



VIT[®]

Vellore Institute of Technology

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

PROJECT REPORT

Submitted by

Rohan Gupta	20BCI0260
Harsh Rajpal	20BCI0271
Divakar Singhal	20BCI0261

Course Title: Security of E-Based Systems

Under the guidance of

Dr. Gopichand G.
Associate Professor, SCOPE,
VIT, Vellore.

SECURING FILE USING HYBRID CRYPTOGRAPHY

1. INTRODUCTION

Cloud storage gives us access to our files anywhere through an internet connection and removes a wide range of hardware malfunctions which is one of the most common causes of data loss.

We have created a web application that allows users to store their data in a cloud securely. The background of the app is coded using Python Edit Language and the front end is enhanced using HTML and CSS.

To improve the model we have used cryptography to provide data security which is the most popular technology currently used for security purposes. To increase the security complexity a combination of three different encryption methods is used, each of which is used to encrypt part of a separate file (simultaneously using detailed readings). Algorithm selection is done using the Round Robin method. In addition to this we also use Digital Signatures.

The app is scalable and can be configured in the future to allow cloud computing to be associated with multiple users at once. This will allow people to work together on the same file - thus greatly improving the efficiency of the work.

2. LITERATURE SURVEY

2.1. EXISTING MODELS

In today's society we are seeing a transition from a mobile site to a cloud and that is why it is so important to ensure security in the cloud. Storing data in the cloud poses many security challenges to common algorithms. It is therefore important to create an algorithm that combines various strategies to ensure security.

[1] Information storage (storing) is one of the most important services provided by cloud computing. The information of information owners is stored on the servers of cloud specialist organisations, and clients can access their information from these servers. The viewpoint of information storage poses multiple security difficulties because information, owners, and servers all have different personalities. To guarantee that information is properly facilitated to the cloud storage server, a free component is necessary. The paper looks at the many options for storing safe data in the cloud.

[2] The major worry in cloud computing is security and privacy. Because of the lack of accuracy in the protection of information sharing, many cloud computing enterprises and customers do not trust cloud providers to keep their sensitive data in public cloud storage. The primary focus of this article is on the security concerns and issues that various cloud systems confront (Education, Enterprise, and Healthcare).

[3] The new worry about cloud storage stems mostly from the fact that most cloud storage service providers now have unfettered access to their clients' private and sensitive data, giving them the ability to read it. There's also the worry that such service providers haven't been able to give adequate assurance and trust that any data or information they keep on their cloud's infrastructure is safe from unauthorized or

unlawful access and change. The major focus of this study is on file security and cloud storage security, with a special emphasis on new trends and mechanisms of hybrid cryptography systems.

[4] When it comes to efficiently look for the best option for storing and retrieving cloud data, blockchain innovation provides significant input. This paper looks at how blockchain technology may be used to safeguard cloud computing. A framework for safe data storage in a cloud computing environment is also presented in this research article. Smart contracts and an access list are used in this architecture to ensure data security.

[5] Everywhere security is a critical problem across a wide range of applications. Many various ways have been proposed to guarantee data protection in the cloud, however existing systems frequently fail when only one type of encoding is used, The main problem with this approach is that each encryption is done using encryption keys, and if these keys are leaked in any way, the entire data is lost, thus we need a solution with more security. As a consequence, this research employs hybrid cryptography, in which existing encryption techniques are merged with three novel ways.

[6] The paper explores the literature on data security and privacy problems, data encryption technologies, and related countermeasures in cloud storage systems in this study. First, the paper goes through the basics of cloud storage, including definitions, classifications, architecture, and applications. Second, it goes into the issues and needs of data security and privacy protection in cloud storage systems in great depth. Finally, a summary of data encryption technology and protection approaches is provided. Finally, it goes through a few open data security research areas for cloud storage.

[7] Cloud users become victims of the security vulnerabilities revealed by cloud storage owing to a lack of understanding and sufficient security measures. To prevent security risks, both the cloud user and the Cloud Storage Provider (CSP) should utilise suitable approaches and follow best practises. The security weaknesses in cloud storage are examined in this study, as well as contemporary state-of-the-art solutions for enhancing its security. The best practises and security policies developed by various organisations are discussed further.

[8] In this document, a hybrid cryptographic protocol that delivers Blowfish and Paillier encryption algorithms has been introduced and its capabilities compared to the existing hybrid Advanced Encryption Standard (AES) and Rivest Shamir Adleman (RSA) methods. Algorithms for secure data storage protocols in two categories are introduced. The proposed hybrid protocol seeks to improve cloud storage capacity by reducing computation time and ciphertext size.

[9] Even with stronger network security mechanisms and data protection algorithms, information privacy system service providers cannot guarantee a 100% secure solution. Let's forget the network transfer and other aspects of security; at the end of the day, the file lives at specific storage with corporate ownership, which may be referred to as a storage service provider, and company employees with the greater authority having constant access to the customer's data. The proposed solution addresses malicious user attack security and prevents file access by storing files in multiple cloud environments.

[10] When data is exchanged through the internet, the security of the data is compromised. The most important data mechanisms are integrity, accountability, privacy, and access are all examples of safeguards. Blockchain is a technology that allows improved cloud computing. Blockchain overcomes the Cloud computing security problems. This survey's goal is assessing and contrasting various cloud problems using blockchain to solve environmental and security concerns.

[11] From 2015 until early 2019, a research of hybrid cryptography was conducted in this publication. In this study, papers relating to the topic were searched, and roughly 20 were chosen based on filtering. Eight (8) of them are based on a user-friendly tabular survey, while the other twelve are in-depth questionnaires. The major goal of this review study is to assist new researchers, students in this subject, and novices in cryptography with more and more knowledge. The research gap found is the failure to consider user authentication and the deployment of hybrid algorithms incorrectly.

2.2. PROBLEM STATEMENT

With the growing need for data security, we need a model that incorporates encryption using computer-generated non-compliant algorithms. Once encryption is complete using these algorithms, a path must be set to assign a key after encryption, in order for the encoding process to take place. In line with this, the model must adhere to the basic principles of modern cryptography, namely Confidentiality, Integrity and Authentication.

3. OVERVIEW OF THE PROJECT

3.1. OBJECTIVE

Our goal is to build a cloud-protected file storage system that satisfies all the basic principles of cryptography namely Confidentiality, Integrity and Verification without compromising the speed of an advanced application. We have used Hybrid cryptography (using 3 encryption algorithms: AES, Fernet, ChaChaPoly1305), Round Robin Algorithm and Digital Signatures to achieve this goal.

3.2. SOFTWARE REQUIREMENTS

Software and APIs: The program must have Python 2.7 and its Flask API installed

Operating system: Can work on most popular applications such as Microsoft Windows, Linux, and Apple macOS.

3.3. HARDWARE REQUIREMENTS

The minimum requirements for three different parameters are listed below:

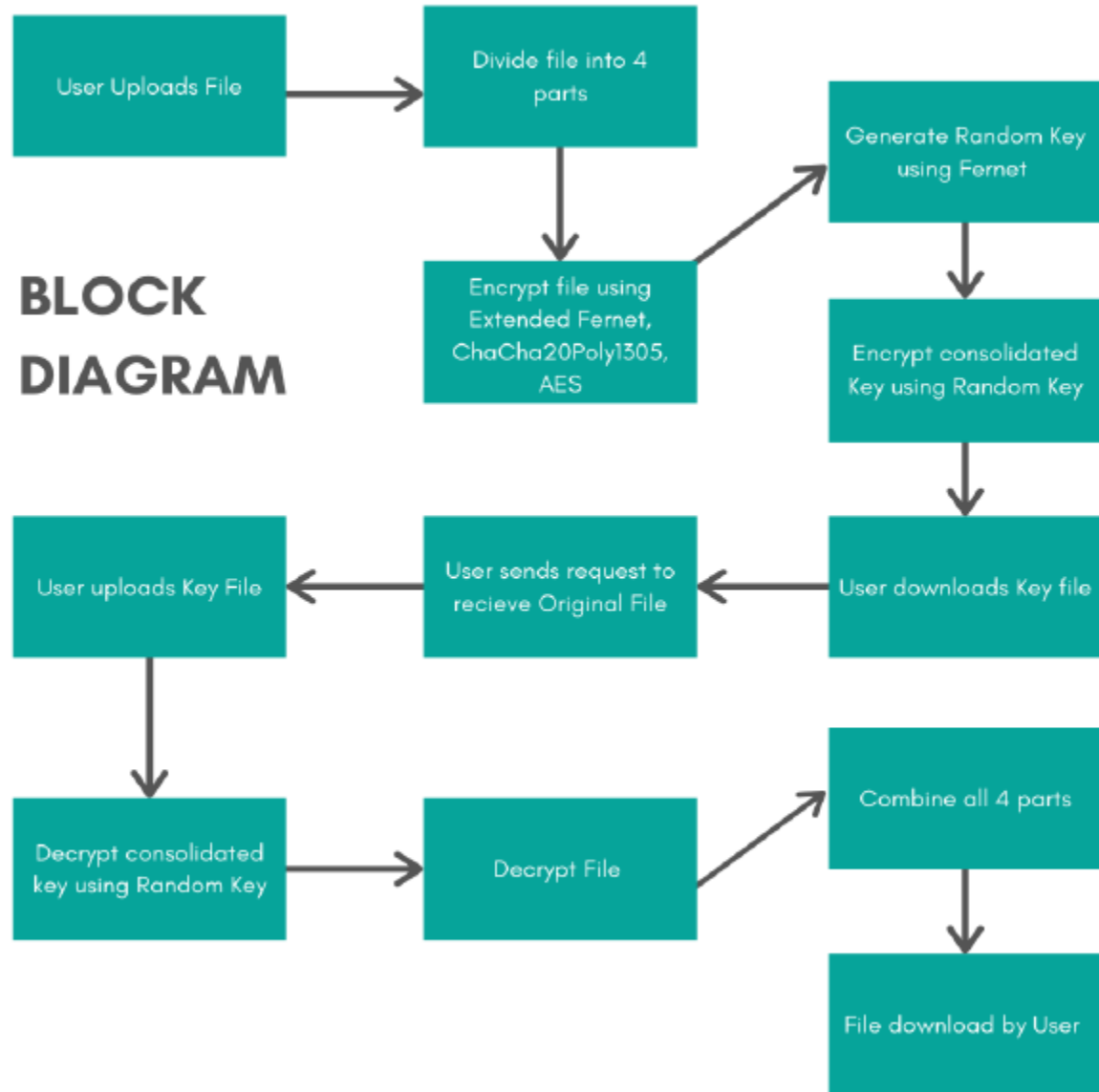
Memory: 512 MB

Disk space: 300 MB

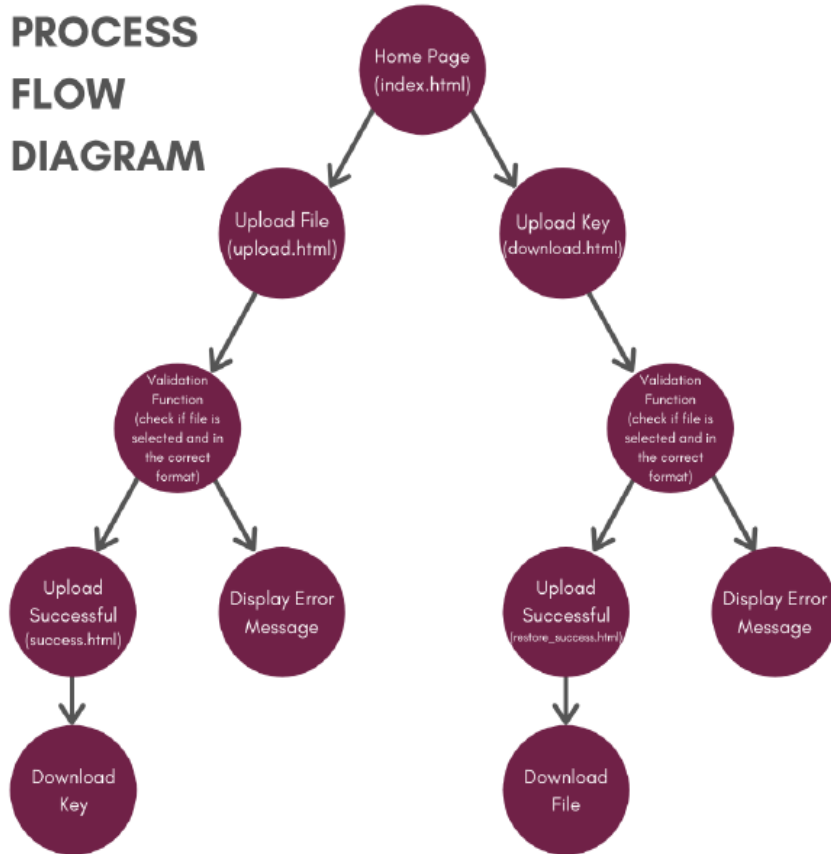
Minor processor speed: 800 MHz

4. SYSTEM DESIGN

4.1. BLOCK DIAGRAM



4.2. PROCESS FLOW DIAGRAM



4.3. METHODOLOGY

Steps involved in saving the file:

1. The file is uploaded to the server after the user uploaded it.
2. The file is divided into N sections.
3. All parts of the file are encrypted using any selected algorithms (For each component, the algorithm is rotated in a robin circular manner).
4. The keys to the cryptography algorithms are then protected using a different algorithm. The user is provided with a public key for this algorithm. After completing the above 4 steps, N files will be stored on the server in their own encrypted way and the user has a download key as a public key that can be used to decrypt N-components and download the original file after re-deleting N deleted. parts.

Steps involved in file retrieval:

1. Key uploaded to server.
2. The keys to all algorithms have been removed.
3. All N sections of the file are encrypted using the same algorithms used for encryption.
4. Now, all the N components are assembled to form the first file and the user is given the first file to download.

5. IMPLEMENTATION

5.1. DESCRIPTION OF MODULES / PROGRAMS

1. Module os: This python module gives us applications to work with the application.
2. Module battery: This module allows secure data storage. This module provides encryption functions.
3. Module extractor: This module provides encryption functions.
4. Module sys: This python module provides a variety of functions and variables that are used to manage different parts of the python runtime environment.
5. Crypto Package: This package contains algorithms for protecting the privacy of data.
6. ChaCha20: ChaCha20 is a streaming cipher and the Poly1305 is authentic. They are cryptographic algorithms designed to achieve high security, while maintaining good performance. Faster page rendering and better battery life are achieved by spending less time encrypting. The ChaCha20 is designed to provide 256-bit level protection. The design of Poly1305 is such that it verifies that spam messages are rejected with $1 - (n / 2^{102}) 16 * n$ byte messages, even after 2^{64} official messages have been sent. No 20- Poly1305 is a certified cipher with related data (AEAD). It works with a 32 bytes secret key and nonce that should not be reused on all encryption performed under the same key. A 16 byte mark is produced by this cipher. The recipient must use this tag to verify the message.
7. Fernet: The fernet module cryptography package has built-in functions for encrypting, encrypting text into encrypted text, and cipher encoding into abstract encryption using encryption and sequence encryption methods. The fernet module ensures that encrypted data can not be further manipulated or read without keys.

Methods used:

- generate_key (): This method generates a new fernet key. The key should be kept secure as it is the most important part of removing cipher text encryption. If the key is lost the user will no longer be able to encrypt the message. And when a criminal or hacker gets access to a key they can not only read the data but also create the data.
 - encrypt (data): Encrypts data transmitted as a parameter along the path. The result of this crucifixion is known as the "Fernet token" which is the cipher inscription. The encrypted token also contains the current stamp when it is generated in plain text. The encryption method does the opposite when data is not available in bytes.
 - decrypt(token, ttl = None): This method removes the encryption of the Fernet token transmitted as a parameter along the way. In deleting successful encryption the actual clear text is available as a result, without which exceptions is thrown.
8. Advanced Writing Standards (AES):
 - AES cipher is a block cipher. In block ciphers there are algorithms in which data is encrypted on the basis of each block.

- The AES algorithm is related to Rijndael encryption which is a family of encryption algorithms with different key sizes and block sizes.
- It has continuous serial functions, some of which include the output of certain output (modified) and other fragmentation (permissions).
- All calculations in the AES algorithm are done by bits instead of bytes. In AES, 128 bits of empty data are considered a block of 16 bytes.
- For analysis, these 16 bytes are arranged in a 4x4 matrix.
- The AES algorithm is divided into 3 types, AES-128bit, AES-192bit, and AES-256bit. For each duplicate data encrypted and cleared encryption blocks using 128-bits or 192-bits or 256-bits keys.

9. AES with Galois / Counter Mode (AES GCM):

AES-GCM is implemented using 128-bit input data block, Startup Vector (IV) and Authenticated Extra Data (AAD) to provide maximum encryption / output encryption. AES-GCM has a key length of 256-bit.

10. Digital Signature:

A digital signature is used to provide integrity. This uses the pattern of each saved file to determine its conversion. Cryptographic hash functions are methods used to produce a pattern. The names of all the files to be protected and their hash codes are stored on the website. File integrity can be tested by generating its own hash code and comparing it with existing ones on the website. Access is granted only after the file has been verified. Further notice of problems is sent to the administrator and a saved copy of the file can be returned.

5.2. Code

1. Encrypt.py

```

1 from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
2 from cryptography.hazmat.backends import default_backend
3 from cryptography.hazmat.primitives import hashes
4 import os
5
6 def AES(key, iv):
7     f = open(os.path.join(os.getcwd(), "0.txt"), "r")
8     content = f.read()
9     f.close()
10    content = content.encode()
11    b = len(content)
12    while(b % 16 != 0):
13        content += "\0"
14        b = len(content)
15    backend = default_backend()
16    cipher = Cipher(algorithms.AES(key), modes.CBC(iv), backend=backend)
17    encryptor = cipher.encryptor()
18    cont = encryptor.update(content) + encryptor.finalize()
19    open(os.path.join(os.getcwd(), "0.txt"), "w").close()
20    f = open(os.path.join(os.getcwd(), "0.txt"), "wb")
21    f.write(cont)
22    f.close()
23
24 def Blowfish(key, iv):
25     f = open(os.path.join(os.getcwd(), "1.txt"), "r")
26     content = f.read()
27     f.close()
28     content = content.encode()
29     b = len(content)
30     while(b % 16 != 0):
31         content += "\0"
32         b = len(content)
33     backend = default_backend()
34     cipher = Cipher(algorithms.Blowfish(key), modes.CBC(iv), backend=backend)
35     encryptor = cipher.encryptor()
36     cont = encryptor.update(content) + encryptor.finalize()
37     open(os.path.join(os.getcwd(), "1.txt"), "w").close()
38     f = open(os.path.join(os.getcwd(), "1.txt"), "wb")
39     f.write(cont)
40     f.close()

```

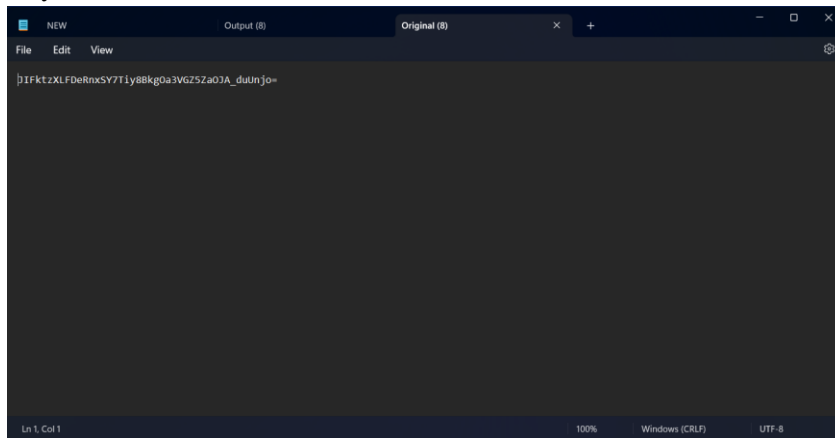

2. Decrypt.py

```
Decrypt.py > DAES
1 from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
2 from cryptography.hazmat.backends import default_backend
3 from cryptography.fernet import Fernet
4 import os
5
6
7 def DAES(key,iv):
8     f=open(os.path.join(os.getcwd()+"/Segments","0.txt"),"rb")
9     content=f.read()
10    f.close()
11    backend = default_backend()
12    cipher = Cipher(algorithms.AES(key), modes.CBC(iv), backend=backend)
13    decryptor = cipher.decryptor()
14    content=decryptor.update(content) + decryptor.finalize()
15    f=open(os.path.join(os.getcwd()+"/Segments","0.txt"),"wb")
16    f.write(content)
17    f.close()
18
19 def DBlowfish(key,iv):
20     f=open(os.path.join(os.getcwd()+"/Segments","1.txt"),"rb")
21     content=f.read()
22     f.close()
23     backend = default_backend()
24     cipher = Cipher(algorithms.Blowfish(key), modes.CBC(iv), backend=backend)
25     decryptor = cipher.decryptor()
26     content=decryptor.update(content) + decryptor.finalize()
27     f=open(os.path.join(os.getcwd()+"/Segments","1.txt"),"wb")
28     f.write(content)
29     f.close()
30
31 def DTripbleDES(key,iv):
32     f=open(os.path.join(os.getcwd()+"/Segments","2.txt"),"rb")
33     content=f.read()
34     f.close()
35     backend = default_backend()
36     cipher = Cipher(algorithms.TripbleDES(key), modes.CBC(iv), backend=backend)
37     decryptor = cipher.decryptor()
38     content=decryptor.update(content) + decryptor.finalize()
```

3. App.py

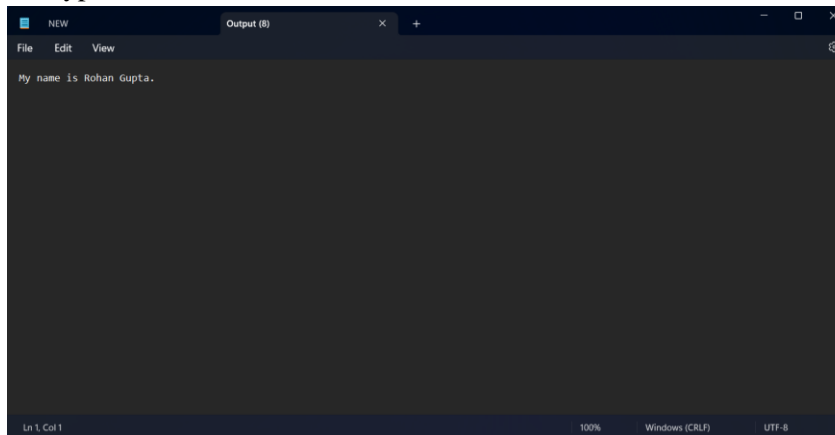
```
app.py > DecryptMessage
1 import os
2 from flask import Flask, request, redirect, url_for, render_template, send_from_directory, flash
3 from werkzeug.utils import secure_filename
4 from dataProcessing import *
5 from Threads import *
6 from flask import send_file
7 import time
8 import os
9 script = ''
10
11 UPLOAD_FOLDER = '.'
12 ALLOWED_EXTENSIONS = set(['txt'])
13
14 # api = API(app)
15 app = Flask(__name__)
16 app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
17 app.config['CACHE_TYPE'] = "null"
18
19 def resultI():
20     path = "/Segments"
21     dir_list = os.listdir(path)
22     print(dir_list)
23     return render_template('Result.html',dir_list = dir_list)
24
25 def resultD():
26     return render_template('resultD.html')
27
28 @app.route('/encrypt/')
29 def EncryptInput():
30     Segment()
31     gatherInfo()
32     HybridCrypt()
33     return resultI()
34
35 @app.route('/decrypt/')
36 def DecryptMessage():
37     st=time.time()
38     HybridDecrypt()
```

4. Key stored in cache



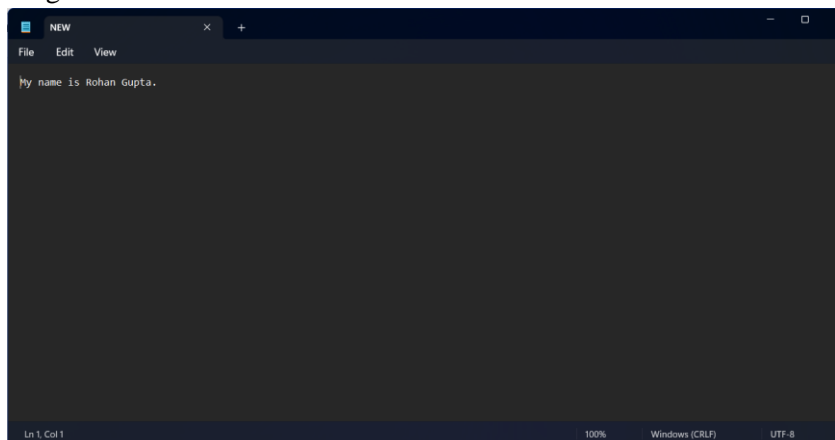
A screenshot of a text editor window with a dark theme. The window has a title bar with 'NEW' and a tab labeled 'Original (8)'. The menu bar includes 'File', 'Edit', and 'View'. The text area contains a single line of a long alphanumeric string: `hIfktzXLFDeRnxSY7Tly8Bkg0a3VGZ5Za0JA_dutInJo=`. The status bar at the bottom shows 'Ln 1, Col 1', '100%', 'Windows (CRLF)', and 'UTF-8'.

5. Decrypted Text



A screenshot of a text editor window with a dark theme. The window has a title bar with 'NEW' and a tab labeled 'Output (8)'. The menu bar includes 'File', 'Edit', and 'View'. The text area contains a single line of text: `My name is Rohan Gupta.`. The status bar at the bottom shows 'Ln 1, Col 1', '100%', 'Windows (CRLF)', and 'UTF-8'.

6. Original Text



A screenshot of a text editor window with a dark theme. The window has a title bar with 'NEW' and a tab labeled 'NEW'. The menu bar includes 'File', 'Edit', and 'View'. The text area contains a single line of text: `My name is Rohan Gupta.`. The status bar at the bottom shows 'Ln 1, Col 1', '100%', 'Windows (CRLF)', and 'UTF-8'.

7. Final.html

```
1 <!DOCTYPE html>
2 <html>
3 <title>Secure File Storage</title>
4 <head>
5 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-e039d37l1isc0/b36fsc0c"
6 </head>
7
8 <style>
9 body{
10 background: url('../static/img/successful_encrypt.png');
11 color: #28a745;
12 background-size: cover;
13 background-repeat: no-repeat;
14 }
15
16 .btn {
17 border: none;
18 border-radius: 0;
19 width: auto;
20 box-sizing: border-box;
21 padding: 2px 10px;
22 transition: all 0.6s;
23 color: #ffffff;
24 font-size: 15px;
25 vertical-align: middle;
26 text-transform: uppercase;
27 margin-left: 10px;
28 }
29
30 .btn:hover {
31 box-shadow: 0 0 5px #28a745;
32 }
33 </style>
34 <body>
35 <center> <div style="margin-top: 30px;"> Thank You For Using Our Service!!! Your File Is Saved Successfully: </div> <div>
36 <a href="/return-files-key/" target="blank"><button class="btn btn-danger" style="color: black;">Download Key</button></a>
37 <a href="/decrypt/" target="blank"><button class="btn btn-danger" style="color: black;">Decrypt</button></a>
38 </div>
39 </body>
40 </html>
```

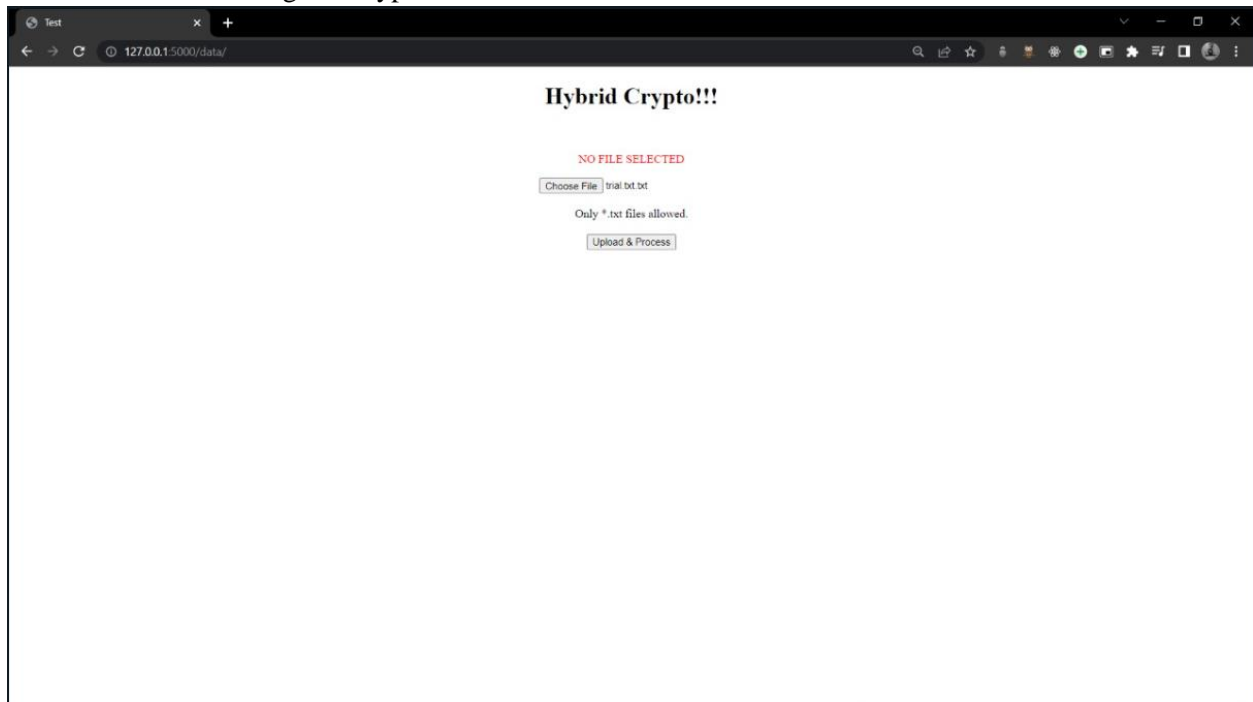
6. RESULTS AND ANALYSIS

Here, we will be explaining a little about how our code works and what basically is happening. Our home page looks like this:



So, first of all, we upload our file which we want to encrypt by on the button of upload file. The button can be seen in the above image.

After clicking on the upload file button, you will be redirected to a localhost where the person will upload the file he/she needs to get encrypted.



After that, the user just has to click on the encryption button.



The user will be directed to a page where he will be able to download the key to decrypt his encrypted file. Also, the user can decrypt any encrypted file by uploading the key file and then can download the original file.



7. CONCLUSION

We have successfully implemented a very secure and robust file storage system by the use of symmetric key cryptography techniques. Files uploaded by the users are stored on the cloud server and with this, unauthorized access to the outside world is avoided. Our model not only provides confidentiality but also integrity and authentication.

8. REFERENCES

- [1] Devang Pratap Singh, Prakarsh Kaushik, Manjari Jain, "Data Storage Security Issues in Cloud Computing", 2021 International Conference on Innovative Practices in Technology and Management (ICIPTM), 2021 IEEE, January 2022.
- [2] G. Nagarajan, Dr. K. Sampath Kumar, "Security Threats and Challenges in Public Cloud Storage", 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), 2021 IEEE, June 2021.
- [3] Aman Singh, Shivashankar Reddy Ginni, Dr. Advin Manhar, "Securing File Storage on the Cloud using Cryptography", International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), Vol. 10, Issue 4, April 2021.
- [4] Praveen Kumar Kollu, Monika Saxena, Khongdet Phasinam, Thanwamas Kassaruk, Malik Mustafa, "Blockchain Techniques for Secure Storage of Data in Cloud Environment", Turkish Journal of Computer and Mathematics Education, Vol.12 No. 11(2021), 1515-1522, May 2021.
- [5] P. Bharathi, G. Annam, J. B. Kandi, V. K. Duggana and A. T., "Secure File Storage using Hybrid Cryptography," 2021 6th International Conference on Communication and Electronics Systems (ICCES), 2021, pp. 1-6.
- [6] Pan Yang, Neal N. Xiong, Jigli Ren, "Data Security and Privacy Protection for Cloud Storage: A Survey", IEEE

Access, vol 4, pp(99) : 1-1, July 2020.

[7] Bharati Mishra, Debsish Jena, "Security of Cloud Storage: A Survey", 2019 International Conference on Information Technology (ICIT), 2019 IEEE, March 2020.

[8] Bijeta Seth, Surjeet Dalal, Da-Nhuong Le, Vivek Jaglan, Neeraj Dahiya, Akshat Agrawal, Mayank Mohan Sharma, Deo Prakash, K.D. Verma, "Secure Cloud Data Storage System Using Hybrid Paillier-Blowfish Algorithm", Computers, Materials and Continua (CMC), Vol. 67, no. 1, November 2020.

[9] Manoj V. Brahme, Dr. Milind V. Sarode, Dr. Meenakshi S. Arya, "Design and Implementation of Secure File Storage using Distributed Cloud Mechanism", International Journal of Research and Analytical Reviews (IJRAR), Vol. 6, Issue 1, February 2019.

[10] S. Pavithra, S. Ramya, Soma Prathibha, "A Survey On Cloud Security Issues and Blockchain", 3rd International Conference on Computing and Communication Technologies (ICCCCT), 2019 IEEE, 136-140, 2019.

[11] S. A. Ahmad and A. B. Garko, "Hybrid Cryptography Algorithms in Cloud Computing: A Review," 2019 15th International Conference on Electronics, Computer and Computation (ICECCO), 2019, pp. 1-6.