryptography

Cryptographic algorithm is a mathematical function used in the encryption and decryption process. A cryptographic algorithm works in combination with a key a word, number, or phrase to encrypt the plaintext. The same plaintext encrypts to different ciphertext with different keys.

Punam V.Maitri et all [1], To storing huge amount of data. We can retrieve data from cloud on request of user. To store data on cloud we have to face many issues. To provide the solution to these issues there are n number of ways. Cryptography and steganography techniques are more popular now a day's for data security. Use of a single algorithm is not effective for high level security to data in cloud computing

2.2 Storing Data in AES Algorithm

Encryption is one of the most common ways to protect sensitive data. Encryption works by taking plain text and converting it into cipher text, which is made up of some random characters. Only those who have the special key can decrypt it. AES uses symmetric key encryption, which involves the use of only one secret key to cipher and decipher information.

V.S. Mahalle et all [2], Sharing data in secure manner while preserving data from an untrusted cloud is still a challenging issue. Our approach ensures the security and privacy of client sensitive information by storing data across single cloud, using AES algorithm.

2.3 Secure file access in cloud

Cloud file storage is a method for storing data in the cloud that provides servers and applications access to data through shared file systems. This compatibility makes cloud file storage ideal for workloads that rely on shared file systems and provides simple integration without code changes. J. Reardon[3], Traditional access control is based on the plaintext, which cannot be directly applied in the cloud environment. There needs to realize that an authorized person can access the files (except for deleted files) which he has the right and that an unauthorized person cannot access any of the files at any time. In the early studies of the time-based secure deletion, the time property is used in the access control.

2.4 Secure file access in cloud using hybrid cryptography algorithm

The security of network and the network data is primary aspect of the network providers and service providers Combination of Symmetric key encryption and asymmetric key encryption.

Rawal, B. S. et all [6],The system requires a file as input which is encrypted using cryptography techniques and then stored at a remote location. Shared file can be viewed by the user by downloading the encrypted file from remote locations and decrypting using decryption algorithm on users machine using the metadata information shared with user by the owner.There are two kinds of users of the system.

Chapter 3

PROJECT DESCRIPTION

3.1 Existing System

In existing system single algorithm is used for data encode and decode purpose. But use of single algorithm is not accomplish high level security. If we use single symmetric key cryptography algorithm than we have to face security problem because in this type of algorithm applies a single key for data encode and decode. So key transmission problem occur while sharing key into multi user environment. Public key cryptography algorithms accomplish high security but maximum delay is needed for data encode and decode.

3.2 Proposed System

I propose to develop a website which will provide Encryption, Storing and Decryption of files. Encryption is done using three algorithms. First we encrypt the file using AES-256. We then store the encrypted file in the cloud server where the user can access it using the key which is sent to the registered user's email address. We can store all types of files such as Image, pdf, docx, audio, video, excel sheet. Other combination of algorithms may not encrypt all types of files such as audio and video which has continuous bits of data which may result in loss of data after encryption,but the proposed system is robust enough to encrypt all types of files without any loss of data which makes it useful for real time purposes. AES-256 is the most robust security protocol, it uses higher length key size that is 256 bit. It is useful in encrypting audio and video files.

3.3 Feasibility Study

A brief explanation of the feasibility study a algorithm and the technical implementation. The various economic and technical help in understanding the operational aspects of the application and give a clear understanding of how the operations built in the application.

3.3.1 Economic Feasibility

The definite financial advantages to the association that the proposed framework will give. It incorporates evaluation and ID of every one of advantages anticipated. This assessment regularly includes a cost and advantages investigation.

3.3.2 Technical Feasibility

asy manner. It is an assessment of the equipment and programming and how it addresses the issue of the draft. The present specialized assets of the association and their relevance to the reasonable needs of the proposed framework.

3.4 System Specification

3.4.1 Hardware Specification

- Hard ware: Pentium
- RAM : 1GB
- Hard Disc :20GB
- Flooy Drive:1.44 MB
- Key Board:standard Windows Keyboard
- Monitor :SVGA

3.4.2 Software Specification

- Operating System : Windows
- Coding Language : Python - version(3.5)
- Web Technologies: Html,CSS
- IDE:Tomcat
- Database :My SQL

Chapter 4

MODULE DESCRIPTION

4.1 General Architecture

The user signs in if already registered, or signs up to register themselves by providing their details such as name, email id, phone number, password for account etc. The user then selects the file that is to be uploaded by browsing from local storage. The user then selects the encryption algorithm that they want to use.

In this Figure shows the proposed system provides the choice between using a combination of AES. The selected file gets uploaded after getting encrypted using the selected encryption algorithm combination. The user also has the option of viewing the files that they have uploaded or have access to and downloading them. On selecting a file to download it, the user is sent the decryption key on their email id that was entered on registration or sign-up. Using this key, the user can download the decrypted or original file. The system also provides a comparison with respect to security between the two hybrid encryption algorithm combinations.

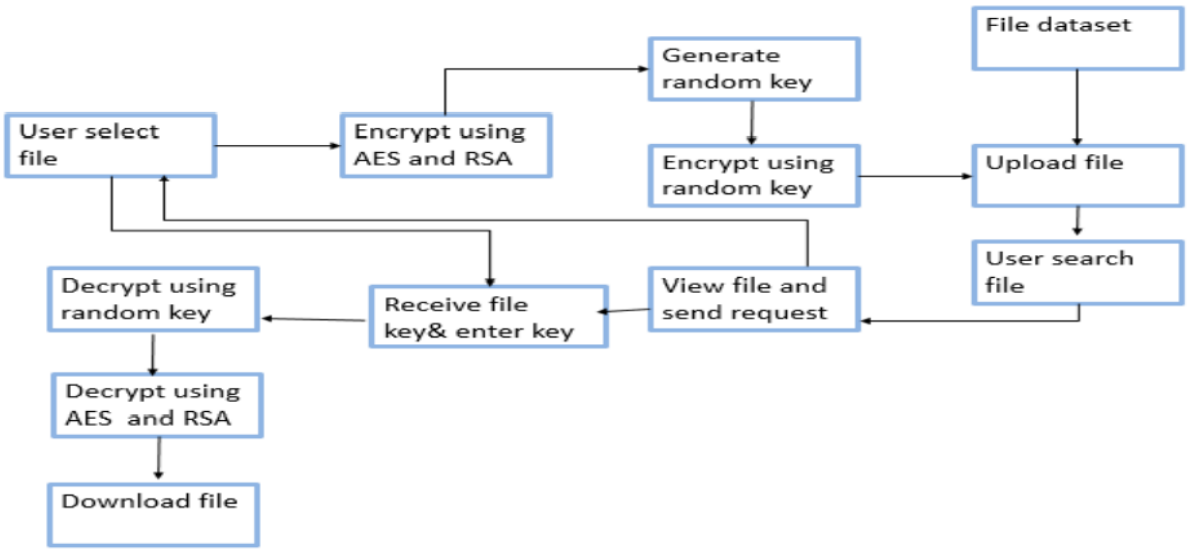

Figure 4.1: **Architecture Diagram**

4.2 Design Phase

The design phase involves an assessment of an organisation's existing network, data centre and application environment and produces a series of detailed design documents that articulate network, application migration and implementation processes.

4.2.1 Data Flow Diagram

In this Figure describes the way information flows through a process or system. It includes data inputs and outputs, data stores, and the various sub processes the data moves through.You can use these diagrams to map out an existing system and make it better or to plan out a new system for implementation.

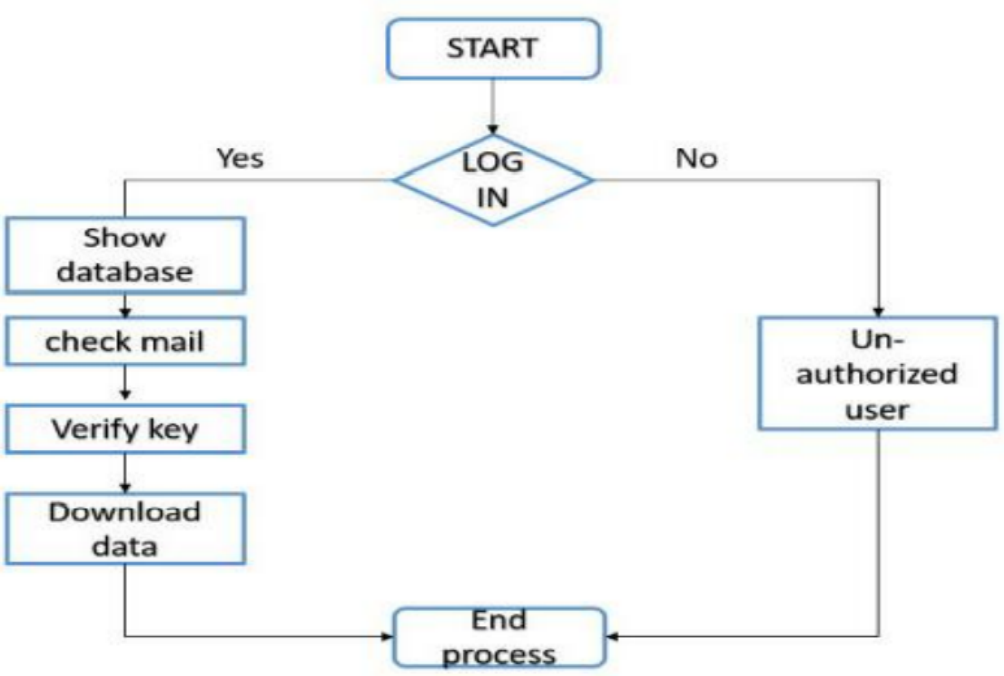

Figure 4.2: **Data Flow Diagram**

After the first stage of encryption algorithm triggered to bind the encrypted data with the user identity resulting in re-encrypting the data and is stored as is in the cloud cache database. For an unauthenticated user, the application flows through the 'no' flow.

4.2.2 Use case Diagram

It describes the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its functions

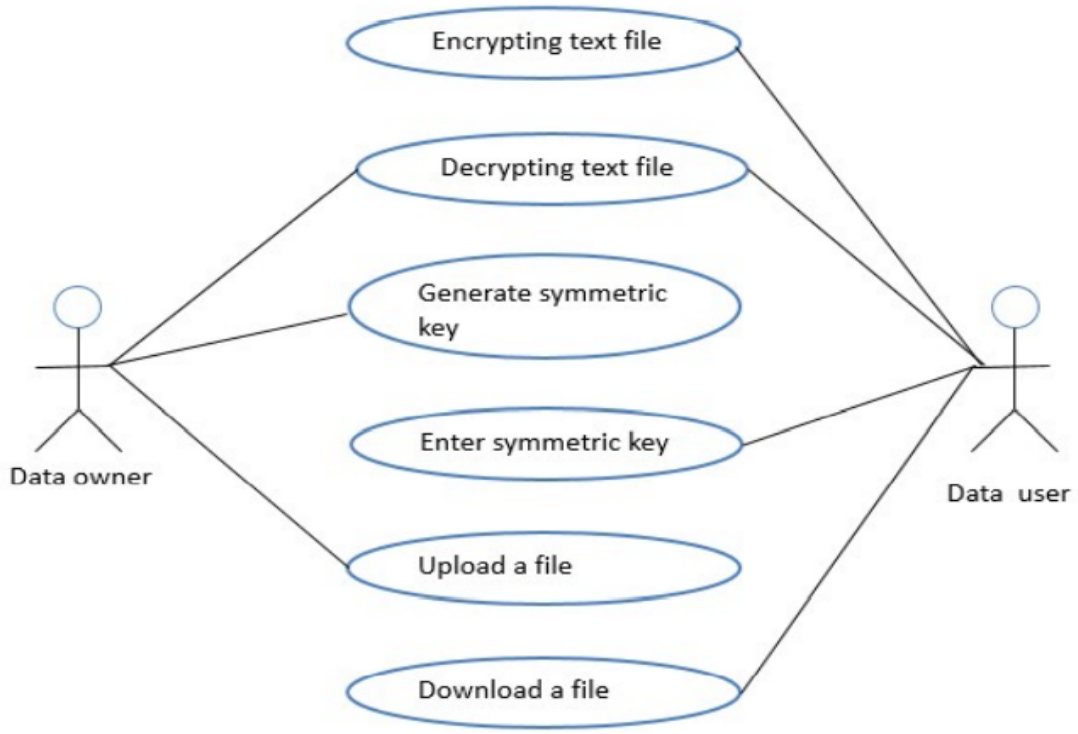

Figure 4.3: **Use Case Diagram**

It shows the purpose of use case is to present overview of the functionality provided by the system in terms of actors, their goals and any dependencies between those use cases. User register login file upload to cloud server key response to user cloud servers.

Chapter 5

IMPLEMENTATION AND TESTING

This chapter is to give an idea of the operations performed by the user in the applications and the various pages of the application that user traverses to perform operations. Depending on the user intent in the application depending on his privileges of activities.

5.1 Input and Output

Here input is given in the form of data and we can get output in the form of encrypted data. Input is given in the visual studio code.

5.1.1 Input Design

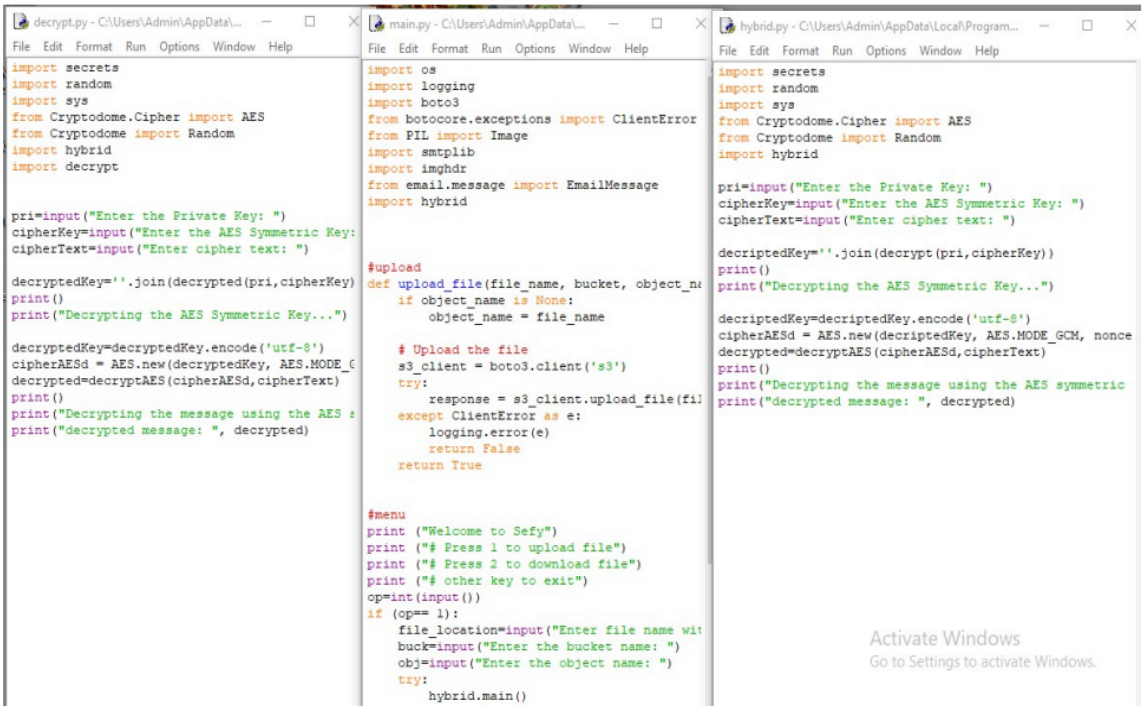

Figure5.1.1:Input Design

5.1.2 Output Design

It shows involves the data is encrypted and decrypted a message using AES and RSA. After entering a message the data is converted into keys. The key is private which is organized only by the user.

Figure5.1.1:Output Design

5.2 Types of Testing

5.2.1 Performance Testing

It failure due to one user action on the cloud should not affect other users performance Manual or automatic scaling should not cause any disruption On all types of devices, the performance of the application should remain the same overbooking at supplier end should not hamper the application performance.

5.2.2 Functional Testing

The valid input should give the expected results Service should integrate properly with other applications A system should display customer account type when successfully login to the cloud When a customer chose to switch to other services the running service should close automatically.

5.3 Security Testing

An only authorized customer should get access to data Data must be encrypted well Data must be deleted completely if it is not in use by a client Data should be accessible with insufficient encryption Administration on suppliers end should not access the customers' data check for various security settings like firewall, Anti-virus etc.

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Efficiency of the Proposed System

The file or downloaded for the further usage encryption is using data uploaded. Every data stored is encrypted data. The concept using encrypted data convert into binary value fully secure for the database.The File Storage module holds the file stored for usage by the data consumer and the files can be viewed and downloaded based on periodic time keys.

6.2 Comparison of Existing and Proposed System

The stored file is completely secured, as the file is being encrypted by using symmetric key cryptography and stenography techniques. The system is very secure and robust. Data of the users is secured on a cloud server which helps in avoiding unauthorized access from the outside world. Data security is a major priority. This system can be implemented in the banking and corporate sectors to securely transfer confidential data.

6.3 MainCode

```
import os
import logging
import boto3
from botocore.exceptions import ClientError
from PIL import Image
import smtplib
import imghdr
from email.message import EmailMessage
import hybrid



#upload
def upload_file(file_name, bucket, object_name=None):
    if object_name is None:
        object_name = file_name

    # Upload the file
    s3_client = boto3.client('s3')
    try:
        response = s3_client.upload_file(file_name, bucket, object_name)
    except ClientError as e:
        logging.error(e)
        return False
    return True


#menu
print ("Welcome to Sefy")
print ("# Press 1 to upload file")
print ("# Press 2 to download file")
print ("# other key to exit")
op=int(input())
if (op== 1):
    file_location=input("Enter file name with path: (with \\) ")
    buck=input("Enter the bucket name: ")
    obj=input("Enter the object name: ")
    try:
        hybrid.main()
        upload_file(file_location, buck, obj)
        print("DONE!")
    except:
        print ("Something went wrong!")
elif (op==2):
    buck1= input("Enter bucket name :")
    obj1= input("Enter Object name: ")
    file1= input("Enter File name: ")
    s3 = boto3.client('s3')
```

```
    s3.download_file(buck1, obj1, file1)
else:
    os._exit(0)
import secrets
import random
import sys
from Crypto.Cipher import AES
from Crypto import Random

import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders


def gcd(a, b):
        '''Euclid's algorithm '''
        while b != 0:
            temp=a % b
            a=b
            b=temp
        return a

def multiplicativeInverse(a, b):
        """Euclid's extended algorithm"""
        x = 0
        y = 1
        lx = 1
        ly = 0
        oa = a
        ob = b
        while b != 0:
            q = a // b
            (a, b) = (b, a % b)
            (x, lx) = ((lx - (q * x)), x)
            (y, ly) = ((ly - (q * y)), y)
        if lx < 0:
            lx += ob
        if ly < 0:
            ly += oa
        return lx

def generatePrime(keysize):
    while True:
        num = random.randrange(2*(keysize-1), 2*(keysize))
        if isPrime(num):
            return num

def isPrime(num):
```

```
99      if (num < 2):
100         return False
101     lowPrimes = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79,
            83, 89,
102                  97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179,
                         181, 191,
103                  193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277,
                         281, 283, 293,
104                  307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397,
                         401, 409, 419,
105                  421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509,
                         521, 523, 541,
106                  547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641,
                         643, 647, 653,
107                  659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761,
                         769, 773, 787,
108                  797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887,
                         907, 911, 919,
109                  929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997]
110
111     if num in lowPrimes:
112         return True
113
114     for prime in lowPrimes:
115         if (num % prime == 0):
116             return False
117
118     return millerRabin(num)
119
120
121 def millerRabin(n, k = 7):
122     if n < 6:
123         return [False, False, True, True, False, True][n]
124     elif n & 1 == 0:
125         return False
126     else:
127       s, d = 0, n - 1
128       while d & 1 == 0:
129          s, d = s + 1, d >> 1
130       for a in random.sample(range(2, min(n - 2, sys.maxsize)), min(n - 4, k)):
131          x = pow(a, d, n)
132          if x != 1 and x + 1 != n:
133             for r in range(1, s):
134                x = pow(x, 2, n)
135                if x == 1:
136                   return False
137                elif x == n - 1:
138                   a = 0
139                   break
140             if a:
```

```
                return False
        return True

def KeyGeneration(size=8):
    p=generatePrime(size)
    q=generatePrime(size)
    if not (isPrime(p) and isPrime(q)):
        raise ValueError('Both numbers must be prime.')
    elif p == q:
        raise ValueError('p and q cannot be equal')
    n = p * q
    phi = (p-1) * (q-1)
    e = random.randrange(1, phi)
    g = gcd(e, phi)
    while g != 1:
        e = random.randrange(1, phi)
        g = gcd(e, phi)
    d = multiplicativeInverse(e, phi)
    return ((n, e), (d, n))

def encrypt(pk, plaintext):
    n, e = pk
    c = [(ord(char) ** e) % n for char in plaintext]
    print(c)
    return c

def decrypt(pk, ciphertext):
    d, n = pk
    m = [chr((char ** d) % n) for char in ciphertext]
    return m


def encryptAES(cipherAESe,plainText):
    return cipherAESe.encrypt(plainText.encode("utf-8"))

def decryptAES(cipherAESd,cipherText):
    dec= cipherAESd.decrypt(cipherText).decode('utf-8')
    return dec




def main():
    print("**********************")
    print("**********************")
    print("Welcome...")
    print("We're going to encrypt and decrypt a message using AES and RSA")
    print("**********************")
    print("**********************")
```

```
191
192     #Obtains public key.
193     print("Genering RSA public and Privite keys......")
194     pub,pri=KeyGeneration()
195
196     #Generates a fresh symmetric key for the data encapsulation scheme.
197     print("Genering AES symmetric key......")
198     key = secrets.token_hex(16)
199     KeyAES=key.encode('utf-8')
200
201     #Encrypts the message under the data encapsulation scheme, using the symmetric key just
            generated.
202     plainText = input("Enter the message: ")
203     cipherAESe = AES.new(KeyAES,AES.MODE_GCM)
204     nonce = cipherAESe.nonce
205     print("Encrypting the message with AES......")
206     cipherText=encryptAES(cipherAESe,plainText)
207     f=open(r"<text-file>","w+b")
208     f.write(cipherText)
209     f.close()
210     print("Upload Done")
211     #Encrypt the symmetric key under the key encapsulation scheme, using Alices public key.
212     cipherKey=encrypt(pub,key)
213     print("Encrypting the AES symmetric key with RSA......")
214
215     mail_content = ("Hello, \nThis mail contains all those important details that you will need to
            access your file.. \nIn this mail we are sending decript.py through which you can decrypt
            the text file from AWS Cloud.\nThank You \n Private Key: " + str(pri) + "\n AES Symmetric
            Key: " + str(cipherKey))
216     sender_address = '<sender-emailID>'
217     sender_pass = '<password>'
218     receiver_address = '<receiver-emailID'
219     message = MIMEMultipart()
220     message['From'] = sender_address
221     message['To'] = receiver_address
222     message['Subject'] = 'Important Keys for Decryption'
223     message.attach(MIMEText(mail_content, 'plain'))
224     attach_file_name = (r'<file-name-with-location>')
225     attach_file = open(attach_file_name, 'rb') # Open the file as binary mode
226     payload = MIMEBase('application', 'octate-stream')
227     payload.set_payload((attach_file).read())
228     encoders.encode_base64(payload) #encode the attachment
229     #add payload header with filename
230     payload.add_header('Content-Decomposition', 'attachment', filename=attach_file_name)
231     message.attach(payload)
232     #Create SMTP session for sending the mail
233     session = smtplib.SMTP('smtp.gmail.com', 587) #use gmail with port
234     session.starttls() #enable security
235     session.login(sender_address, sender_pass) #login with mail_id and password
236     text = message.as_string()
```

```
237     session.sendmail(sender_address, receiver_address, text)
238     session.quit()
239     print('Mail Sent')
```

Output

Figure 6.1: **Encrypted data**

6.4 Decrypt code

```
import secrets
import random
import sys
from Crypto.Cipher import AES
from Crypto import Random
import hybrid

pri=input("Enter the Private Key: ")
cipherKey=input("Enter the AES Symmetric Key: ")
cipherText=input("Enter cipher text: ")

decriptedKey=''.join(decrypt(pri,cipherKey))
print()
print("Decrypting the AES Symmetric Key...")

decriptedKey=decriptedKey.encode('utf-8')
cipherAESd = AES.new(decriptedKey, AES.MODE_GCM, nonce=nonce)
decrypted=decryptAES(cipherAESd,cipherText)
print()
print("Decrypting the message using the AES symmetric key.....")
print("decrypted message: ", decrypted)
```

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

The main aim of this system is to securely store and retrieve data on the cloud that is only controlled by the owner of the data. Cloud storage issues of data security are solved using cryptography and steganography techniques. Data security is achieved using AES algorithm. Key information is safely stored. Less time is used for the encryption and decryption process. With the help of the proposed security mechanism, we have accomplished better data integrity, high security, low delay, authentication, and confidentiality. In the future we can add public key cryptography to avoid any attacks during the transmission of the data from the client to the server.

7.2 Future Enhancements

Cloud computing is powerful and expansive and will continue to grow in the future and provide many benefits. Cloud computing is extremely cost-effective and companies can use it for their growth. The future of cloud computing is bright and will provide benefits to both the host and the customer.

REFERENCES

[1] Punam V. Maitri, Arun Verma, "Secure File storage in Cloud Computing using Hybrid Cryptography Algorithm". IEEE-International Conference On Advances In Engineering, Science And Management.

[2] V.S. Mahalle, A. K. Shahade, "Enhancing the Data Security in Cloud by Implementing Hybrid (RSA AES)Encryption Algorithm", IEEE, INPAC, pp 146-149.

[3] J. Reardon, D. A. Basin, and S. Capkun. Sok: Secure data deletion. In IEEE Symposium on Security and Privacy, pp. 301-315.

[4] Vivek Kapoor, "Secure File Storage on Cloud Using Cryptography". International Research Journal of Engineering and Technology (IRJET).