



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

(SCHOOL OF COMPUTER SCIENCE AND ENGINEERING ,
SCOPE)

“SECURE FILE STORAGE ON CLOUD USING HYBRID CRYPTOGRAPHY”

PROJECT REPORT

Submitted for the course: **Operating Systems (CSE2005)**

By

RAJARSHI BHATTACHARYAY (19BCE0296)

RISHAV NASKAR (19BCE2324)

ABHIRAM SREEKANTHAM (19BCE2294)

BOLINENI GANESH SAI (19BCE0199)

YANDAPALLI SRI LAKSHMI JASWANTH (19BCE0096)

Slot: F1

Name of faculty: PROF. SANTHI H

ABSTRACT:

In this era cloud computing 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 and steganography. In this proposed system Fernet Encryption, Diffie Hellman and AES (Advanced Encryption Standard) algorithms are used to provide security to data. All the algorithms use 128-bit keys. LSB steganography 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. These individual parts of the file will be encrypted using different encryption algorithm

simultaneously with the help of multithreading technique. The key information is inserted into an image using the LSB technique. Our methodology guarantees better security and protection of customer data by storing encrypted data on a single cloud server, using AES, Fernet and Diffie Hellman algorithm.

INTRODUCTION :

Technological advancements are resulting in trends and movements that improve the quality of life. In this fast life where every person uses a smartphone and has access to the internet, the major concern that the people face is regarding the security of their information present online. This security concern is also about the file that is stored online on a cloud. This can be solved with the help of cryptography. Cryptography techniques convert original data into Cipher text. So only legitimate users with the right key can access data from the cloud storage server. The main aim of cryptography is to keep the security of the data from hackers, online/software crackers, and any third party users. Nonlegitimate user access to information results in loss of confidentiality. Security has the characteristics to block or stop this kind of unauthorized access or any other kind of malicious attacks on the data here by securing the users' trust. In the cloud computing environment, security is deemed to be a crucial aspect due to the significance of information stored on the cloud and the different services provided to the users. This data can be confidential and extremely sensitive. Hence, the data management and security should be completely reliable. It is necessary that the data in the cloud is protected from malicious attacks. So for the security of the data, we introduced a new mechanism in which we are using a combination of multiple symmetric key cryptography algorithm and steganography. In this proposed system Fernet Encryption, Advanced Encryption

Standard (AES) and Diffie Hellman algorithms are used to provide security to data. LSB algorithm is used for image steganography. Sensitive data of the user is hidden into a cover image for security purposes. AES, Fernet, and Diffie Hellman algorithms are combined to form a hybrid algorithm to accomplish better security. The steganography part assists in storing the key information safely. It makes it difficult for the attacker to recover the secret file of the user.

ABOUT HYBRID CRYPTOGRAPHY:

In cryptography, a **hybrid cryptosystem** is one which combines the convenience of a public-key cryptosystem with the efficiency of a symmetric-key cryptosystem. Public-key cryptosystems are convenient in that they do not require the sender and receiver to share a common secret in order to communicate securely (among other useful properties). However, they often rely on complicated mathematical computations and are thus generally much more inefficient than comparable symmetric-key cryptosystems. In many applications, the high cost of encrypting long messages in a public-key cryptosystem can be prohibitive. This is addressed by hybrid systems by using a combination of both.

A hybrid cryptosystem can be constructed using any two separate cryptosystems:

- a **key encapsulation** scheme, which is a public-key cryptosystem, and
- a **data encapsulation** scheme, which is a symmetric-key cryptosystem.

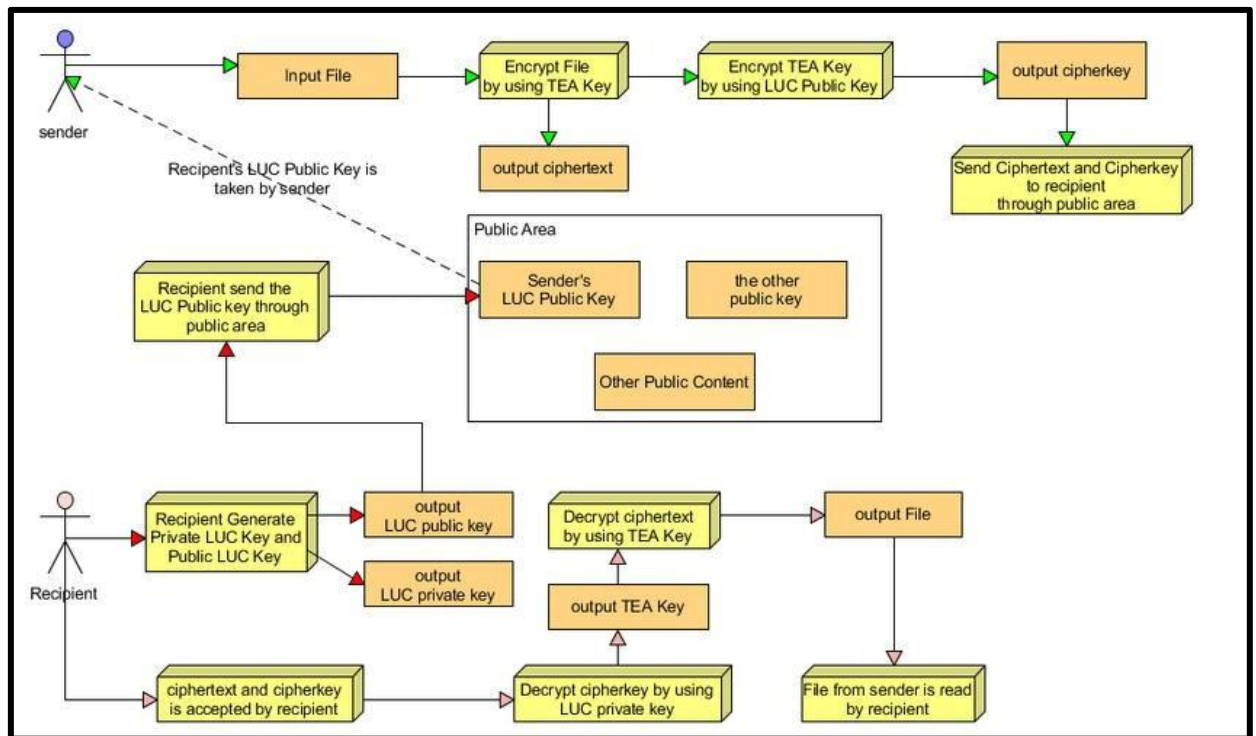
The hybrid cryptosystem is itself a public-key system, whose public and private keys are the same as in the key encapsulation scheme.

Note that for very long messages the bulk of the work in encryption/decryption is done by the more efficient symmetric-key scheme, while the inefficient public-key scheme is used only to encrypt/decrypt a short key value.

All practical implementations of public key cryptography today employ the use of a hybrid system. Examples include the TLS protocol which uses a public-key mechanism for key exchange (such as Diffie-Hellman) and a symmetric-key mechanism for data encapsulation (such as AES).

The OpenPGP (RFC 4880) file format and the PKCS #7 (RFC 2315) file format are other examples.

Sr. No.	Algorithm and Variant	SHA 2			
		SHA 0	SHA 1	SHA 2	
1	Output size	160 bits	256/224 bits	512/384 bits	
2	Internal state size	160 bits	256 bits	512 bits	
3	Block size	512 bits	512 bits	1024 bits	
4	Max message size	$2^{64} - 1$ bits	$2^{64} - 1$ bits	$2^{128} - 1$ bits	
5	Word size	32 bits	32 bits	64 bits	
6	Rounds	80	64	80	
7	Operations	AND, OR, XOR, shr, ROT, ADD (2^{32})	AND, OR, XOR, shr, ROT, ADD (2^{32})	AND, OR, XOR, shr, ROT, ADD (2^{64})	
8	Security bits	<34 (Collision found)	<63 (Collision found)	112 128	192 256 112 128



Literature Survey

RESEARCH PAPER 1:

https://www.researchgate.net/publication/334418542_A_Review_Paper_on_Cryptography

SUMMARY :

The basic concept of a cryptographic system is to cipher information or data in order to achieve confidentiality of the information in a way that an unauthorized person would be unable to derive its meaning. Two of the most common uses of cryptography would be using it to transmit data through an insecure channel, such as the internet, or ensuring that unauthorized people do not understand what they are looking at in a scenario in which they have accessed the information. They have also discussed about some historic algorithms like 1) Caesar Cipher 2) Simple Substitution Cipher and 3) Transposition Cipher And modern Algorithms like 1) Stream Ciphers and 2) Block Ciphers.

BASE PAPER :

https://www.academia.edu/41902889/Secure_File_Storage_Using_Hybrid_Cryptography

SUMMARY :

In Their proposed system there are four blocks each having different functionality.

1. The file is divided into chunks and then every chunk is encrypted using AES algorithm and digital signature for file is generated. A metadata file is created consisting of secret keys and information about file chunks.
2. On server files are stored and a table is maintained to map hash codes with file names.
3. A different server is maintained as a trusted center for distribution of public key.
4. Lastly there is a block for downloading the file. The file downloaded is decrypted then it's digital signature is verified before showing the file to the user.

MODIFICATIONS MADE BY US :

1. We used three different symmetric encryption algorithms rather than only one as it greatly increases the security.
2. We also secured the key for encryption and decryption by encoding it under an image using steganography concept which makes the storage even more secure.

RESEARCH PAPER 2:

<https://irjet.net/archives/V5/i3/IRJET-V5I3475.pdf>

SUMMARY :

In the above Research Paper they have discussed about Symmetric Key and Asymmetric key cryptography. They have also explained in details the DES standard I.e an algorithm that takes a fixed-length string of plaintext bits and transforms it through a series of complicated operations into another ciphertext bitstring of the same length.

They have implemented File security in clouds using cryptography . From this idea we will establish security of file management in OS using Cryptography DES Algorithm.

In this Research paper They have also discussed About AES (Advanced Encryption Standard) But we wont be using that.

RESEARCH PAPER 3:

<http://www.ijecs.in/index.php/ijecs/article/view/240/199>

SUMMARY :

In the above Research Paper They have discussed about Cryptography Algorithms like use of key rotations in data security in cloud file managements system. They have also discussed in brief about Block Status table (BST).

RESEARCH PAPER 4 :

<http://iaetsdjaras.org/gallery/22-march-570.pdf>

SUMMARY :

In the above research Paper They have discussed about Encryption and Decryption of file management for security purpose using Hybrid cryptography algorithms.

In this paper they have gave a very detailed explanation and implementation of hybrid cryptography algorithms .The hybrid cryptosystem uses a combination of:

Blowfish Algorithm coupled with File Splitting and Merging mechanism

SRNN Algorithm In a hybrid scheme, the performance of symmetric algorithm is integrated with security of asymmetric algorithm. The symmetric algorithm (Blowfish) used in hybrid

cryptosystem has best practice to avoid data misuse when compared with other symmetric algorithms. Also, in terms of throughput, Blowfish has best performance. The SRNN used serves as a good balance between speed and security. In hybrid cryptosystem, firstly, files uploaded files are sliced and each slice is encrypted by the corresponding key Blowfish key provided by the user. Secondly, each of the n keys are encrypted using SRNN where n is the number of slices.

RESEARCH PAPER 5:

https://www.researchgate.net/publication/305017745_Role_of_File_System_in_Operating_System

SUMMARY :

The above Research Paper gave us a deep insight on the role and importance of File management system in operating System.

RESEARCH PAPER 6:

https://www.researchgate.net/publication/303318872_A_Hybrid_Cryptography_Technique_for_Improving_Network_Security

SUMMARY :

The security of network and the network data is primary aspect of the network providers and service providers. Therefore during the data exchange the cryptographic techniques are utilized for securing the data during various communications. On the other hand the traditional cryptographic techniques are well known and the attackers are known about the solution. Therefore new kind of cryptographic technique is required which improve the security and complexity of data cipher. In this paper a hybrid cryptographic technique for improving data security during network transmission is proposed and their implementation and results are reported. The proposed secure cryptographic technique promises to provide the highly secure cipher generation technique using the RSA, DES and SHA1 technique. The implementation of the proposed technique is

provided using the JAVA technology and their performance in terms of space and time complexity is estimated and compared with the traditional RSA cryptography. The proposed cryptographic technique found the efficient and improved cipher text during comparative performance analysis.

RESEARCH PAPER 7:

<https://www.irjet.net/archives/V3/i1/IRJET-V3I1233.pdf>

SUMMARY :

Secure Communication in remote access is expected in server – client architecture and peer to peer devices. In-secure transmission may lead to leak sensitive credentials and information. Strong security policies are required to provide proper level of security policies to achieve confidentiality, authentication and integrity. To maintain confidentiality, Digital Envelope, which is the combination of the encrypted message and signature with the encrypted symmetric key, is also used. This research paper proposed a hybrid model to achieve confidentiality, authentication and integrity in same manner.

TABULAR ANALYSIS OF CRYPTOGRAPHY ALGORITHMS:

S.No	Algorithm	Features	Drawback and Attacks
1	RSA	1) Computational problem- integer factorization problem. 2) Encryption key size-1024 bit. 3) Application- Used in all worlds banking circle for security purpose and the transaction.	1) Drawback- Key size is large, so it is 15 times slower than ECC/ Need more memory and computing power and more battery in RSA use. 2) Attacks- Chosen-cipher text attack, Discrete- Logarithm-Attack, Men-In-The-Middle-Attack.
2	Diffie Hellman	1)Computational problem- Discrete Logarithm Problem, Decision Problem 2) Encryption key size-1024 bits. 3)Application- Key Exchange	1)Drawback- The value of private key K is smaller in size which can be easily understood and decoded. 2) Attacks- It used only in key exchange only
3	DSA	1)Computational problem- Discrete Logarithm Problem 2)Encryption key size- No Encryption key in DSA 3)Application- DSA is used for digital signature.	1)Drawback- a) Vulnerable DSS design. b) The second complaint regards the size of prime 512 bits. 2) Attacks-In known message attack, attacker given valid signatures for many types of messages which is known by attacker but not chosen by attacker. In an adaptive chosen message attack, the attacker first understands and learns signatures on arbitrary messages of the attacker's needs.
4	NTRU Encrypt	1)Computational problem- Closest vector problem in lattices 2) Encryption key size- The table shows the required sizes of NTRU key strength. NTRU 251, NTRU 347 and NTRU 503 provide roughly equivalent security to RSA 1024, 2048 and 4096. Key Strength Pre-master Secret Size NTRU 25 120 bytes, 347 32 bytes and 503 48 Bytes3) Application- Digital signature	1) Drawback-"Shor's algorithm" is quantum algorithm used to integer factorization and would be able to break ECC or RSA of any practical size in negligible time period. So NTRU's security is slightly vulnerable through quantum computers. 2) Attacks- Security issues-Elementary (why N should be prime), Standard(men-in-middle attack), Implementation(choosing-cipher attack, hashing and padding issues)
5	ElGamal	1)Computational problem- Discrete logarithm problem 2)Encryption key size-512 bits. 3) Application- It used in PGP Key exchange, Authentication, encryption and decryption of small messages.	1) Drawback-The encrypted message becomes very big in size, it becomes the twice the size of original message m.ElGamal's need large amount of space to store the three part public key and two part encrypted message. 2) Attacks- Low-Modulus attack, Known-plaintext attack

Table I. Comparative Analysis of Symmetric Encryption Algorithms					
Factors	DES	3DES	AES	E-DES	References
Key length (bits)	56	112, 168	128, 192 or 256	1024	Stallings[2]; Riman [23]; Agrawal et al. [9]
Cipher Type	Sym.	Sym.	Sym.	Sym.	Stallings[2]; Riman [23]
Block (bits)	64	64	128	128	Stallings[2]; Riman [23]
Rounds	16	48	10,12,14	16	Stallings[2]; Riman [23]
Developed	1975	1978	1998	2013	Stallings[2]; Riman [23]
Security	Not good	Passing	Secure	Secure	Hamdan[18]
Possible Key	2^{56}	2^{112}	2^{128}	2^{1024}	Hamdan[18] Riman [23]
Time for Brute Force key attack (10^{12} keys/sec)	<1 day	1.6×10^{14} years	10^{19} years	10^{152} years	Calculated in Section 4 part C.
Avalanche effect	Resists	Resists	Resists	Resists	Singh et al. [15]
Encryption Software	Fast	Slow	Medium	Very fast	Calculated in Section 4 part B
Time to encrypt 48KB	7s	21s	13s	2s	Calculated in Section 4 part B

CONTRIBUTIONS MADE:

I would like to thank my teammates and my respected professor who constantly guided and supported me throughout the project and always tried their best to resolve my doubts. Each one of our teammates have contributed thoroughly and the project would not have been possible without Their contribution.

In this project, I contributed by researching about our topic and domain on the internet.

I also contributed in designing the encoding and decoding of the keys required for encryption and decryption under images using steganography, coding of the peripherals of the application i.e. loading of the files from cloud on the system and dividing them into three parts and merging them and again uploading files on the cloud.

I also contributed by establishing The AWS cloud environment which was the cloud service we used for our project, and I also designed and compiled the final code by inserting my other members' code.

My Hardware Specifications :

System	
Processor:	Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.20 GHz
Installed memory (RAM):	16.0 GB (15.8 GB usable)
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	No Pen or Touch Input is available for this Display

My Software Specifications :

Cloud server/environment : Amazon Cloud Services (AWS)

Programming Language : Python 3.7

Database Query Language IDE : MySQL

MY CODING PART :

STEGANOGRAPHY CODE :

```
from PIL import Image

# Convert encoding data into 8-bit binary
# form using ASCII value of characters
def genData(data):
    # list of binary codes
    # of given data
    newd = []

    for i in data:
        newd.append(format(ord(i), '08b'))
    return newd

# Pixels are modified according to the
# 8-bit binary data and finally returned
def modPix(pix, data):
    datalist = genData(data)
    lendata = len(datalist)
    imdata = iter(pix)

    for i in range(lendata):

        # Extracting 3 pixels at a time
        pix = [value for value in imdata.__next__()[ :3] +
                imdata.__next__()[ :3] +
                imdata.__next__()[ :3]]

        # Pixel value should be made
        # odd for 1 and even for 0
        for j in range(0, 8):
            if (datalist[i][j] == '0' and pix[j] % 2 != 0):
                pix[j] -= 1

            elif (datalist[i][j] == '1' and pix[j] % 2 == 0):
                if (pix[j] != 0):
                    pix[j] -= 1
```

```
        pix[j] -= 1
    else:
        pix[j] += 1
    # pix[j] -= 1
```

```
# Eighth pixel of every set tells
# whether to stop or read further.
# 0 means keep reading; 1 means the
# message is over.
```

```
if (i == lendata - 1):
    if (pix[-1] % 2 == 0):
        if (pix[-1] != 0):
            pix[-1] -= 1
        else:
            pix[-1] += 1
```

```
    else:
        if (pix[-1] % 2 != 0):
            pix[-1] -= 1
```

```
    pix = tuple(pix)
    yield pix[0:3]
    yield pix[3:6]
    yield pix[6:9]
```

```
def encode_enc(newimg, data):
    w = newimg.size[0]
```

```
    (x, y) = (0, 0)
```

```
    for pixel in modPix(newimg.getdata(), data):
```

```
        # Putting modified pixels in the new image
        newimg.putpixel((x, y), pixel)
```

```
        if (x == w - 1):
            x = 0
            y += 1
```

```
        else:
            x += 1
```

```
# Encode data into image
def encode():
    img = input("Enter image name(with extension) : ")
    image = Image.open(img, 'r')

    data = input("Enter data to be encoded : ")
    if (len(data) == 0):
        raise ValueError('Data is empty')

    newimg = image.copy()
    encode_enc(newimg, data)

    new_img_name = input("Enter the name of new image(with extension) : ")
    newimg.save(new_img_name, str(new_img_name.split(".")[1].upper()))
```

```
# Decode the data in the image
def decode():
    img = input("Enter image name(with extension) : ")
    image = Image.open(img, 'r')

    data = ""
    imgdata = iter(image.getdata())

    while (True):
        pixels = [value for value in imgdata.__next__()[0:3] +
                  imgdata.__next__()[0:3] +
                  imgdata.__next__()[0:3]]

        # string of binary data
        binstr = ""

        for i in pixels[0:8]:
            if (i % 2 == 0):
                binstr += '0'
            else:
                binstr += '1'

        data += chr(int(binstr, 2))
```

```
if (pixels[-1] % 2 != 0):  
    return data
```

```
# Main Function
```

```
def main():  
    d = int(input(":: Welcome to Steganography ::\n"  
                  "1. Encode\n2. Decode\n"))  
    if (d == 1):  
        encode()  
  
    elif (d == 2):  
        print("Decoded Word : " + decode())  
    else:  
        raise Exception("Enter correct input")
```

```
main()
```

PERIPHERAL CODE

```
print("HELLO!! WELCOME TO 3 STEP SECURITY FILE  
STORAGE IN AWS")  
print("1. LOGIN")  
print("2. SIGN UP")  
a = int(input("ENTER YOUR CHOICE: "))  
os.system("cls")  
if a == 2:  
    name = input("ENTER YOUR NAME: ")  
    passwd = input("ENTER YOUR PASSWORD: ")  
    write_key()  
    st = "INSERT INTO file_sec(name,password,key_code)  
VALUES(%s,%s,%s);"  
    record = (name, passwd, key)  
    my_db = mysql.connector.connect(host="localhost",  
user="rajarshi", passwd="root", database="rajarshi",  
autocommit=True)  
    cur = my_db.cursor()  
    cur.execute(st, record)  
    print("YOUR ACCOUNT IS ALL SET!! PLEASE COPY THE  
KEY BELOW")
```



```

print(key)
print("PLEASE PRESS 3 TO PROCEED FURTHER...")
b = int(input())
os.system("cls")

if b == 3:
    print("WELCOME TO FINAL STEP OF SIGN UP
    PROCEDURE. KINDLY SELECT AN IMAGE AND PASTE UR
    KEY AS HIDDEN MESSAGE AND ENCODE IT.")
    encode()
    ss = "update file_sec set steg_file_name = %s where name
    = %s"
    rec = (new_img_name, name)
    cur.execute(ss, rec)
    print("YOU ARE ALL SET!!KINDLY EXIT AND RERUN
    FOR LOGIN BY PRESSING 4")
    q = int(input())
    if q == 4:
        os.system("exit")
    else :
        print("INVALID CHOICE")
else:
    uname = input("ENTER USERNAME : ")
    pas = input("ENTER PASSWORD : ")
    os.system("cls")
    print("IF U WANT TO SELECT A FILE FROM UR PC AND
    UPLOAD IT TO AWS USING 3SFS(PRESS1)")
    print("IF U WANT TO DOWNLOAD ENCRYPTED FILES
    FROM AWS AND DECRYPT THEM FOR VIEWING(PRESS
    2)")
    fg = int(input())
    os.system("cls")
    if fg == 1:
        my_db = mysql.connector.connect(host="localhost",
        user="rajarshi", passwd="root", database="rajarshi",
        autocommit=True)
        cur = my_db.cursor()
        cur.execute("select key_code from file_sec where name =
        '"+uname+"'")
        for i in cur:
            res = i[0]
            print("WELCOME TO THE MAIN PAGE")

```

```

    print("PLEASE GIVE THE FILE NAME(WITH
EXTENSION) WHICH YOU WANT TO STORE IN CLOUD: ")
    fname = input()
    nf1 = open(str(fname.split(".")[0])+'1.'+str(fname.split(".")[1]),
"w")
    nf2 = open(str(fname.split(".")[0])+'2.'+str(fname.split(".")[1]),
"w")
    nf3 = open(str(fname.split(".")[0])+'3.'+str(fname.split(".")[1]),
"w")
    with open(fname, "r", encoding="utf8") as file:
        data = file.readlines()
        si = len(data)
        for i in range(si//3):
            nf1.write(data[i])
        for j in range(si//3,2*si//3):
            nf2.write(data[j])
        for k in range(2*si//3,si):
            nf3.write(data[k])
    encrypt(str(fname.split(".")[0])+'1.'+str(fname.split(".")[1]),
res)
    print("FILE 1st SUBPART HAS BEEN SUCCESSFULLY
ENCRYPTED USING AES ENCRYPTION")
    encrypt(str(fname.split(".")[0])+'2.'+str(fname.split(".")[1]),
res)
    print("FILE 2nd SUBPART HAS BEEN SUCCESSFULLY
ENCRYPTED USING FERNET ENCRYPTION")
    encrypt(str(fname.split(".")[0])+'3.'+str(fname.split(".")[1]),
res)
    print("FILE 3rd SUBPART HAS BEEN SUCCESSFULLY
ENCRYPTED USING CAESAR & FIBONACCI
ENCRYPTION")

s3.upload_file(str(fname.split(".")[0])+'1FE.'+str(fname.split(".")[1]),
), 'os-project',str(fname.split(".")[0])+'1FE.'+str(fname.split(".")[1]))
    print("FILE 1 UPLOADED SUCCESSFULLY")
    s3.upload_file(str(fname.split(".")[0]) + '2FE.' +
str(fname.split(".")[1]), 'os-project',
                str(fname.split(".")[0]) + '2FE.' +
str(fname.split(".")[1]))
    print("FILE 2 UPLOADED SUCCESSFULLY")
    s3.upload_file(str(fname.split(".")[0]) + '3FE.' +

```

```

str(fname.split(".")[1]), 'os-project',
        str(fname.split(".")[0]) + '3FE.' +
str(fname.split(".")[1]))
    print("FILE 3 UPLOADED SUCCESSFULLY")
    print("FILE HAS BEEN UPLOADED SECURELY USING
3SES. PLEASE PRESS 5 TO EXIT THE SYSTEM")
    h = int(input())
    if h == 4:
        os.system("exit")
    else:
        print("INVALID COMMAND")
    else:
        bname = input("ENTER THE NAME OF THE FILE THAT
YOU ENCRYPTED : ")
        s3.download_file('os-
project',str(bname.split(".")[0])+ '1FE.'+str(bname.split(".")[1]),

str(bname.split(".")[0])+ '1DE.'+str(bname.split(".")[1]))
        print("FILE 1 HAS BEEN SUCCESSFULLY
DOWNLOADED FROM AWS")
        s3.download_file('os-
project',str(bname.split(".")[0])+ '2FE.'+str(bname.split(".")[1]),
            str(bname.split(".")[0]) + '2DE.' +
str(bname.split(".")[1]))
        print("FILE 2 HAS BEEN SUCCESSFULLY
DOWNLOADED FROM AWS")
        s3.download_file('os-project',str(bname.split(".")[0]) + '3FE.'
+ str(bname.split(".")[1]),
            str(bname.split(".")[0]) + '3DE.' +
str(bname.split(".")[1]))
        print("FILE 3 HAS BEEN SUCCESSFULLY
DOWNLOADED FROM AWS")
        my_db = mysql.connector.connect(host="localhost",
user="rajarshi", passwd="root", database="rajarshi",
autocommit=True)
        cur = my_db.cursor()
        cur.execute("select key_code from file_sec where name = '" +
uname + "';")
        for i in cur:
            res1 = i[0]

```

```

decrypt_new(str(bname.split(".")[0])+'1.'+str(bname.split(".")[1]),
str("decrypt_"+bname.split(".")[0]+'1.'+str(bname.split(".")[1]))
    print("SUCCESSFULLY DECRYPTED FILE No 1!!!")

decrypt_new(str(bname.split(".")[0])+'2.'+str(bname.split(".")[1]),
str("decrypt_"+bname.split(".")[0]+'2.'+str(bname.split(".")[1]))
    print("SUCCESSFULLY DECRYPTED FILE No 2!!!")

decrypt_new(str(bname.split(".")[0])+'3.'+str(bname.split(".")[1]),
str("decrypt_"+bname.split(".")[0]+'3.'+str(bname.split(".")[1]))
    print("SUCCESSFULLY DECRYPTED FILE No 3!!!")

merge_file(str("decrypt_"+bname.split(".")[0]+'1.'+str(bname.split
(".")))[1]),

str("decrypt_"+bname.split(".")[0]+'2.'+str(bname.split(".")[1]),

str("decrypt_"+bname.split(".")[0]+'3.'+str(bname.split(".")[1]))
    print("FILE HAS BEEN SAVED IN UR SPECIFIED
DIRECTORY.")

```

PROBLEM STATEMENT:

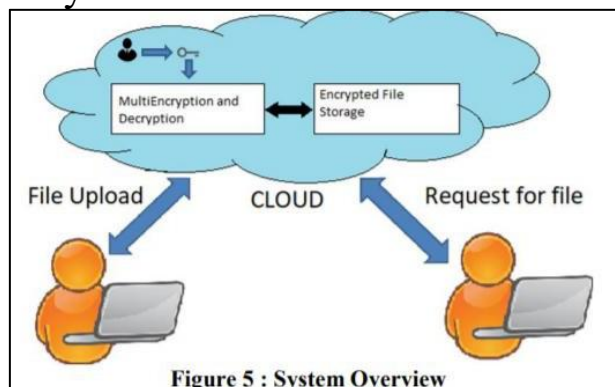
With the increasing number of data stored in the internet, people rely on cloud systems to have better access to their files from multiple devices and multiple locations. Hence the data stored in cloud servers must be strongly encrypted so that attackers can't access valuable information and our users have their data secure in the cloud systems.

OBJECTIVE:

With this project we will have an efficient and secure way of storing files in a cloud system with a very strong encryption protocol.

PROPOSED SYSTEM FRAMEWORK:

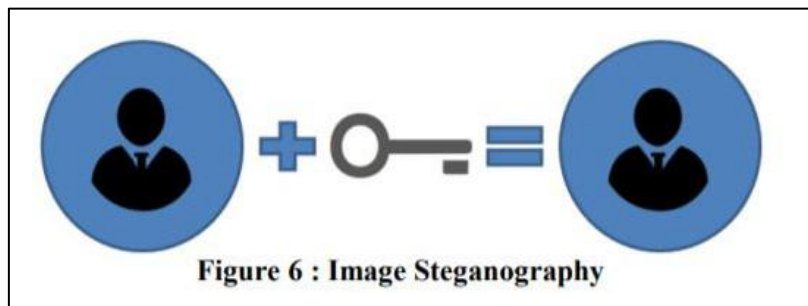
In the proposed system, a method for securely storing files in the cloud using a hybrid cryptography algorithm is presented. In this system, the user can store the file safely in online cloud storage as these files will be stored in encrypted form in the cloud and only the authorized user has access to their files.



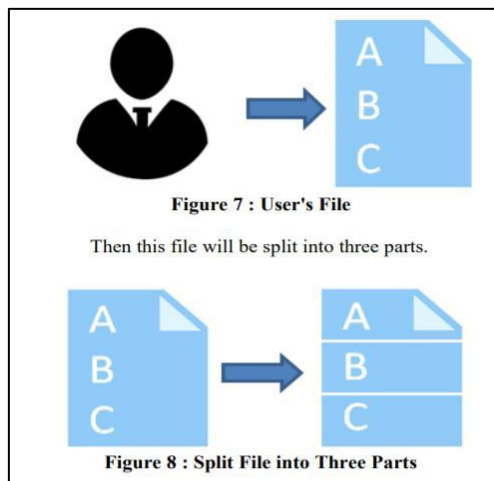
The above figure gives an overview of the system. As in the above figure, the files that the user will upload on the cloud will be encrypted with a user-specific key and store safely on the cloud

1) Registration of User For accessing the services the user must first register themselves. During the registration process various data like the name, username and password will be requested to enter. Using this data the server will produce unique user-specific keys that will be used for the encryption and decryption purpose. But this key will not be stored in the database instead it will be stored using the

steganography algorithm in an image that will be used as the user's profile picture.



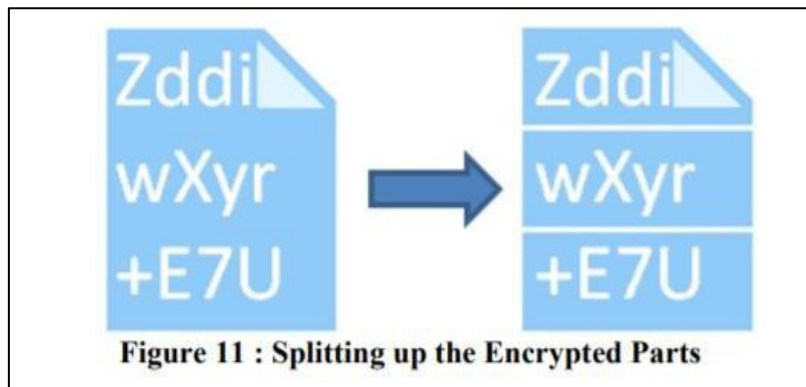
2) Uploading a File on Cloud When the user uploads a file on the cloud first it will be uploaded in a temporary folder.



These three parts will be encrypted using cryptographic algorithms. Every part will use a different encryption algorithm. These three parts will be encrypted using three different algorithms that are AES, Diffie Hellman and Fernet. The key to these algorithms will be retrieved from the steganographic image created during the registration.

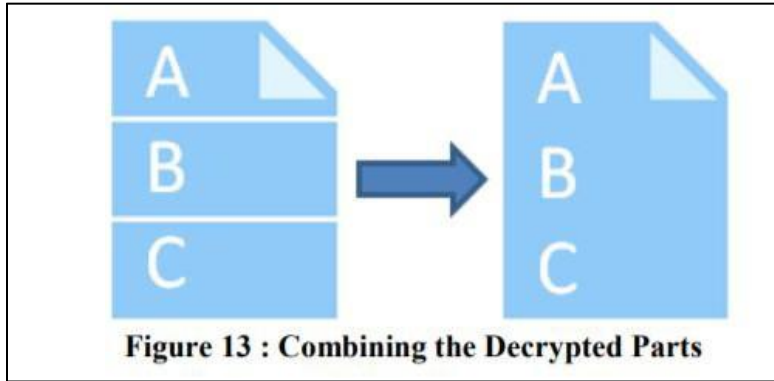
After the split encryption, the file reassembled and stored in the user's specific folder. The original file is removed from the temporary folder.

3) Downloading a File from the Cloud When the user requests a file to be downloaded first the file is split into three parts.

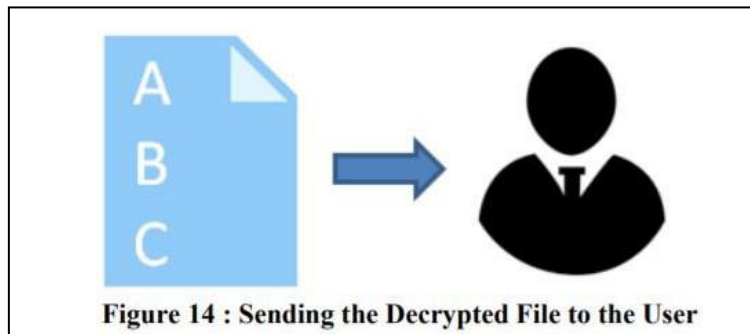


Then these three parts will be decrypted using the same algorithms with which they were encrypted. The key to the algorithms for the decryption process will be retrieved from the steganographic image created during the registration.

Then these parts will be re-combined to form a fully decrypted file.



Then this file will be sent to the user for download.



MODULES DESCRIPTION:

- 1. Advanced Encryption Standard (AES)** – The AES algorithm is related to Rijndael's encryption. Rijndael is a family of encryption algorithms with different keys and block sizes. It consists of a continues serial operations, some of them involve the input of certain outputs (substitutions) and others the mixing of bits (permutations). All AES calculations algorithm is executed in bytes instead of bits. Therefore, for Advanced Encryption Standard, 128 bits of plain data is considered as a block of 16 bytes These 16 bytes are arranged in a 4x4 matrix for the processing.
- 2. Fernet Encryption** –Fernet is a symmetric encryption method which makes sure that the message encrypted cannot be manipulated/read without the key. It uses URL safe encoding for the keys. Fernet also uses 128-bit AES in CBC mode and PKCS7 padding, with HMAC using SHA256 for authentication. The IV is created from os.random(). All of this is the kind of thing that good software needs.AES is top drawer encryption, and SHA-256 avoids many of the problems caused by MD5 and SHA-1 (as the length of the hash values is too small). With CBC (Cipher Block Chaining) we get a salted output, and which is based on a random value (the IV value). And with HMAC we can provide authenticated access from both sides.And for PKCS7, that's a standard too. Fernet is used to define best practice cryptography methods, and Hazmat supports core cryptographical primitives
- 3. Diffie Hellman Encryption** - The Diffie-Hellman algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.

For the sake of simplicity and practical implementation of the algorithm, we will consider only 4 variables one prime P and G (a primitive root of P) and two private values a and b.

P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly, the opposite person received the key and from that generates a secret key after which they have the same secret key to encrypt.

IMPLEMENTATION OF THE PROJECT

CODE :

NOTE : code is written in Pycharm CE IDE

```
import boto3
import tqdm
import mysql.connector
import sys
from cryptography.fernet import Fernet
from PIL import Image
import os

global s3
s3 = boto3.client('s3')

def percent(part, total):
    return 100 * float(part) / float(total)

def WriteDataToFilePrompt(data, newline):
    filename = input("Filename: ")
    file = open(filename, 'w')
    location = 0
    while location < len(data):
        if newline == True:
            file.writelines(data[location] + "\n")
        else:
            file.writelines(data[location])
        location += 1
```

```
file.close()
```

```
class Fibonacci():
    def __init__(self):
        pass

    def Single(self, n): # Returns a single Fibonacci number
        if n == 0:
            return 0
        elif n == 1:
            return 1
        else:
            return self.Single(n - 1) + self.Single(n - 2)

    def SetOfFibonacci(self, start, end): # Returns a set of Fibonacci numbers
from start to end.
        returnData = []
        location = start
        while location < end:
            returnData.append(str(self.Single(location)))
            location += 1
        return returnData
```

```
fib = Fibonacci()
encryptionList = fib.SetOfFibonacci(1, 11)
```

```
class FibonacciEncryption():
    def __init__(self):
        pass

    def Encrypt(self, textToEncrypt):
        location = 0
        encryptedText = []
        offsetIndex = 0
        while location < len(textToEncrypt):
            if offsetIndex < 9: # Let's make sure we keep it in-bounds.
                offsetIndex += 1
            elif offsetIndex == 9:
                offsetIndex = 0
            charValue = ord(textToEncrypt[location])
            offsetValue = encryptionList[offsetIndex]
            finalValue = int(charValue) + int(offsetValue)
            encryptedText.append(str(finalValue))
            location += 1
```

```

location = 0
finalString = ""
while location < len(encryptedText):
    finalString += str(encryptedText[location])
    if location != len(encryptedText):
        finalString += " "
    location += 1
return finalString

def Decrypt(self, textToDecrypt): # Anything other than a string input
will break this
    location = 0
    decryptedText = ""
    offsetIndex = 0
    nextwhitespace = 0
    while location < len(textToDecrypt):
        if offsetIndex < 9:
            offsetIndex += 1
        elif offsetIndex == 9:
            offsetIndex = 0
        nextwhitespace = textToDecrypt.find(" ", location)
        if nextwhitespace != -1:
            tempTextToDecrypt = textToDecrypt[location:nextwhitespace]
            offsetValue = encryptionList[offsetIndex]
            finalText = chr(int(tempTextToDecrypt) - int(offsetValue))
            decryptedText += finalText
        if nextwhitespace < location:
            return decryptedText
        else:
            location = nextwhitespace + 1

fibe = FibonacciEncryption()

class AESEncryption:

    def __init__(self, filename): # Constructor
        self.filename = filename

    def encryption(self): # Allows us to perform file operation

        try:
            original_information = open(self.filename, 'rb')

        except (IOError, FileNotFoundError):
            print('File with name {} is not found.'.format(self.filename))
            sys.exit(0)

```

```
try:
```

```
    encrypted_file_name = 'cipher_' + self.filename  
    encrypted_file_object = open(encrypted_file_name, 'wb')
```

```
    content = original_information.read()  
    content = bytearray(content)
```

```
    key1 = 192  
    print('Encryption Process is in progress...!')  
    for i, val in tqdm(enumerate(content)):  
        content[i] = val ^ key1
```

```
    encrypted_file_object.write(content)
```

```
except Exception:
```

```
    print('Something went wrong with {}'.format(self.filename))
```

```
finally:
```

```
    encrypted_file_object.close()  
    original_information.close()
```

```
class AESDecryption:
```

```
    def __init__(self, filename):  
        self.filename = filename
```

```
    def decryption(self): # produces the original result
```

```
        try:
```

```
            encrypted_file_object = open(self.filename, 'rb')
```

```
        except (FileNotFoundError, IOError):
```

```
            print('File with name {} is not found'.format(self.filename))  
            sys.exit(0)
```

```
        try:
```

```
            decrypted_file = input('Enter the filename for the Decryption file  
with extension:') # Decrypted file as output
```

```
            decrypted_file_object = open(decrypted_file, 'wb')
```

```
            cipher_text = encrypted_file_object.read()
```

```
            key1 = 192
```

```

cipher_text = bytearray(cipher_text)

print('Decryption Process is in progress...!')

for i, val in tqdm(enumerate(cipher_text)):
    cipher_text[i] = val ^ key1

decrypted_file_object.write(cipher_text)

except Exception:
    print('Some problem with Ciphertext unable to handle.')

finally:
    encrypted_file_object.close()
    decrypted_file_object.close()

def genData(data):
    # list of binary codes
    # of given data
    newd = []

    for i in data:
        newd.append(format(ord(i), '08b'))
    return newd

# Pixels are modified according to the
# 8-bit binary data and finally returned
def modPix(pix, data):
    datalist = genData(data)
    lendata = len(datalist)
    imdata = iter(pix)

    for i in range(lendata):

        # Extracting 3 pixels at a time
        pix = [value for value in imdata.__next__():3] +
            imdata.__next__():3] +
            imdata.__next__():3]

        # Pixel value should be made
        # odd for 1 and even for 0
        for j in range(0, 8):
            if (datalist[i][j] == '0' and pix[j] % 2 != 0):
                pix[j] -= 1

```

```

        elif (datalist[i][j] == '1' and pix[j] % 2 == 0):
            if (pix[j] != 0):
                pix[j] -= 1
            else:
                pix[j] += 1
            # pix[j] -= 1

# Eighth pixel of every set tells
# whether to stop or read further.
# 0 means keep reading; 1 means the
# message is over.
if (i == lendata - 1):
    if (pix[-1] % 2 == 0):
        if (pix[-1] != 0):
            pix[-1] -= 1
        else:
            pix[-1] += 1

    else:
        if pix[-1] % 2 != 0:
            pix[-1] -= 1

pix = tuple(pix)
yield pix[0:3]
yield pix[3:6]
yield pix[6:9]

def encode_enc(newimg, data):
    w = newimg.size[0]
    (x, y) = (0, 0)

    for pixel in modPix(newimg.getdata(), data):

        # Putting modified pixels in the new image
        newimg.putpixel((x, y), pixel)
        if (x == w - 1):
            x = 0
            y += 1
        else:
            x += 1

def merge_file(f, ff, fff):
    data = data2 = data3 = ""

```



```

# Reading data from file1
with open(f) as fp:
    data = fp.read()

# Reading data from file2
with open(ff) as fp:
    data2 = fp.read()

with open(fff) as fp:
    data3 = fp.read()
# Merging 3 files
# To add the data of file2
# from next line
data += "\n"
data += data2
data += "\n"
data += data3

with open('final_project.txt', 'w') as fp:
    fp.write(data)

def decrypt_new(fileq, filea):
    with open(fileq) as f:
        with open(filea, "w") as f1:
            for line in f:
                f1.write(line)

# Encode data into image
def encode():
    global new_img_name
    img = input("Enter image name(with extension) : ")
    image = Image.open(img, 'r')

    data = input("Enter data to be encoded : ")
    if (len(data) == 0):
        raise ValueError('Data is empty')

    newimg = image.copy()
    encode_enc(newimg, data)

    new_img_name = input("Enter the name of new image(with extension) : ")
    newimg.save(new_img_name)

# Decode the data in the image

```

```

def decode():
    imgg = input("Enter image name(with extension) : ")
    image = Image.open(imgg, 'r')

    data = ""
    imgdata = iter(image.getdata())

    while (True):
        pixels = [value for value in imgdata.__next__()[0:3] +
                    imgdata.__next__()[0:3] +
                    imgdata.__next__()[0:3]]

        # string of binary data
        binstr = ""

        for i in pixels[0:8]:
            if (i % 2 == 0):
                binstr += '0'
            else:
                binstr += '1'

        data += chr(int(binstr, 2))
        if (pixels[-1] % 2 != 0):
            return data

def write_key():
    global key
    key = Fernet.generate_key()

# def load_key():

def encrypt(filename, key):
    f = Fernet(key)
    with open(filename, "rb") as file:
        file_data = file.read()
        encrypted_data = f.encrypt(file_data)
        with open(str(filename.split(".")[0])+'FE.'+str(filename.split(".")[1]),
                  "wb") as file1:
            file1.write(encrypted_data)
        print("")

def decrypt(filename, key):
    f = Fernet(key)

```

```

with open(filename, "rb") as file:
    # read the encrypted data
    encrypted_data = file.read()
    # decrypt data
    decrypted_data = f.decrypt(encrypted_data)
    # write the original file
    with
open(str("decrypt_" + filename.split(".")[0] + '.' + str(filename.split(".")[1]),
"wb") as file7:
    file7.write(decrypted_data)

print("HELLO!! WELCOME TO 3 STEP SECURITY FILE STORAGE IN
AWS")
print("1. LOGIN")
print("2. SIGN UP")
a = int(input("ENTER YOUR CHOICE: "))
os.system("cls")
if a == 2:
    name = input("ENTER YOUR NAME: ")
    passwd = input("ENTER YOUR PASSWORD: ")
    write_key()
    st = "INSERT INTO file_sec(name,password,key_code)
VALUES(%s,%s,%s);"
    record = (name, passwd, key)
    my_db = mysql.connector.connect(host="localhost", user="rajarshi",
passwd="root", database="rajarshi",
autocommit=True)
    cur = my_db.cursor()
    cur.execute(st, record)
    print("YOUR ACCOUNT IS ALL SET!! PLEASE COPY THE KEY
BELOW")
    print(key)
    print("PLEASE PRESS 3 TO PROCEED FURTHER...")
    b = int(input())
    os.system("cls")
    if b == 3:
        print("WELCOME TO FINAL STEP OF SIGN UP PROCEDURE.
KINDLY SELECT AN IMAGE AND PASTE UR KEY AS HIDDEN
MESSAGE AND ENCODE IT.")
        encode()
        ss = "update file_sec set steg_file_name = %s where name = %s"
        rec = (new_img_name, name)
        cur.execute(ss, rec)
        print("YOU ARE ALL SET!!KINDLY EXIT AND RERUN FOR
LOGIN BY PRESSING 4")
        q = int(input())

```

```

        if q == 4:
            os.system("exit")
        else :
            print("INVALID CHOICE")
else:
    uname = input("ENTER USERNAME : ")
    pas = input("ENTER PASSWORD : ")
    os.system("cls")
    print("IF U WANT TO SELECT A FILE FROM UR PC AND UPLOAD IT TO AWS USING 3SFS(PRESS1)")
    print("IF U WANT TO DOWNLOAD ENCRYPTED FILES FROM AWS AND DECRYPT THEM FOR VIEWING(PRESS 2)")
    fg = int(input())
    os.system("cls")
    if fg == 1:
        my_db = mysql.connector.connect(host="localhost", user="rajarshi",
        passwd="root", database="rajarshi",
        autocommit=True)

        cur = my_db.cursor()
        cur.execute("select key_code from file_sec where name =
        '"+uname+"'")
        for i in cur:
            res = i[0]
            print("WELCOME TO THE MAIN PAGE")
            print("PLEASE GIVE THE FILE NAME(WITH EXTENSION) WHICH YOU WANT TO STORE IN CLOUD: ")
            fname = input()
            nf1 = open(str(fname.split(".")[0])+1.'+str(fname.split(".")[1]), "w")
            nf2 = open(str(fname.split(".")[0])+2.'+str(fname.split(".")[1]), "w")
            nf3 = open(str(fname.split(".")[0])+3.'+str(fname.split(".")[1]), "w")
            with open(fname, "r", encoding="utf8") as file:
                data = file.readlines()
                si = len(data)
                for i in range(si//3):
                    nf1.write(data[i])
                for j in range(si//3,2*si//3):
                    nf2.write(data[j])
                for k in range(2*si//3,si):
                    nf3.write(data[k])
            encrypt(str(fname.split(".")[0])+1.'+str(fname.split(".")[1]), res)
            print("FILE 1st SUBPART HAS BEEN SUCCESSFULLY ENCRYPTED USING AES ENCRYPTION")
            AESencrypt(str(fname.split(".")[0])+2.'+str(fname.split(".")[1]), res)
            print("FILE 2nd SUBPART HAS BEEN SUCCESSFULLY ENCRYPTED USING FERNET ENCRYPTION")
            DFHencrypt(str(fname.split(".")[0])+3.'+str(fname.split(".")[1]), res)
            print("FILE 3rd SUBPART HAS BEEN SUCCESSFULLY

```

```

    ENCRYPTED USING CAESAR & FIBONACCI ENCRYPTION")
    s3.upload_file(str(fname.split(".")[0])+'1FE.'+str(fname.split(".")[1]),
'os-project',str(fname.split(".")[0])+'1FE.'+str(fname.split(".")[1]))
    print("FILE 1 UPLOADED SUCCESSFULLY")
    s3.upload_file(str(fname.split(".")[0]) + '2FE.' + str(fname.split(".")[1]),
'os-project',
                    str(fname.split(".")[0]) + '2FE.' + str(fname.split(".")[1]))
    print("FILE 2 UPLOADED SUCCESSFULLY")
    s3.upload_file(str(fname.split(".")[0]) + '3FE.' + str(fname.split(".")[1]),
'os-project',
                    str(fname.split(".")[0]) + '3FE.' + str(fname.split(".")[1]))
    print("FILE 3 UPLOADED SUCCESSFULLY")
    print("FILE HAS BEEN UPLOADED SECURELY USING 3SES.
PLEASE PRESS 5 TO EXIT THE SYSTEM")
    h = int(input())
    if h == 4:
        os.system("exit")
    else:
        print("INVALID COMMAND")
    else:
        bname = input("ENTER THE NAME OF THE FILE THAT YOU
    ENCRYPTED : ")
        s3.download_file('os-
project',str(bname.split(".")[0])+'1FE.'+str(bname.split(".")[1]),
                    str(bname.split(".")[0])+'1DE.'+str(bname.split(".")[1]))
        print("FILE 1 HAS BEEN SUCCESSFULLY DOWNLOADED
    FROM AWS")
        s3.download_file('os-
project',str(bname.split(".")[0])+'2FE.'+str(bname.split(".")[1]),
                    str(bname.split(".")[0]) + '2DE.' + str(bname.split(".")[1]))
        print("FILE 2 HAS BEEN SUCCESSFULLY DOWNLOADED
    FROM AWS")
        s3.download_file('os-project',str(bname.split(".")[0]) + '3FE.' +
str(bname.split(".")[1]),
                    str(bname.split(".")[0]) + '3DE.' + str(bname.split(".")[1]))
        print("FILE 3 HAS BEEN SUCCESSFULLY DOWNLOADED
    FROM AWS")
        my_db = mysql.connector.connect(host="localhost", user="rajarshi",
passwd="root", database="rajarshi",
                    autocommit=True)
        cur = my_db.cursor()
        cur.execute("select key_code from file_sec where name = '" + uname +
";")
        for i in cur:
            res1 = i[0]
            decrypt_new(str(bname.split(".")[0])+'1.'+str(bname.split(".")[1]),
                    str("decrypt "+bname.split(".")[0]+'1.'+str(bname.split(".")[1]))

```

```

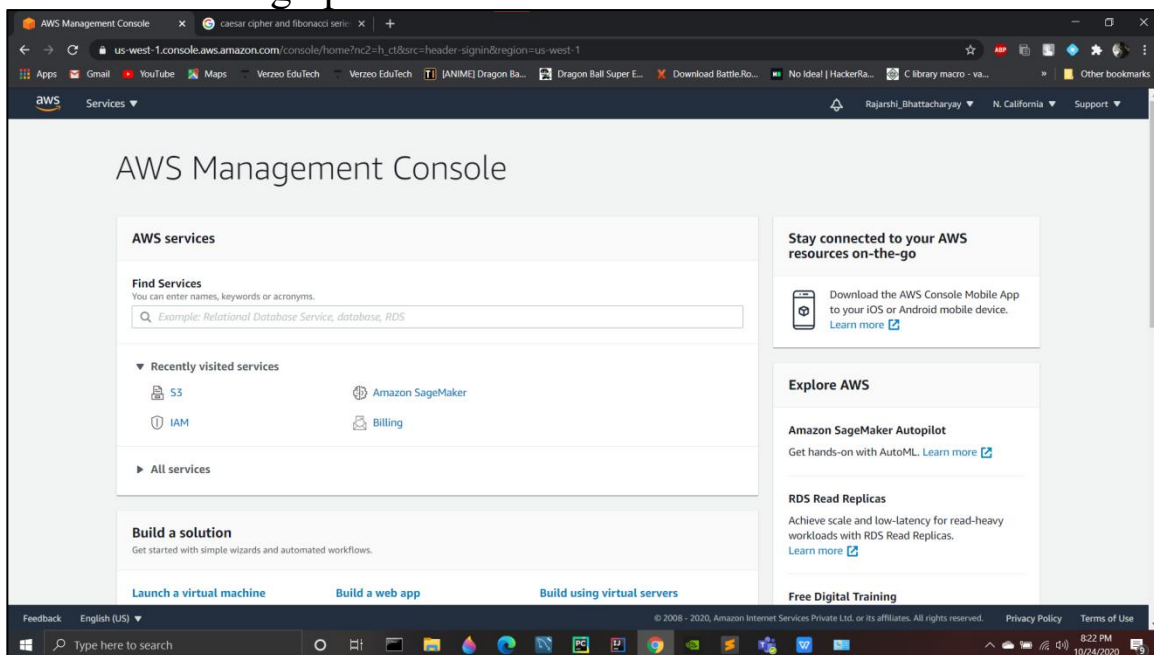
print("SUCCESSFULLY DECRYPTED FILE No 1!!!")
AESdecrypt_new(str(bname.split(".")[0])+'2.'+str(bname.split(".")[1]),
               str("decrypt_"+bname.split(".")[0])+'2.'+str(bname.split(".")[1]))
print("SUCCESSFULLY DECRYPTED FILE No 2!!!")
DFHdecrypt_new(str(bname.split(".")[0])+'3.'+str(bname.split(".")[1]),
               str("decrypt_"+bname.split(".")[0])+'3.'+str(bname.split(".")[1]))
print("SUCCESSFULLY DECRYPTED FILE No 3!!!")

merge_file(str("decrypt_"+bname.split(".")[0])+'1.'+str(bname.split(".")[1]),
           str("decrypt_"+bname.split(".")[0])+'2.'+str(bname.split(".")[1]),
           str("decrypt_"+bname.split(".")[0])+'3.'+str(bname.split(".")[1]))
print("FILE HAS BEEN SAVED IN UR SPECIFIED DIRECTORY.")

```

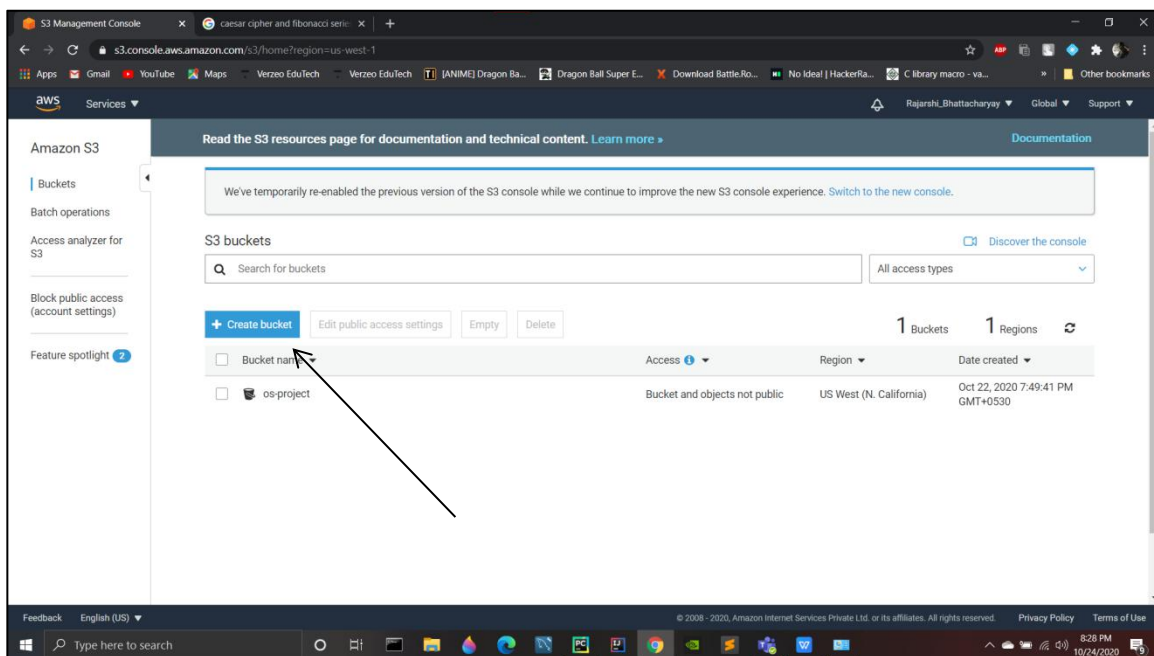
SCREEN-SHOTS

STEP 1 : Setting up the cloud environment

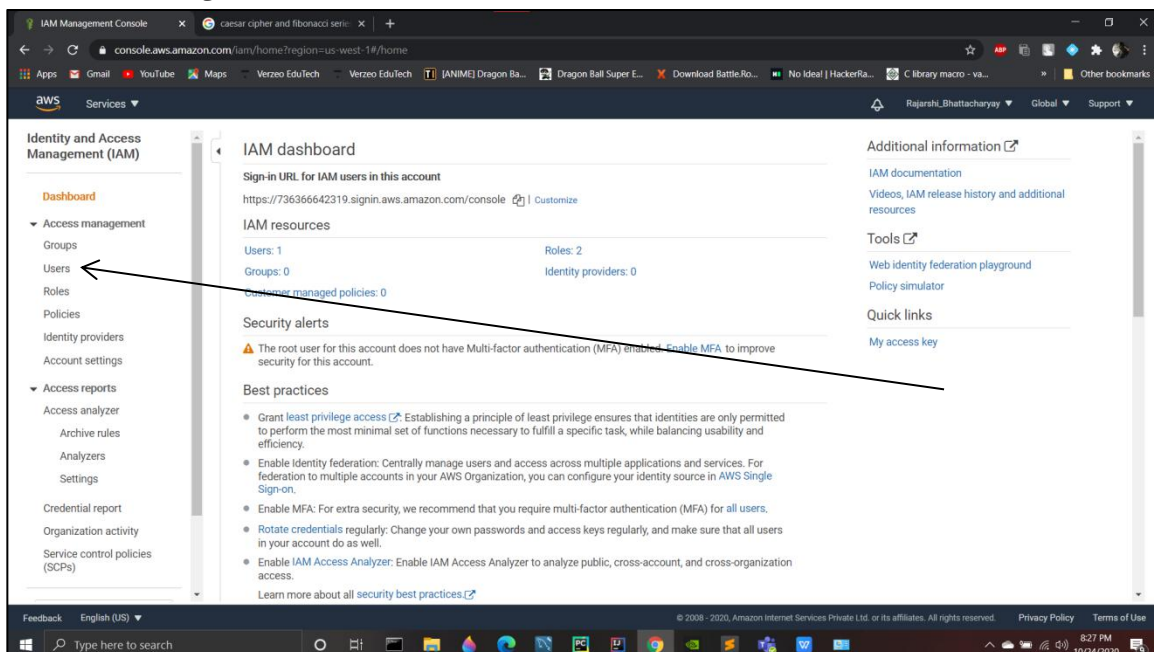


STEP 2 : Configuring the cloud on command line interface by registering the required credentials

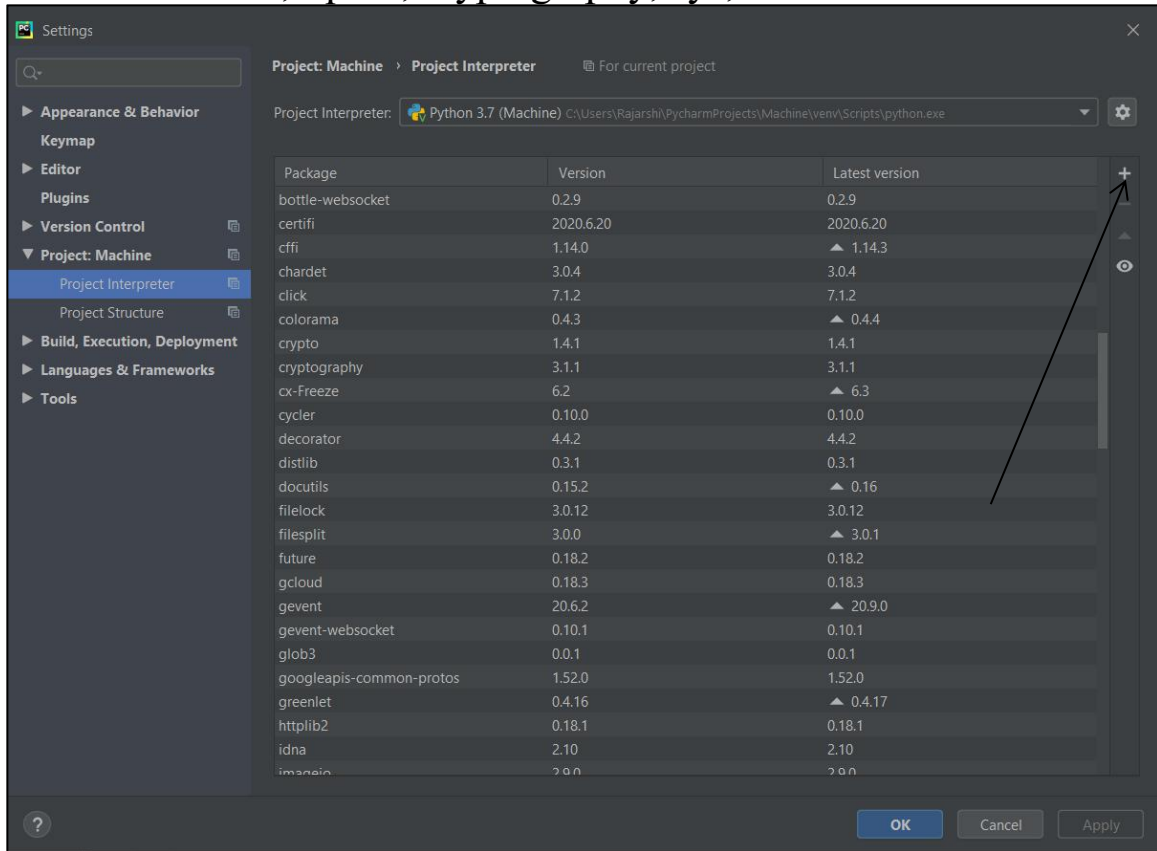
STEP 3 : Creating a Bucket in S3 of AWS



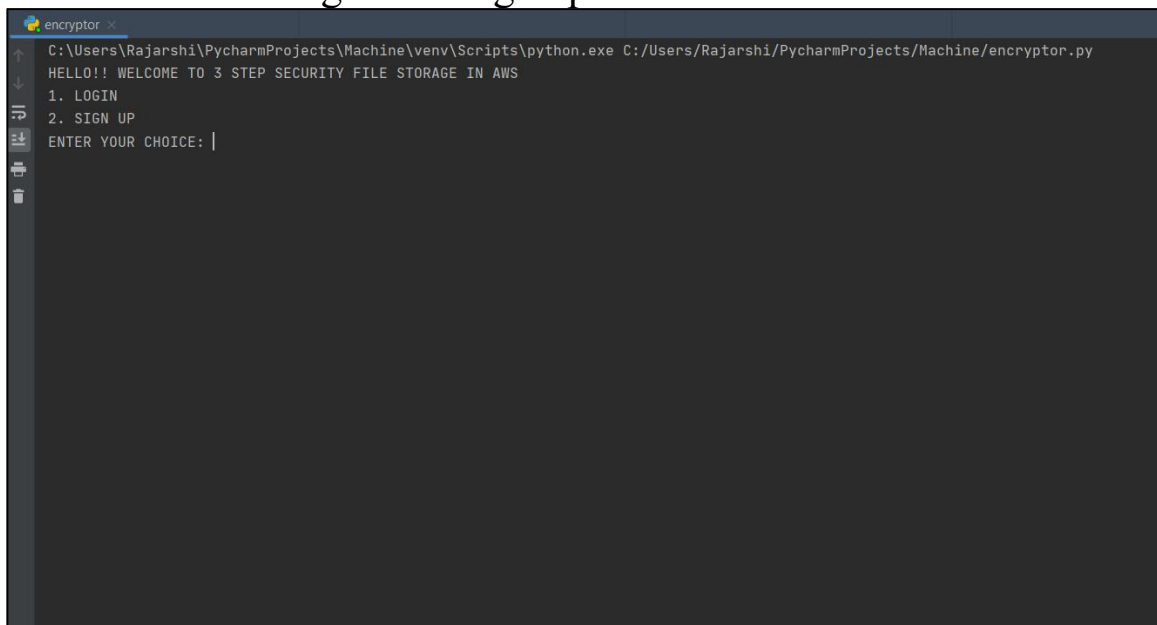
STEP 4 : Creating a user in AWS by which we can access the bucket of S3



STEP 5 : Install and import all the required libraries in python i.e. Boto3, tqdm , cryptography, sys, os



STEP 6 : Either Sign in or sign up as a User



STEP 7 : Select a file from your local system which you want to encrypt and upload on the cloud


```

C:\Users\Rajarshi\PycharmProjects\Machine\venv\Scripts\python.exe C:/Users/Rajarshi/PycharmProjects/Machine/encryptor.py
HELLO!! WELCOME TO 3 STEP SECURITY FILE STORAGE IN AWS
1. LOGIN
2. SIGN UP
ENTER YOUR CHOICE: 1
ENTER USERNAME : RAJARSHI
ENTER PASSWORD : RAJARSHI
IF U WANT TO SELECT A FILE FROM UR PC AND UPLOAD IT TO AWS USING 3SFS(PRESS1)
IF U WANT TO DOWNLOAD ENCRYPTED FILES FROM AWS AND DECRYPT THEM FOR VIEWING(PRESS 2)
1
WELCOME TO THE MAIN PAGE
PLEASE GIVE THE FILE NAME(WITH EXTENSION) WHICH YOU WANT TO STORE IN CLOUD:
|

```

STEP 8 : Divide the File into 3 parts and encrypt them individually using AES, Fernet and Diffie Hellman Encryption.

```

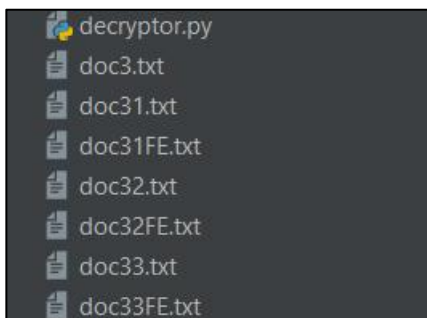
WELCOME TO THE MAIN PAGE
PLEASE GIVE THE FILE NAME(WITH EXTENSION) WHICH YOU WANT TO STORE IN CLOUD:
doc3.txt

FILE 1st SUBPART HAS BEEN SUCCESSFULLY ENCRYPTED USING AES ENCRYPTION

FILE 2nd SUBPART HAS BEEN SUCCESSFULLY ENCRYPTED USING FERNET ENCRYPTION

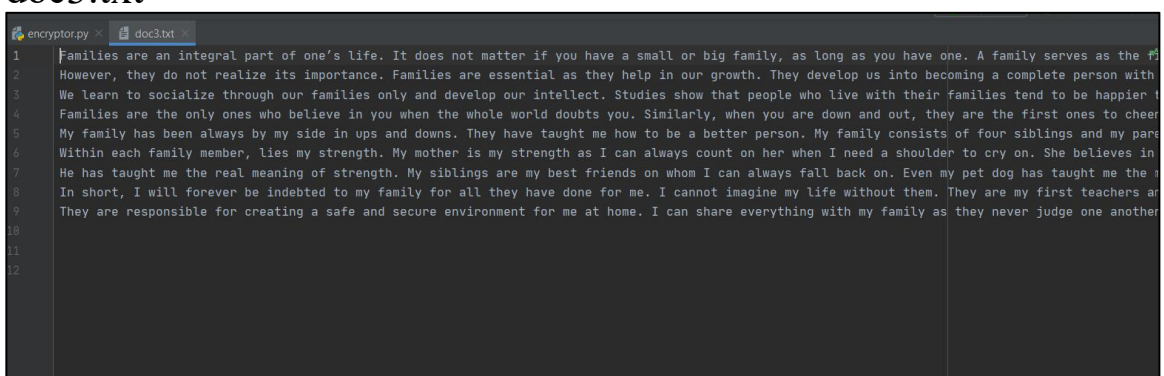
FILE 3rd SUBPART HAS BEEN SUCCESSFULLY ENCRYPTED USING CAESAR & FIBONACCI ENCRYPTION
FILE 1 UPLOADED SUCCESSFULLY
FILE 2 UPLOADED SUCCESSFULLY
FILE 3 UPLOADED SUCCESSFULLY
FILE HAS BEEN UPLOADED SECURELY USING 3SES. PLEASE PRESS 5 TO EXIT THE SYSTEM

```

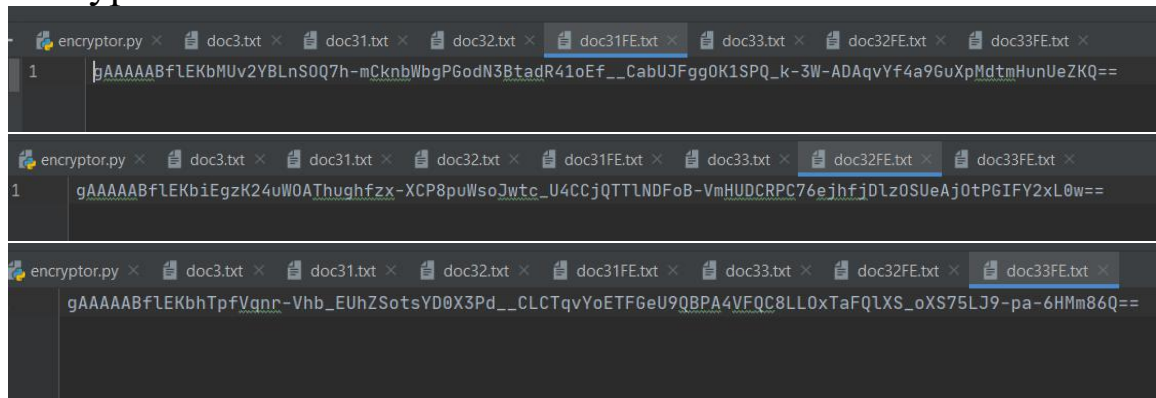


Here doc3.txt is the original file and doc31.txt, doc32.txt, doc33.txt are the three subfiles and the doct31FE.txt, doc32FE.txt and doc33FE.txt are the encrypted versions of the respective subfiles

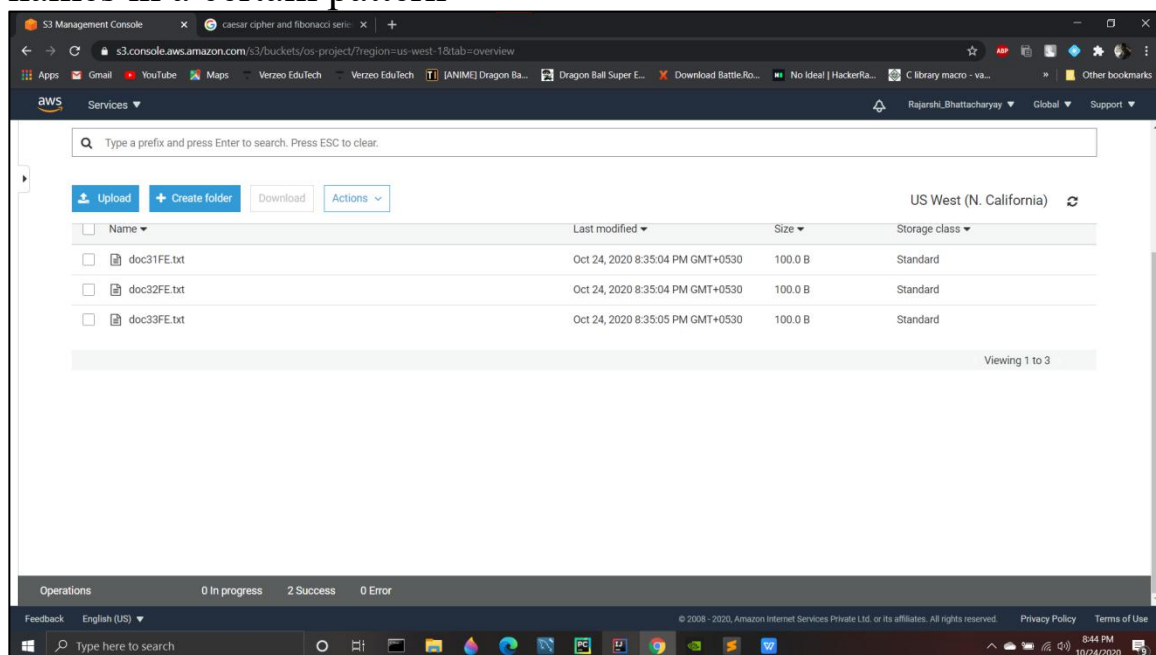
doc3.txt



Encrypted files look like :



STEP 9 : Now upload them individually on the cloud with the file names in a certain pattern



STEP 10 : Secure File Upload is Completed

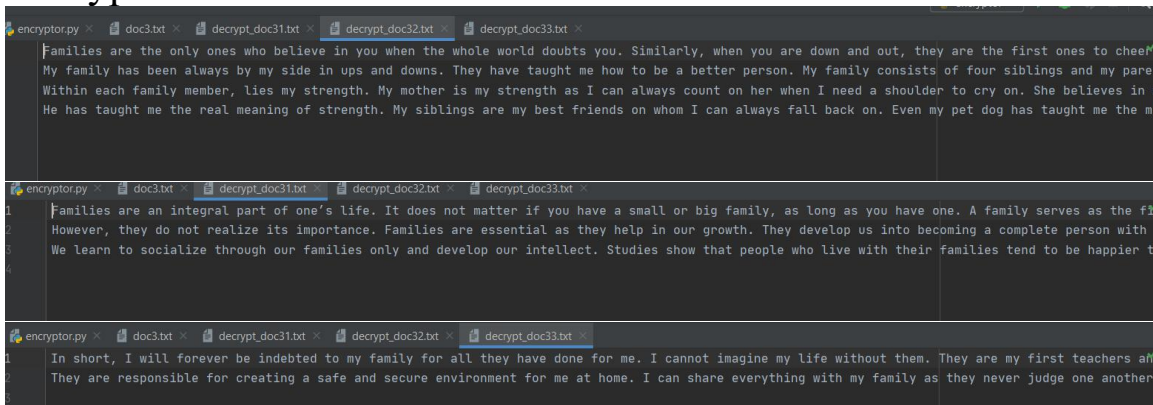
STEP 11 : If you want to download a file , then sign in as a user and enter the name of the original file which you uploaded on cloud.

```
C:\Users\Rajarshi\PycharmProjects\Machine\venv\Scripts\python.exe C:/Users/Rajarshi/PycharmProjects/Machine/encryptor.py
HELLO!! WELCOME TO 3 STEP SECURITY FILE STORAGE IN AWS
1. LOGIN
2. SIGN UP
ENTER YOUR CHOICE: 1
ENTER USERNAME : RAJARSHI
ENTER PASSWORD : RAJARSHI
IF U WANT TO SELECT A FILE FROM UR PC AND UPLOAD IT TO AWS USING 3SFS(PRESS1)
IF U WANT TO DOWNLOAD ENCRYPTED FILES FROM AWS AND DECRYPT THEM FOR VIEWING(PRESS 2)
2
ENTER THE NAME OF THE FILE THAT YOU ENCRYPTED : doc3.txt
FILE 1 HAS BEEN SUCCESSFULLY DOWNLOADED FROM AWS
FILE 2 HAS BEEN SUCCESSFULLY DOWNLOADED FROM AWS
FILE 3 HAS BEEN SUCCESSFULLY DOWNLOADED FROM AWS
SUCCESSFULLY DECRYPTED FILE No 1!!!
SUCCESSFULLY DECRYPTED FILE No 2!!!
SUCCESSFULLY DECRYPTED FILE No 3!!!
FILE HAS BEEN SAVED IN UR SPECIFIED DIRECTORY.

Process finished with exit code 0
```

STEP 12 : After getting the name, using the pattern detecting algorithm we retrieve the individual encrypted files, decrypt them individually and finally merge them into the required file.

Decrypted sub-files look like :



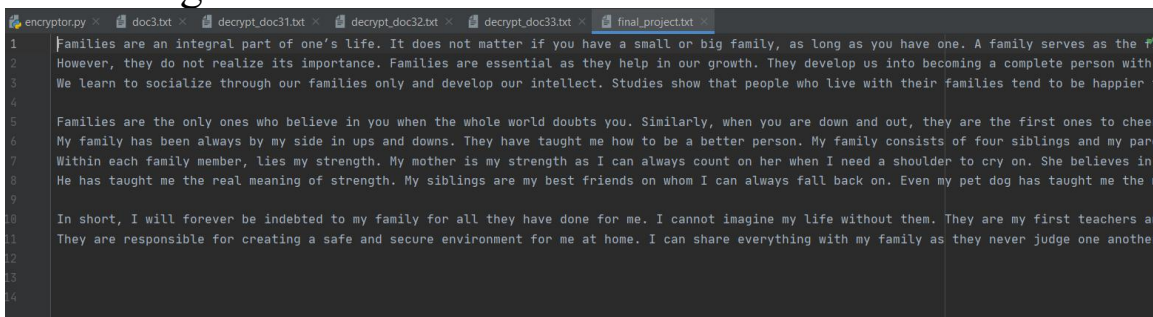
```
encryptor.py x doc3.txt x decrypt_doc31.txt x decrypt_doc32.txt x decrypt_doc33.txt x
Families are the only ones who believe in you when the whole world doubts you. Similarly, when you are down and out, they are the first ones to cheer
My family has been always by my side in ups and downs. They have taught me how to be a better person. My family consists of four siblings and my pare
Within each family member, lies my strength. My mother is my strength as I can always count on her when I need a shoulder to cry on. She believes in
He has taught me the real meaning of strength. My siblings are my best friends on whom I can always fall back on. Even my pet dog has taught me the me

encryptor.py x doc3.txt x decrypt_doc31.txt x decrypt_doc32.txt x decrypt_doc33.txt x
1 Families are an integral part of one's life. It does not matter if you have a small or big family, as long as you have one. A family serves as the fir
2 However, they do not realize its importance. Families are essential as they help in our growth. They develop us into becoming a complete person with
3 We learn to socialize through our families only and develop our intellect. Studies show that people who live with their families tend to be happier th
4

encryptor.py x doc3.txt x decrypt_doc31.txt x decrypt_doc32.txt x decrypt_doc33.txt x
1 In short, I will forever be indebted to my family for all they have done for me. I cannot imagine my life without them. They are my first teachers and
2 They are responsible for creating a safe and secure environment for me at home. I can share everything with my family as they never judge one another.
3
```

STEP 13 : Decrypted File has been Successfully Retrieved From the cloud.

Final merged file looks like :



```
encryptor.py x doc3.txt x decrypt_doc31.txt x decrypt_doc32.txt x decrypt_doc33.txt x final_project.txt x
1 Families are an integral part of one's life. It does not matter if you have a small or big family, as long as you have one. A family serves as the fir
2 However, they do not realize its importance. Families are essential as they help in our growth. They develop us into becoming a complete person with
3 We learn to socialize through our families only and develop our intellect. Studies show that people who live with their families tend to be happier th
4
5 Families are the only ones who believe in you when the whole world doubts you. Similarly, when you are down and out, they are the first ones to cheer
6 My family has been always by my side in ups and downs. They have taught me how to be a better person. My family consists of four siblings and my pare
7 Within each family member, lies my strength. My mother is my strength as I can always count on her when I need a shoulder to cry on. She believes in
8 He has taught me the real meaning of strength. My siblings are my best friends on whom I can always fall back on. Even my pet dog has taught me the m
9
10 In short, I will forever be indebted to my family for all they have done for me. I cannot imagine my life without them. They are my first teachers and
11 They are responsible for creating a safe and secure environment for me at home. I can share everything with my family as they never judge one another
12
13
14
```

Which is the same as The original file:

```
encryptor.py  doc3.txt  decrypt_doc31.txt  decrypt_doc32.txt  decrypt_doc33.txt  final_project.txt
1 Families are an integral part of one's life. It does not matter if you have a small or big family, as long as you have one. A family serves as the f
2 However, they do not realize its importance. Families are essential as they help in our growth. They develop us into becoming a complete person with
3 We learn to socialize through our families only and develop our intellect. Studies show that people who live with their families tend to be happier t
4 Families are the only ones who believe in you when the whole world doubts you. Similarly, when you are down and out, they are the first ones to cheer
5 My family has been always by my side in ups and downs. They have taught me how to be a better person. My family consists of four siblings and my pare
6 Within each family member, lies my strength. My mother is my strength as I can always count on her when I need a shoulder to cry on. She believes in
7 He has taught me the real meaning of strength. My siblings are my best friends on whom I can always fall back on. Even my pet dog has taught me the
8 In short, I will forever be indebted to my family for all they have done for me. I cannot imagine my life without them. They are my first teachers an
9 They are responsible for creating a safe and secure environment for me at home. I can share everything with my family as they never judge one another
10
11
```

Signing Up of an user :

```
C:\Users\Rajarshi\PycharmProjects\Machine\venv\Scripts\python.exe C:/Users/Rajarshi/PycharmProjects/Machine/encryptor.py
HELLO!! WELCOME TO 3 STEP SECURITY FILE STORAGE IN AWS
1. LOGIN
2. SIGN UP
ENTER YOUR CHOICE: 2
ENTER YOUR NAME: ABHIRAM
ENTER YOUR PASSWORD: ABHIRAM
YOUR ACCOUNT IS ALL SET!! PLEASE COPY THE KEY BELOW
b'nj300Y-5on-31X1XHVyEppyesE-DWgAtomWUTQJJczs='
PLEASE PRESS 3 TO PROCEED FURTHER...
3
WELCOME TO FINAL STEP OF SIGN UP PROCEDURE. KINDLY SELECT AN IMAGE AND PASTE UR KEY AS HIDDEN MESSAGE AND ENCODE IT.
Enter image name(with extension) : sugoye.jpg
Enter data to be encoded : nj300Y-5on-31X1XHVyEppyesE-DWgAtomWUTQJJczs=
Enter the name of new image(with extension) : sugoye_key.jpg
YOU ARE ALL SET!!KINDLY EXIT AND RERUN FOR LOGIN BY PRESSING 4
```

	name	password	steg_file_name
▶	RAJARSHI	GGDFR	sugoye.jpg
	RISHAV	RISHAV	france_key.png
	ABHIRAM	ABHIRAM	sugoye_key.jpg

As soon as a user signs up his credentials except his key are stored in MySQL database.

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 Diffie Hellman, Fernet and AES algorithm. Key

information is safely stored using LSB technique (Steganography). Less time is used for the encryption and decryption process using multithreading technique. 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.

REFERENCES:

1.
<https://www.irjet.net/archives/V3/i1/IRJET-V3I1233.pdf>
2.
https://www.researchgate.net/publication/303318872_A_Hybrid_Cryptography_Technique_for_Improving_Network_Security
3.
https://www.researchgate.net/publication/305017745_Role_of_File_System_in_Operating_System
4.
<http://iaetsdjaras.org/gallery/22-march-570.pdf>
5.
<http://www.ijecs.in/index.php/ijecs/article/view/240/199>
6.
<https://irjet.net/archives/V5/i3/IRJET-V5I3475.pdf>
7.
https://www.researchgate.net/publication/334418542_A_Review_Paper_on_Cryptography