###### **CRYPTOGRAPHIC PROTECTION WITH NOISE IMAGES AS UNIQUE CIPHER KEYS**

*A Project report submitted in partial fulﬁllment of the requirements*

*For the award of the Degree of*

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

By

Anusha.M (319136410081) Tanu (319136410116)

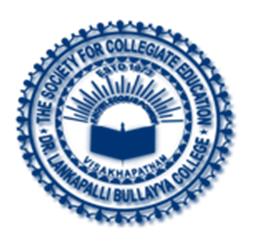
Purvika.K (319136410097) Sai Sambhavi.A(319136410126)

Under the esteemed guidance of

Sri. Syed Mujib Rahman

Associate Professor

Department of Computer Science Engineering



#### **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

#### **Dr. L. BULLAYYA COLLEGE OF ENGINEERING**

(Affiliated to Andhra University,Visakhapatnam)

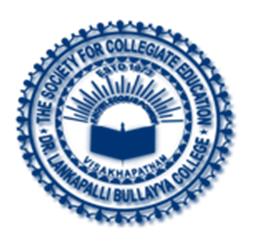
New Resapuvanipalem, Visakhapatnam-530013

Year of Submission 2023

**DR.L.BULLAYYA COLLEGE OF ENGINEERING**

New Resapuvanipalem, Visakhapatnam-530013

**Department of Computer Science Engineering**



**BONAFIDE CERTIFICATE**

This is to certify that Ms.Anusha, Tanu, Purvika, Attadi Sai Sambhavi bearing register numbers    319136410081, 319136410116, 31913640097, 319136410126, students of 4th year B. Tech in Computer Science Engineering, has carried out the project work titled “FILE ENCRYPTION USING NOISE IMAGE AS KEY” at Dr. L.Bullayya College of Engineering, Visakhapatnam during the academic year 2021-22.

**Project Supervisor**    **Head Of The Department**

 Prof. Syed Mujib Rahaman                                                        Dr. D Madhavi

Professor                                           Professor

 Dept. of Computer Science Engineering                                    Dept. of Computer Science Engineering

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **TITLE** | **PAGE NUMBERS** |
| Abstract | 05 |
| Acknowledgements | 06 |
| Declaration | 07 |
| List of Tables | 08 |
| List of Figures | 09 |
| List of Symbols | 10 |
| **1: INTRODUCTION** | 11-12 |
| 1.1Motivation | 13 |
| 1.2 Problem Statement | 13 |
| **2:REQUIREMENTS ELICITATIION AND ANALYSIS** | 14 |
| 2.1 Existing system | 14 |
| 2.2 Proposed system | 14 |
| 2.3 Feasibility Study | 14 |
| 2.4 System Requirements | 15 |
| 2.4.1 Functional requirements | 15-16 |
| 2.4.2Non-functional Requirements | 17 |
| **3: SYSTEM DESIGN** | 18 |
| 3.1 Object oriented Analysis and Design | 18 |
| 3.1.1 Scenarios | 19 |
| 3.1.2 Use case diagram | 20 |
| 3.1.3 Activity diagram | 21 |
| 3.1.4 Class Diagram  3.1.5 Sequence Diagram | 22  23 |
| **4: IMPLEMENTATION DETAILS** | 24 |
| 4.1 Software Environment | 24 |
| 4.2 Software Technologies | 25-28 |
| **5: TESTING** | 29-31 |
| **6: CONCLUSION** | 32 |
| REFERENCES | 33-34 |
| APPENDIX | 35 |
| A- Input/Output Screens | 35-39 |
| B- Sample Code | 40-42 |
|  |  |

### 

### **ABSTRACT**

### 

Everything in today's highly connected digital world is increasingly dependent on instantaneous global data transfers. Internet efficiency facilitates our daily lives. Sharing information online presents significant security dangers and difficulties in the modern day. The use of cryptography is the means through which sensitive information can be protected from various threats. Using AES algorithm the project presents a method of data encryption and decryption that guarantees increased efficiency. In this project, a noise image encryption and decryption scheme are proposed in which the noise signal is selected randomly to set the initial values for a chaotic system which also enhances the security of the system,for understanding the effectiveness of the proposed system. Experimental results confirm that the proposed chaos based cryptosystem is efficient and suitable for information (image) transmission in a highly secured way. We have created a file encryption system where we save encrypted file and decrypt and download it later. The main difference between the existing and proposed system is the file encryption system is implemented using noise images that stores key values inside it.

.

## Keywords: Encrypting, Key, Decryption, Noise picture, Cryptographic

# **ACKNOWLEDGEMENT**

We would like to express our heartfelt gratitude to Prof D.Deepak Chowdary, Principal, Dr. L. Bullayya College of Engineering for giving us the opportunity to do this project.

We would also like to thank Dr. D. Madhavi, Head of the Department of CSE, Dr. L. Bullayya College of Engineering, Visakhapatnam, for her guidance in producing this work.

We are deeply indebted to our project guide and the head of research cluster Prof. Syed Mujib Rahaman, department of Computer Science Engineering, Dr. Lankapalli. Bullayya College of Engineering, Visakhapatnam, for guiding us throughout the project in spite of his busy schedule.

Apart from our efforts, the success of this project depends largely on the encouragement of other faculty members of CSE, Dr. Lankapalli. Bullayya College of Engineering, Visakhapatnam. We take this opportunity to express our gratitude to the entire faculty who has been instrumental in the successful completion of this project.

Also deserving of thanks for our family and friends, for their support and for their confidence in our achievements.

Anusha.M (319136410081)

Purvika.K (319136410097)

Sai Sambhavi.A (319136410126)

Tanu (319136410116)

**DECLARATION**

This is to declare that the Project work entitled “CRYPTOGRAPHIC PROTECTION WITH NOISE IMAGES AS UNIQUE CIPHER KEYS” is a bonafide work done by us under the research cluster group “Privacy Preserving and Network security Data Science” with the esteemed guidance of Prof. Syed Mujib Rahaman, Department of CSE, Dr. Bullayya College of Engineering. This project report is being submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science Engineering during the academic year 2022-2023. This project possesses originality as it is not extracted from any source and it has not been submitted to any other institutions and university.

|  |  |
| --- | --- |
|  |  |
|  |  |
| Anusha (319136410081)  Purvika.K (319136410097)  Sai Sambhavi.A (319136410126)  Tanu (319136410116) |  |
|  |  |

### Place: Visakhapatnam

### Date:

### 

### 

# 

# **LIST OF TABLES**

|  |  |
| --- | --- |
| TABLE | CONTENTS |
| Table 1 | System requirement specifications |
| Table 2 | Test Case |

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# **LIST OF FIGURES**

|  |  |
| --- | --- |
| FIGURE NO. | TITLE |
| 1 | Use case diagram |
| 2 | Activity diagram |
| 3 | Class diagram |
| 4 | Sequence diagram |
| 5 | outputs |

# 

# 

# **LIST OF SYMBOLS**

|  |  |
| --- | --- |
| SYMBOL | MEANING |
|  | Class- represents collection of similar entities grouped together |
|  | Aggregation- it aggregates several classes into single class |
|  | Actor- user of the system that reacts to the system |
|  | Use case- it is an interaction between the system and external environment |
|  | communication- it is communication between use cases |
|  | control flow- it represents the decision making process for objects |
|  | decision box- it represents the decision making process from a constraint |
|  | state- represents the state of process, each state goes through various flows. |

**CHAPTER 1: INTRODUCTION**

Every day, a corporation handles files, so it needs reliable data security methods. 32% of organizations are affected by cybercrime, which costs $6 trillion globally. You need sophisticated security capabilities that interact nicely with your existing platforms and fit within your company budget to mitigate this risk. One of the best security measures is file encryption. With advanced security measures, it protects your business's data. Encrypting a file involves encoding not only the file itself but also any sensitive information that it may contain. This allows the file to be sent via an insecure channel without compromising its integrity. Because of the encoding, nefarious actors are unable to get unauthorized access to the data or modify it in any way. It prevents a file from being accessed by anyone other than the person or individuals for whom it was intended, whether it be one person or multiple people. Encryption of files is a useful solution to this problem. It gives you the ability to make any sensitive information you save unreadable to anybody other than the authorized recipients.

The encryption of files is accomplished by the application of complicated algorithms. A file is considered to be encrypted if an encryption technique has been applied to it to jumble the data contained within the file. After the file has been scrambled, it will no longer be readable; however, this will only be a temporary impact. The sender is responsible for providing the decryption key for the data that has been encrypted. This key, which enables decryption, takes the form of a password or passphrase most of the time. For example, it might be a string of alphanumeric numbers. The decryption key is only given to those people who are legally permitted to access the data. The file will become readable once again after the authorized receiver has entered the appropriate password or passphrase. Most computer operating systems and file systems are equipped with integrated support for file encryption. The system encrypts and stores sensitive data in a secure location, and the decryption key is required to access those files.

Keys are typically required to complete the encryption process. Cryptography is the practice of protecting sensitive information by scrambling it and then reassembling it using secret keys. Symmetric and asymmetric cryptography are the two primary forms that it can take. The use of asymmetric cryptography, sometimes known as public-key cryptography, is commonplace in many businesses. These encryption methods make use of a public key in addition to a private key. Anyone who possesses the public key can utilize it to encrypt files. However, only users who possess the private key can decrypt the data, therefore the files will continue to be protected from access by unauthorized users. The process of encrypting and decrypting data with symmetric encryption only requires the usage of a single private key. To be able to decrypt the encoded files, the two parties that are exchanging information through symmetric encryption need to exchange keys.

In most cases, symmetric encryption can be made to function more quickly and effectively than asymmetric encryption. Whenever an organization has to encrypt a large amount of information, such as an entire database, they typically turn to this method. However, symmetric encryption has the drawback of having a lower level of overall security efficacy. This word indicates that it will likely be more difficult to maintain the secrecy of the key. For instance, if the process of encrypting and decrypting data takes place in various locations, the private key will need to be moved between those sites, which leaves it open to the possibility of being attacked. Programs that are written to encode the data in a variety of predetermined ways are typically used to carry out encryption. Certain sectors have adopted particular encryption standards as their norm. Others are compatible with a limited number of different kinds of databases. The efficiency of the encryption is determined by several distinct aspects, including the appropriateness of the encryption system, the efficacy of the algorithm, and the magnitude of the key. In this article, we are going to discuss the process of encrypting data using noise images as the key, and we are also going to have a look at various cases. Experiments have indicated that the noise picture that is proposed to be used as a key for encryption and decryption methods is safe and reliable, suggesting that it may have the potential to be adapted to high-security image communication applications.

* 1. **Motivation**

File encryption using a noise image as a key can be motivated by a few factors:

Security: The encrypted file is further fortified by the use of a noise picture as the encryption key. Attackers will find it challenging to predict or brute-force the picture key because it can be generated arbitrarily. In the event of a data breach, the image key cannot be easily cracked because it is not a basic text password.

Simpler to use than conventional encryption techniques that rely on text passwords is the use of a noise picture as a key. Users will find it simpler to remember an image key than a lengthy text password, especially since it is simpler to store the image in a secure location.

More significant encryption key: Because a noisy picture may hold more data than a typical text password, the encryption key can be larger, which helps increase the security of the encrypted file.

## 

## **1.2. Problem Statement**

File encryption is the process of converting plaintext data into a form that is unreadable without a key. One method of file encryption that involves using a noise image as a key is to convert the plaintext data into a cipher text using a symmetric key algorithm. The symmetric key is then encrypted using the noise image as a key, resulting in a secure encrypted file.

**CHAPTER 2: REQUIREMENTS ELICITATION AND ANALYSIS**

**2.1 Existing System**

The existing systems encryption and decryption methodologies are not efficient existing system is not done with noise key as an encryption method. The existing system also doesn’t have a scalable architecture and it can’t be used for complex use cases.

**2.2 Proposed System**

Step 1: Initializing Backend The first step is to create a Django backend to work with the file encryption system.

Step 2: Creating algorithm: We will use AES algorithm for doing the encryption –decryption.

Step 3: Make a table for the database using: To manage the users and files that are saved on the server, we will develop a user table in addition to a file handling table.

Step 4: Creating Authentication System :By utilizing a distinct Django server, we will construct an authentication system in order to guarantee the safety of our users. In most cases, the application level is where the encryption of fields at the field level has to take place. This is accomplished by coupling the data with a top-secret key that is familiar only to the application server in question.

**2.3 Feasibility System**

Encrypting a file using a noise image as a key is a possible approach to ensure the security of the file. The idea is to use the noise image as a one-time pad, which is a cryptographic technique that uses a random key to encrypt a message. The key is used only once and then destroyed, making it impossible for an attacker to decrypt the message without the key.

The security of this approach depends on the randomness and secrecy of the noise image. The noise image must be truly random and not reused for any other encryption decryption. It must also be kept secret, as an attacker who knows the noise image can easily decrypt the file.

Overall, using a noise image as a key for file encryption is a feasible approach to ensure the security of a file, but it requires careful attention to key management and may have practical limitations depending on the size of the file and the availability of truly random noise.

## 

## **2.4 Functional Requirements**

1. Encryption: The system must be able to encrypt files of various types using the noise image as a key. The encryption should be strong to prevent unauthorized access to the files.
2. Key Generation: The system should be able to generate a random noise image that can be used as a key for encryption. The noise image should be unique for each encryption decryption.
3. Decryption: The system should be able to decrypt encrypted files using the noise image as a key.
4. Key Management: The system should have a mechanism to securely store and manage the noise image key.
5. User interface: The system should provide a user-friendly interface that allows users to select files to be encrypted, select the noise image to be used as a key and initiate the encryption process. The system should also provide a way to decrypt encrypted files.
6. Performance: The system should be able to encrypt and decrypt files quickly and efficiently, without affecting the performance of the device.
7. Error Handling: The system should be able to handle errors and provide meaningful error messages to the user in case of any issues.
8. Backup and recovery: The system should provide a way to back up and recover the encrypted files and the key.

## **2.4.1 Software Requirement specifications**

## 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.NO | REQUIREMENTS | REQUIREMENT NO. | ESSENTIAL/DESIRABLE | DESCRIPTION |
| 1. | Register | RS1 | Essential | Password should contain special characters and numbers |
| 2. | Login | RS2 | Essential | Credentials are required |
| 3. | File uploading | RS3 | Essential | Upload a file for encryption |
| 4. | Encryption | RS4 | Essential | The file will be converted into bytecode |
| 5. | Key generation | RS5 | Essential | A noise image(key) will be generated |
| 6. | Decryption | RS6 | Essential | Using a noise image to decrypt the file |

## 

**2.5 Non-Functional requirements**

1. Usability: The encryption process should be user-friendly and easy to use. It should not require a high level of technical knowledge.
2. Scalability: The encryption system should be scalable enough to handle the increasing number of files and users without reducing its performance.
3. Reliability: The encryption algorithm should be reliable and consistent. It should be able to encrypt and decrypt files accurately without any data loss or corruption
4. Maintainability: The encryption system should be easy to maintain and update. Any security vulnerabilities discovered should be quickly patched, and the encryption algorithm should be updated regularly to keep it secure against new threats.
5. Legal compliance: The encryption system should comply with relevant legal and regulatory requirements, such as data privacy laws and data protection regulations.
6. Interoperability: The encryption system should be able to integrate with other security systems and tools, such as firewalls, antivirus software, and intrusion detection systems.
7. Audibility: The encryption system should provide logs and audit trails to track all encryption and decryption activities for accountability and forensic purposes.

**Hardware Requirements:**

• Microsoft Server-enabled computers, preferably workstations

• Higher RAM, of about 4GB or above

• Processor of frequency 1.5GHz or above

**Software Requirements:**

• Python 3.6 and higher

•Anaconda software

**CHAPTER 3: SYSTEM DESIGN**

**3.1 Object Oriented Analysis and Design**

Object-oriented analysis and design (OOAD) is a methodology for developing software systems that emphasizes the use of objects, classes, and their interactions to model real-world concepts and requirements. In the case of file encryption using a noise image as a key, we can use OOAD to break down the problem into smaller, manageable components that can be implemented and tested independently.

Here is a possible OOAD approach for designing a file encryption system using a noise image as a key:

Identify the requirements: The first step in any software development project is to identify the requirements of the system. In this case, we need to define the encryption and decryption algorithms to be used, the format of the input and output files, the storage location of the encrypted files, and the user interface for the system.

Identify the objects: Next, we need to identify the objects that will be part of the system. These objects could include the noise image, the input file, the output file, the encryption and decryption algorithms, and the user interface components.

Define the classes: Once we have identified the objects, we can define the classes that will implement the objects. For example, we could define classes for the noise image, the input and output files, the encryption and decryption algorithms, and the user interface components.

Define the class relationships: The classes we have defined need to interact with each other to achieve the goals of the system. We can define these interactions using class relationships such as inheritance, composition, and aggregation.

Define the class methods: Each class needs to have methods that define its behavior. For example, the noise image class might have methods to generate a random noise image, to retrieve the values of specific pixels, and to convert the image to a format that can be used as an encryption key. The encryption and decryption algorithm classes might have methods to perform the encryption and decryption operations on the input and output files.

Define the interfaces: The user interface components need to interact with the other classes in the system. We can define the interfaces between the user interface components and the other classes using method signatures, event handlers, and callbacks.

Implement the classes: Once we have defined the classes, their relationships, and their methods, we can start implementing the code for each class.

Test the system: Finally, we need to test the system to ensure that it meets the requirements and works as expected. We can use automated tests to test the individual classes and integration tests to test the interactions between the classes.

In summary, designing a file encryption system using a noise image as a key requires us to identify the requirements, objects, classes, relationships, methods, and interfaces of the system. By following an OOAD approach, we can break down the problem into smaller, manageable components that can be implemented and tested independently, leading to a more robust and maintainable system.

### **3.1.1 Scenarios**

File encryption using a noise image as a key can be done in different ways depending on the specific encryption algorithm used. However, here are some general scenarios:

One-Time Pad Encryption: One-Time Pad is a simple and secure encryption method that uses a random key that is at least as long as the plaintext to encrypt. In this scenario, a noise image is generated and used as the one-time pad key. The noise image and the plaintext file are XORed to create the encrypted file. The recipient of the file must possess the same noise image to XOR with the encrypted file to retrieve the original plaintext.

Block Cipher Encryption: Block cipher encryption methods divide the plaintext into blocks of fixed size, which are encrypted using a key. In this scenario, the noise image is used as the key to encrypt each block of the plaintext file. The encrypted blocks are then concatenated to create the final encrypted file. The recipient of the file must possess the same noise image to decrypt each block and retrieve the original plaintext.

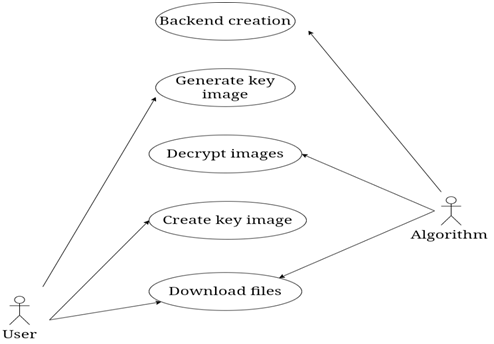
Stream Cipher Encryption: Stream cipher encryption methods encrypt the plaintext stream one bit or byte at a time, using a key stream generated from the key. In this scenario, the noise image is used to generate a pseudo-random key stream that is XORed with the plaintext stream to create the encrypted stream. The recipient of the file must possess the same noise image to generate the same key stream to XOR with the encrypted stream and retrieve the original plaintext.

In all of these scenarios, the noise image acts as a key to encrypt the file, and the recipient must possess the same noise image to decrypt the file. This ensures that the encrypted file remains secure, as anyone without access to the noise image will not be able to retrieve the original plaintext.

### **3.1.2 Use Case Diagram**

The functional requirements that the system satisfies are a critical component of the system. The high-level needs of the system are analyzed using Use Case diagrams. Different use cases are used to express these requirements. This UML diagram has three primary components that we can see:

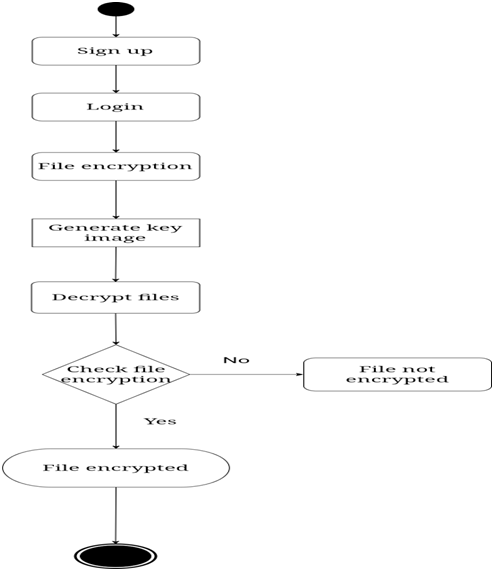
* Functional Requirements
* Actors
* Relationships



Algorithm

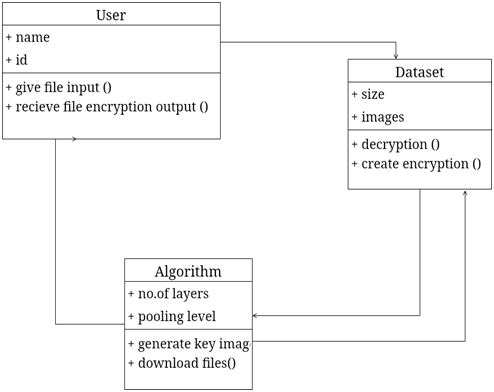
## **3.1.3 Activity Diagram**

An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modeling. They can also describe the steps in a use case diagram. Activities modeled can be sequential and concurrent.



**3.1.4 Class Diagram**

The most common diagram type for software documentation is the class UML diagram. Because most software today is still written using the Object-Oriented Programming paradigm, utilizing class diagrams to document the software is a sensible solution. Because OOP is based on classes and their relationships, this occurs.



### 

### 

### **3.1.5 Sequence Diagram**

### The sequence diagram is a good diagram to use to document a system's requirements and to flush out a system's design. The reason the sequence diagram is so useful is because it shows the interaction logic between the objects in the system in the time order that the interactions take place.

### 

**CHAPTER 4: IMPLEMENTATION DETAILS**

**Introduction**

**4.1 Algorithm used:**

Aims to encrypt and decrypt image data using the AES algorithm. The AES module in the Crypto.Cipher package provides an implementation of this algorithm.The get\_random\_bytes function from the Crypto.Random module is used to generate a secure random key of a specified length. This key is then used to encrypt and decrypt the data.The cv2 module is part of the OpenCV library, which is a computer vision and machine learning software library. It is used for image processing and computer vision tasks.The numpy module is a numerical computing library for Python, used for working with arrays and matrices.

This function generates a 256-bit (32-byte) random key using the get\_random\_bytes() function from the Crypto.Random module. The key is then converted to a bytearray using the bytearray() function and then converted to a flattened NumPy array using the np.array() function from the NumPy library.The grayscale image is then saved as a PNG file using the cv2.imwrite() function from the OpenCV library. The filename for the image is constructed using the filename argument and saved in the media/key/ directory.Overall, this function generates a random key, converts it to a grayscale image, saves the image as a PNG file, and returns the path to the saved file. This key can be used for encrypting and decrypting files using the AES algorithm.

this function encrypts a file using the AES algorithm with a key obtained from a grayscale image file, saves the encrypted file in a binary format, and returns the path to the encrypted file.The function first reads in the AES key from a grayscale image file located in the media/key/ directory. The filename of the key image file is constructed using the filename argument passed to the function. The key image is read in as a grayscale image using the cv2.imread() function from the OpenCV library.

The key image is then flattened into a one-dimensional NumPy array using the ravel() method.

The one-dimensional NumPy array is converted to a bytes object using the np.ndarray.tobytes() method.

The bytes object is used to create a new AES cipher object using the AES.new() function from the Crypto.Cipher module.

The contents of the file specified by the data\_file argument are read in using the open() function in binary mode and stored in the data variable.

The data is encrypted using the encrypt\_and\_digest() method of the AES cipher object. Theresulting ciphertext and tag are stored in the ciphertext and tag variables, respectively.

A new binary file is created with the filename specified by the filename argument in the media/files/ directory using the open() function in binary write mode.

The nonce, tag, and ciphertext are written to the new binary file using the write() method of the file object. The nonce is the initialization vector used in the AES encryption process and is required for decryption. The tag is used to verify the authenticity of the decrypted data.

The function returns the path to the encrypted file.

This function decaes is used to decrypt a file that has been encrypted using the encaes function from the same project. Here's how it works:

It reads the AES key from the image file generated by the generate\_key function and converts it into a numpy array using cv2.imread and np.ndarray.tobytes.

It opens the encrypted file and reads the nonce, tag, and ciphertext using the file\_in.read method.

It initializes the AES cipher with the key, mode (EAX), and nonce.

It decrypts the ciphertext and verifies the authenticity of the decrypted data using the tag.

It writes the decrypted data to a new file in the media/non\_enc\_file directory using the with open statement and the f.write method.Overall, this function allows you to decrypt a file that has been encrypted using the encaes function and restore it to its original state.

from Crypto.Cipher import AES

from Crypto.Random import get\_random\_bytes

import cv2

import numpy as np

def generate\_key(filename):

key = get\_random\_bytes(32)

randomByteArray = bytearray(key)

flatNumpyArray = np.array(randomByteArray)

grayImage = flatNumpyArray.reshape(4,8)

cv2.imwrite(f'media/key/{filename}\_aeskey.png', grayImage)

path = f'/media/key/{filename}\_aeskey.png'

return path

def encaes(data\_file, filename):

img = cv2.imread(f'media/key/{filename}\_aeskey.png', 0)

img = img.ravel()

key = np.ndarray.tobytes(img)

cipher = AES.new(key, AES.MODE\_EAX)

with open(f'{data\_file}', "rb") as f:

data = f.read()

ciphertext, tag = cipher.encrypt\_and\_digest(data)

enc\_file\_path = f'media/files/{filename}.bin'

file\_out = open(f"media/files/{filename}.bin", "wb")

[file\_out.write(x) for x in (cipher.nonce, tag, ciphertext)]

file\_out.close()

return enc\_file\_path

def decaes(data\_file, filename):

img = cv2.imread(f'media/key\_upload/{filename}\_aeskey.png', 0)

img = img.ravel()

key = np.ndarray.tobytes(img)

file\_in = open(f'{data\_file}', 'rb')

nonce, tag, ciphertext = [file\_in.read(x) for x in (16, 16, -1)]

cipher = AES.new(key, AES.MODE\_EAX, nonce)

original = cipher.decrypt\_and\_verify(ciphertext, tag)

with open(f'media/non\_enc\_file/{filename}', 'wb') as f:

f.write(original)

**4.2 Software Technologies:**

**Python**

Python is a popular object-oriented programming language having the capabilities of a high-level programming language. It's easy to learn syntax and portability capability makes it popular these days. The following facts given us the introduction to Python

Python was developed by Guido van Rossum at Stichting Mathematisch Centrum in the Netherlands.

It was written as the successor of the programming language named ‘ABC’. Its first version was released in 1991.

The name Python was picked by Guido van Rossum from a TV show named Monty Python’s Flying Circus.

It is an open source programming language which means that we can freely download it and use it to develop programs. It can be downloaded from [www.python.org..](https://www.python.org/)

Python programming language is having the features of Java and C both. It is having the elegant ‘C’ code and on the other hand, it is having classes and objects like java for object-oriented programming.

### **Characteristics of Python**

Following are important characteristics of Python Programming

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types, supports dynamic type checking.
* It supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

### **Applications of Python**

### As mentioned before, Python is one of the most widely used languages over the web. Few of them are as follows

* Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly
* Easy-to-read − Python code is more clearly defined and visible to the eyes
* Easy-to-maintain - Python's source code is fairly easy-to-maintain
* A broad standard library − Python's bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh
* Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code
* Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms

**Django:**

Django is a high-level, open-source web framework written in Python that follows the model-view-controller (MVC) architectural pattern. It provides a set of tools and functionalities that enable developers to build web applications rapidly and efficiently.

Django includes various built-in features such as an ORM (Object-Relational Mapping) for database operations, a templating engine for rendering views, routing for URL handling, and built-in support for user authentication and authorization. It also provides a robust admin interface for managing application data.

Some of the key features of Django include:

Object-Relational Mapping (ORM): Django's ORM allows developers to define database models as Python classes, which can be used to perform database operations without writing raw SQL queries.

Templating Engine: Django comes with a built-in templating engine that allows developers to create dynamic HTML templates that can be rendered with data from the application.

Routing: Django provides a URL routing system that maps URLs to views, allowing developers to easily handle different HTTP requests and route them to appropriate views.

Authentication and Authorization: Django has built-in support for user authentication and authorization, including features such as user registration, login, and password management.

Admin Interface: Django provides an admin interface that allows developers to easily manage application data through a web-based interface without having to build a custom admin panel.

Middleware: Django allows developers to use middleware, which are processing functions that can be applied to incoming HTTP requests and outgoing responses, allowing for additional processing or modification of requests and responses.

Security: Django has built-in security features such as protection against cross-site scripting (XSS), cross-site request forgery (CSRF), and SQL injection attacks.

Django is widely used for building web applications, from simple web apps to complex, data-driven websites, and it is known for its scalability, flexibility, and security. It follows the "Don't Repeat Yourself" (DRY) principle, which promotes code reusability and maintainability. Django has a large and active community of developers, which makes it a popular choice for web development projects.

**SQLite3:**

SQLite3 is a relational database management system (RDBMS). It provides a software library that allows applications to create, read, update, and delete data stored in a relational database. SQLite3 databases are organized into tables with rows and columns, and data is stored in a structured manner based on predefined schemas.

SQLite3 is a serverless database engine, which means it doesn't require a separate server process to be running. Instead, it is embedded into applications, and the database is stored as a single file on the local file system. This makes it a self-contained and portable database solution that can be used for local data storage in various applications, including embedded systems, desktop applications, mobile apps, and web applications.

SQLite3 supports standard SQL (Structured Query Language), which is a widely used language for querying and manipulating relational databases. It also provides features such as transactions, indexing, and query optimization for efficient data management. Additionally, SQLite3 is ACID-compliant, ensuring the integrity, consistency, and durability of data stored in the database.

In Python, SQLite3 is available as a built-in module called sqlite3, which provides an API for interacting with SQLite3 databases using Python code. This allows developers to use SQLite3 as a local data storage solution in Python applications and perform various database operations, such as creating tables, inserting data, updating data, and querying data, directly from Python code.

**GitHub:**

GitHub provides a platform for collaboration with your project teammates. You can receive feedback on your project from other developers and experts in the field. This can help you improve the quality of your code and identify and fix errors. Uploading your project on GitHub can make it more visible to the wider community. This can lead to increased exposure and potential collaboration opportunities

## **CHAPTER 5: TESTING**

Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.Some prefer saying Software testing definition as a White Box and Black Box Testing

**Integration Testing:**

Integration Testing is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated.

**White Box Testing**

White Box Testing is a testing technique in which software’s internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security. In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing.

The term “WhiteBox” was used because of the see-through box concept. The clear box or WhiteBox name symbolizes the ability to see through the software’s outer shell (or “box”) into its inner workings. Likewise, the “black box” in “Black Box Testing” symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.

How do you perform White Box Testing?

We have divided it into two basic steps to give you a simplified explanation of white box testing. This is what testers do when testing an application using the white box testing technique:

STEP 1) UNDERSTAND THE SOURCE CODE

STEP 2) CREATE TEST CASES AND EXECUTE

**Black Box Testing**

Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.

**Black Box Testing and White Box Testing**

The main focus of black box testing is on the validation of your functional requirements. White Box Testing (Unit Testing) validates internal structure and working of your software code. Black box testing gives abstraction from code and focuses on testing effort on the software system behavior. To conduct White Box Testing, knowledge of underlying programming language is essential. Current day software systems use a variety of programming languages and technologies and it’s not possible to know all of them. Black box testing facilitates testing communication amongst modules. White box testing does not facilitate testing communication amongst module.

**TEST CASE TABLE**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| S.No | Requirements | Req.No. | Essential/Desirable | Description | Expected output | Actual output | Result |
| 1. | Register | RS1 | Essential | Password should contain special characters and numbers | The user’s credentials will be registered to the system. | The system will store the user's login information | Success |
| 2. | login | RS2 | Essential | Login credentials are required. | The user will be verified | The system will allow the authentic user to login. | Success |
| 3. | Uploading the file | RS3 | Essential | Uploading the file for encryption. | The file will be encrypted. | The encrypted file will be stored in the file list. | Success |
| 4. | Encryption | RS4 | Essential | The file will be converted into bytecode by using AES algorithm. | The noise image will be downloaded in the system. | The key for decryption is generated as noise image. | Success |
| 5. | Decryption | RS6 | Essential | Using noise image to decrypt the file. | Decrypting the file involves uploading the generated key. | Uploading the generated key to decrypt the file. | Success |

**CHAPTER 6: CONCLUSION**

File encryption is achieved with the use of complex algorithms. If the information contained in a file has been scrambled using an encryption method, the file is said to be encrypted. A scrambled file is unreadable, but there are still many cyber-attacks happening in many ways, posing a threat to the confidentiality of the information. In this work, we addressed noise pictures as the key for file encryption and examined numerous case studies. Experimental results reveal that the suggested noise picture as a key for encryption and decryption procedures is both secure and reliable, opening the door for its use in high-security image communication applications.

**REFERENCES:**

1.<https://ieeexplore.ieee.org/document/9502246>

2.<https://ieeexplore.ieee.org/document/9706273>

3.<https://ieeexplore.ieee.org/document/9315245>

4.<https://ieeexplore.ieee.org/document/9701774>

5.<https://ieeexplore.ieee.org/document/9847932>

6.<https://ieeexplore.ieee.org/document/9681070>

7.<https://ieeexplore.ieee.org/document/9253112>

8.<https://ieeexplore.ieee.org/document/9924954>

9.<https://ieeexplore.ieee.org/document/9835957>

10.<https://ieeexplore.ieee.org/document/9337031>

[1] T. M. K. Afandi, D. H. Fandiantoro, Endroyono and I. K. E. Purnama, "Medical Images Compression and Encryption using DCT, Arithmetic Encoding and Chaos-Based Encryption," 2021 International Seminar on Intelligent Technology and Its Applications (ISITIA), 2021

[2] C. Qin, J. Hu, F. Li, Z. Qian and X. Zhang, "JPEG Image Encryption with Adaptive DC Coefficient Prediction and RS Pair Permutation," in IEEE Transactions on Multimedia,2022

[3] M. D and S. Vasuhi, "Image Steganography: 2-Bit XOR Algorithm Used In YCbCr Color Model With Crypto-algorithm," 2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP), 2020

[4] J. Tong, Y. Long and Q. Liu, "A File Encryption System Based on Attribute Based Encryption," 2021 17th International Conference on Computational Intelligence and Security (CIS), 2021

[5] A. N, A. V. K and N. R, "Sharing Confidential Images with Abbreviated Shares using Steganography and AES Algorithm," 2022 2nd International Conference on Intelligent Technologies (CONIT), 2022

[6] H. Nazir, I. S. Bajwa, S. Abdullah, R. Kazmi and M. Samiullah, "A Color Image Encryption Scheme Combining Hyperchaos and Genetic Codes," in IEEE Access, 2022

[7] O. Q. J. Al-Thahab and A. A. Hussein, "Implementation Of Stego-Watermarking Technique by Encryption Image Based On Turbo Code For Copyright Application," 2020 1st. Information Technology To Enhance e-learning and Other Application (IT-ELA), 2020

[8] P. Oktivasari, M. Agustin, R. E. M. Akbar, A. Kurniawan, A. R. Zain and F. A. Murad, "Analysis of ECG Image File Encryption using ECDH and AES-GCM Algorithm," 2022 7th International Workshop on Big Data and Information Security (IWBIS), 2022

[9]  S. Patel and T. V, "New Image Encryption Algorithm based on Pixel Confusion-Diffusion using Hash Functions and Chaotic Map," 2022 7th International Conference on Communication and Electronics Systems (ICCES), 2022

[10] M. E. Kahla, M. Beggas, A. Laouid, M. Kara and M. AlShaikh, "Asymmetric Image Encryption Based on Twin Message Fusion," 2021 International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP), 2021

## 

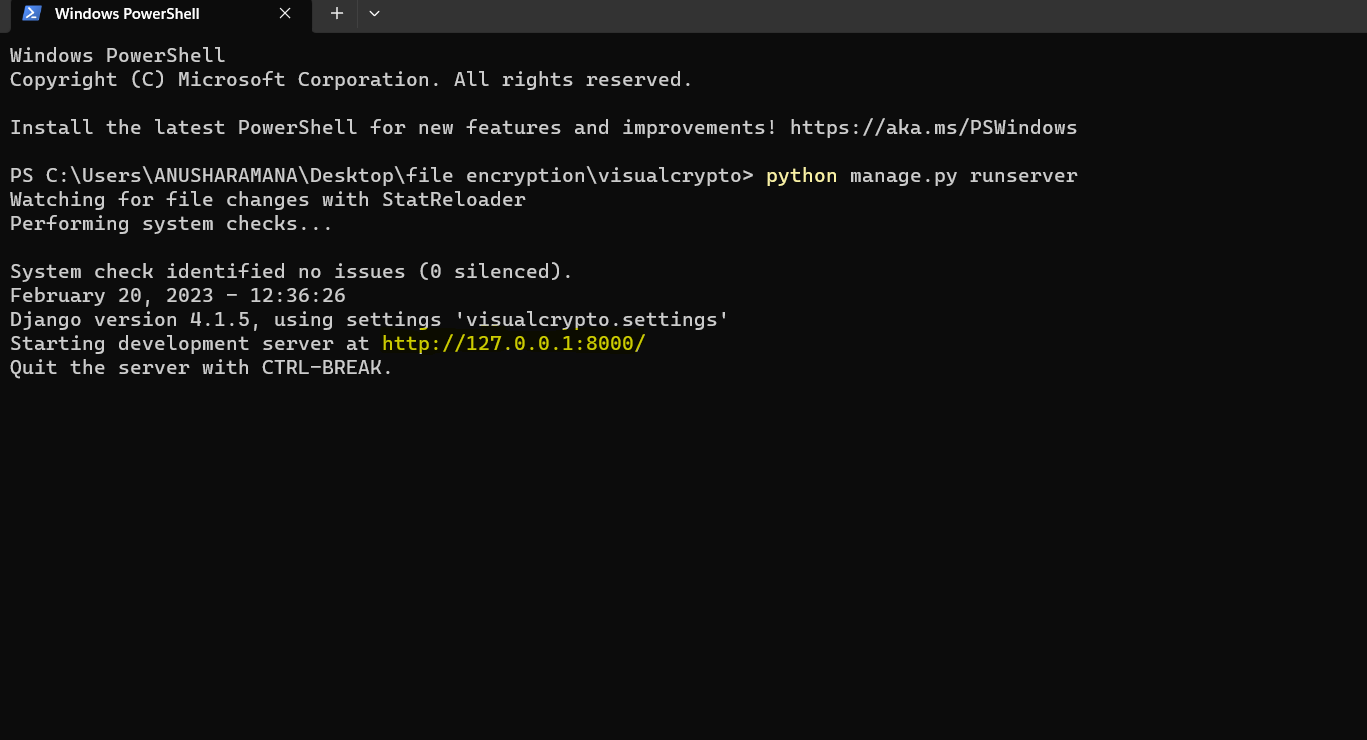
## 

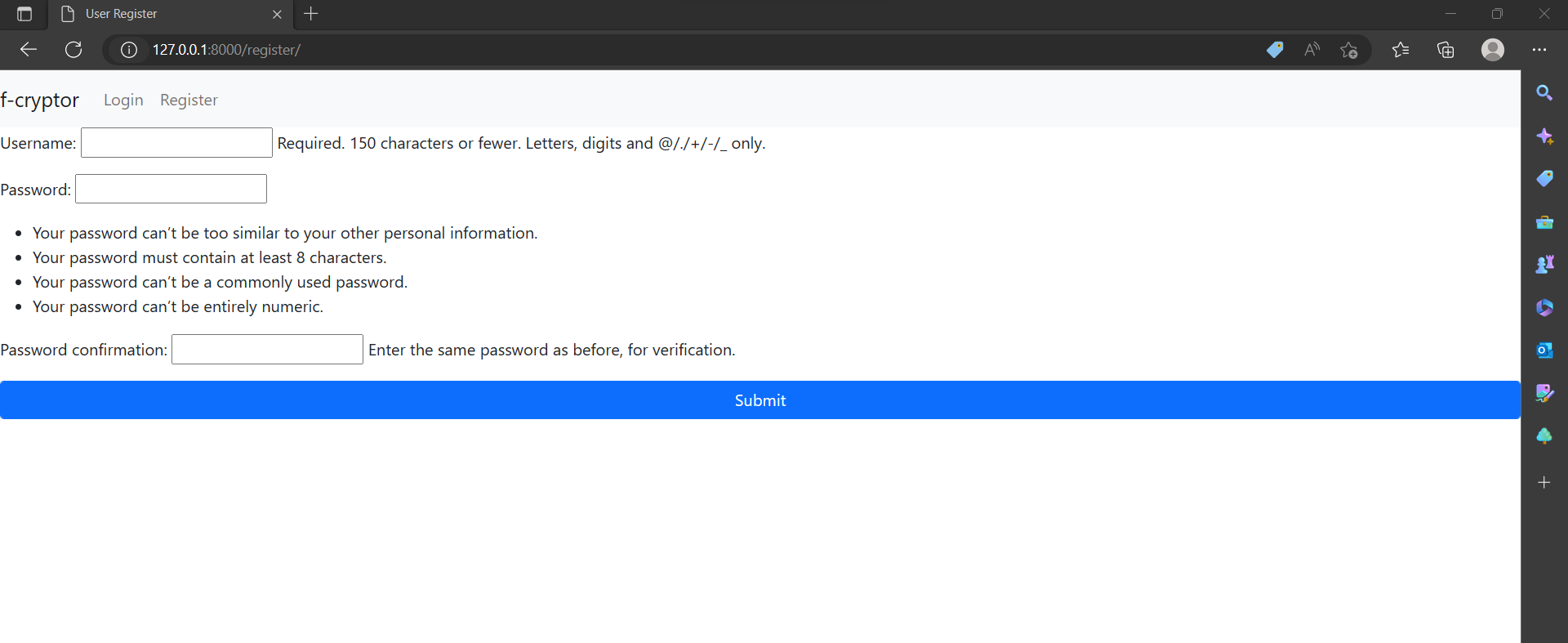
## 

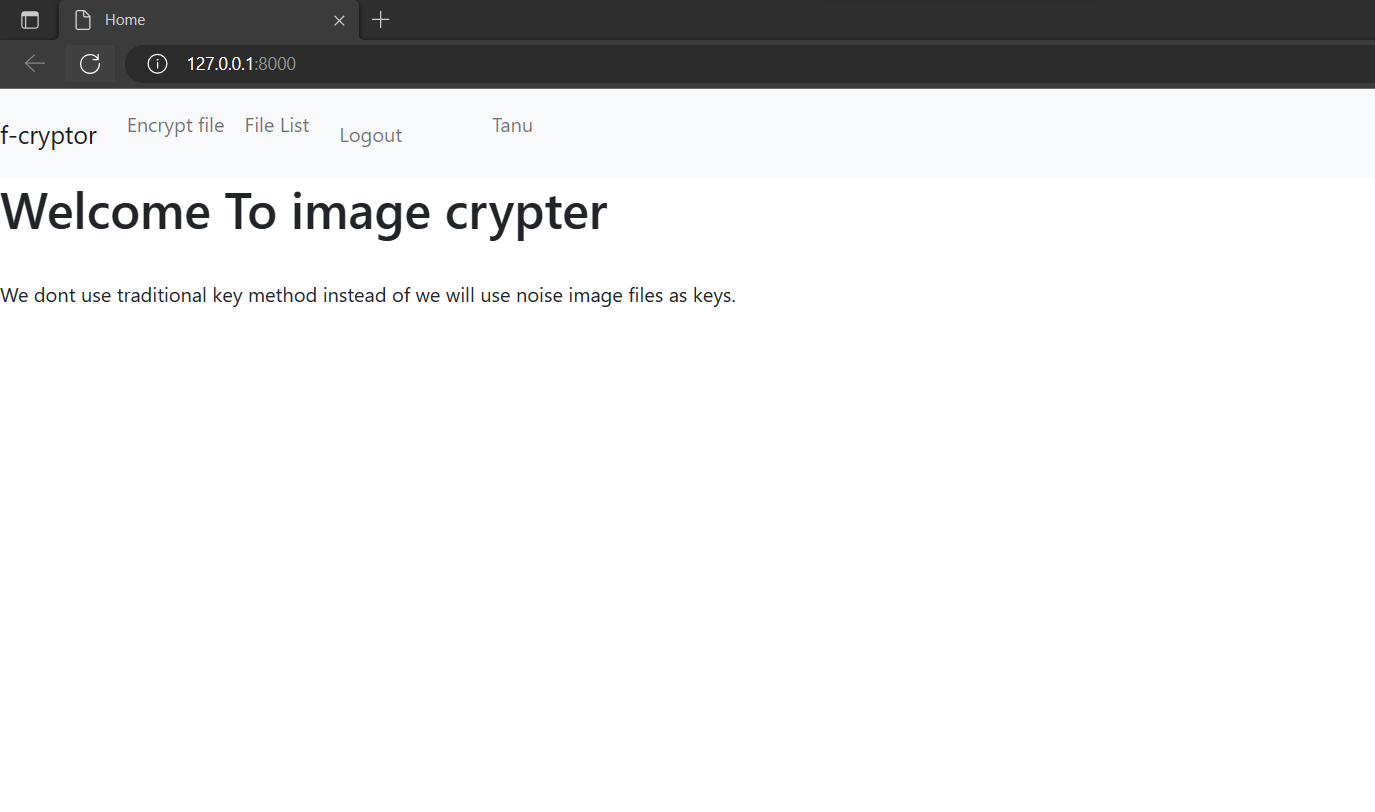
## 

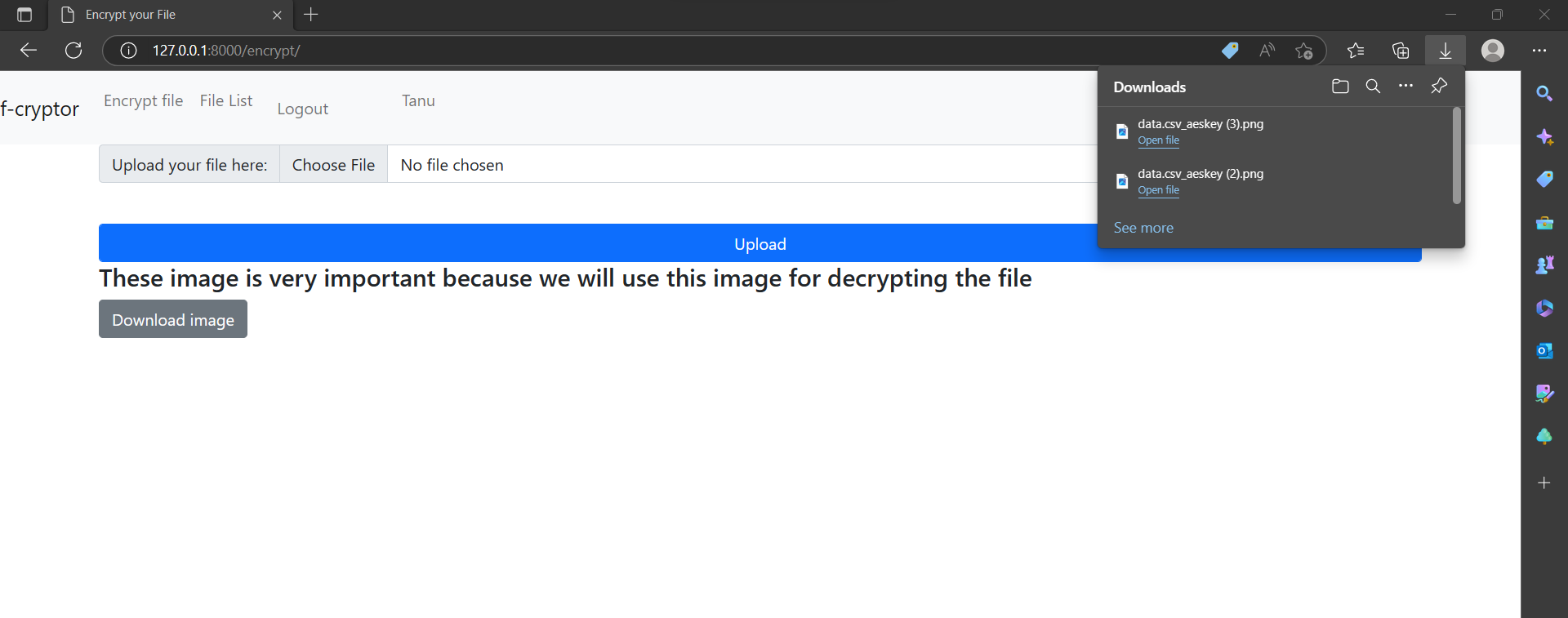
**APPENDIX**

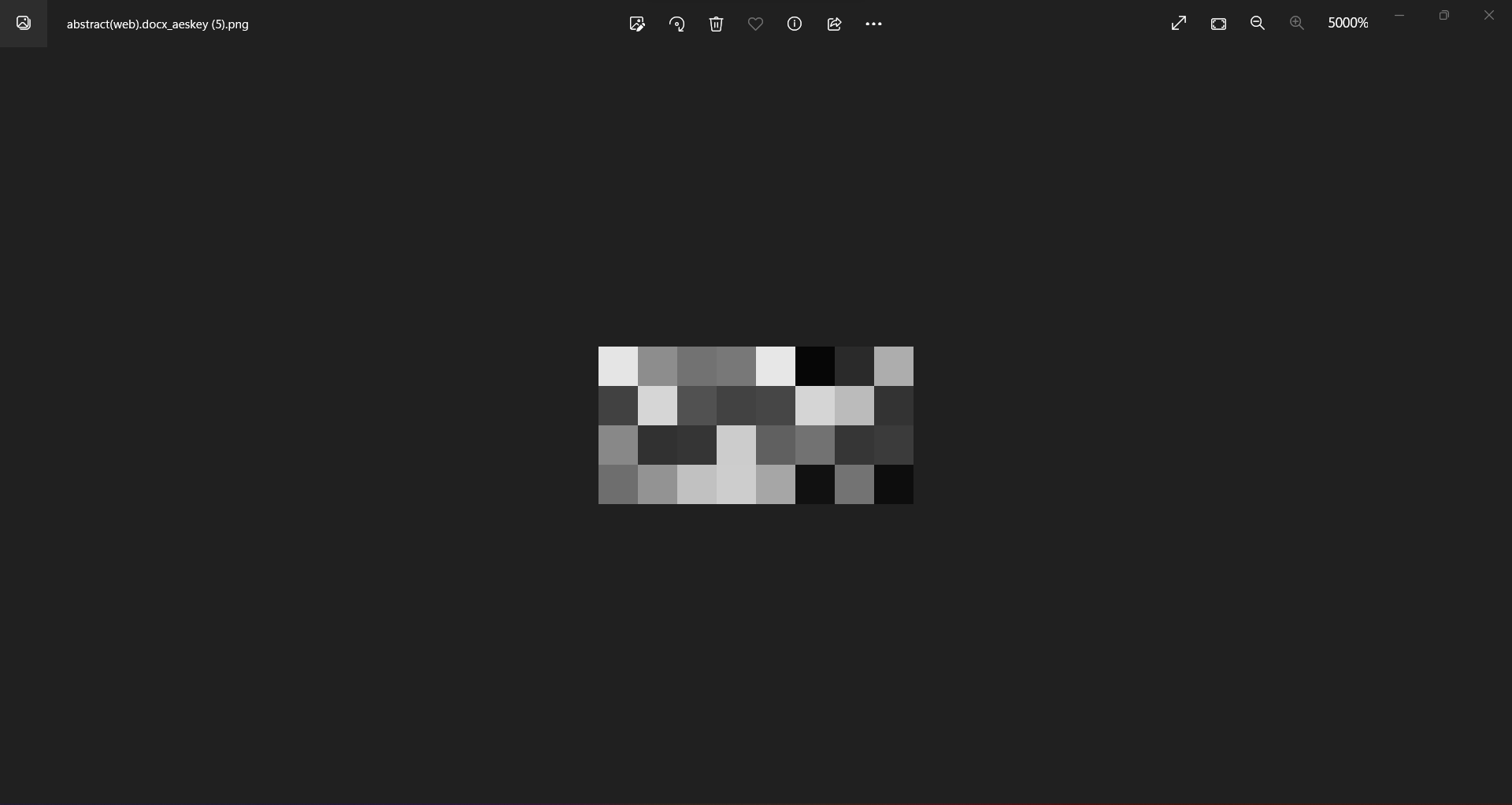
**A-OUTPUT SCREENS**

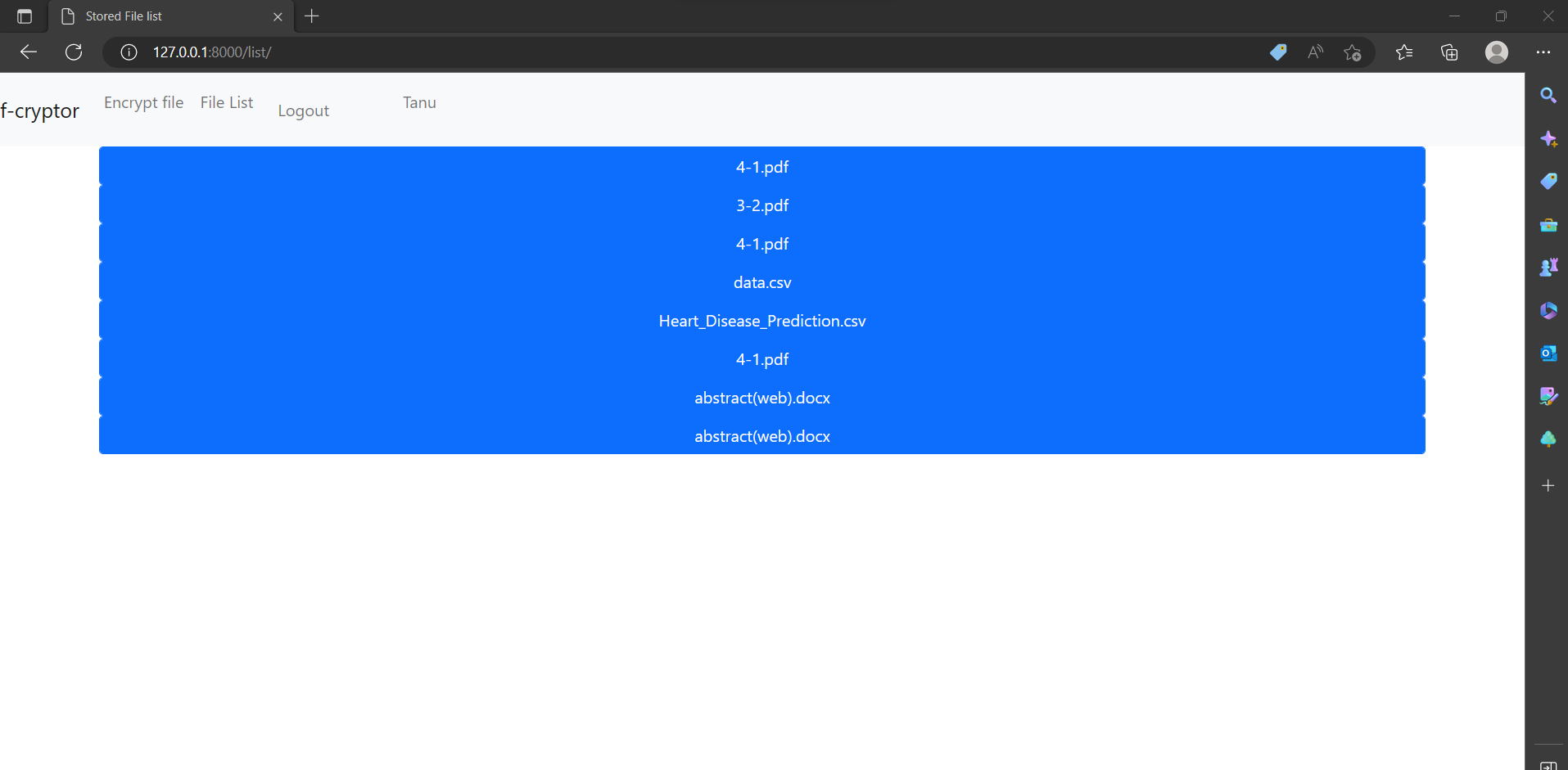


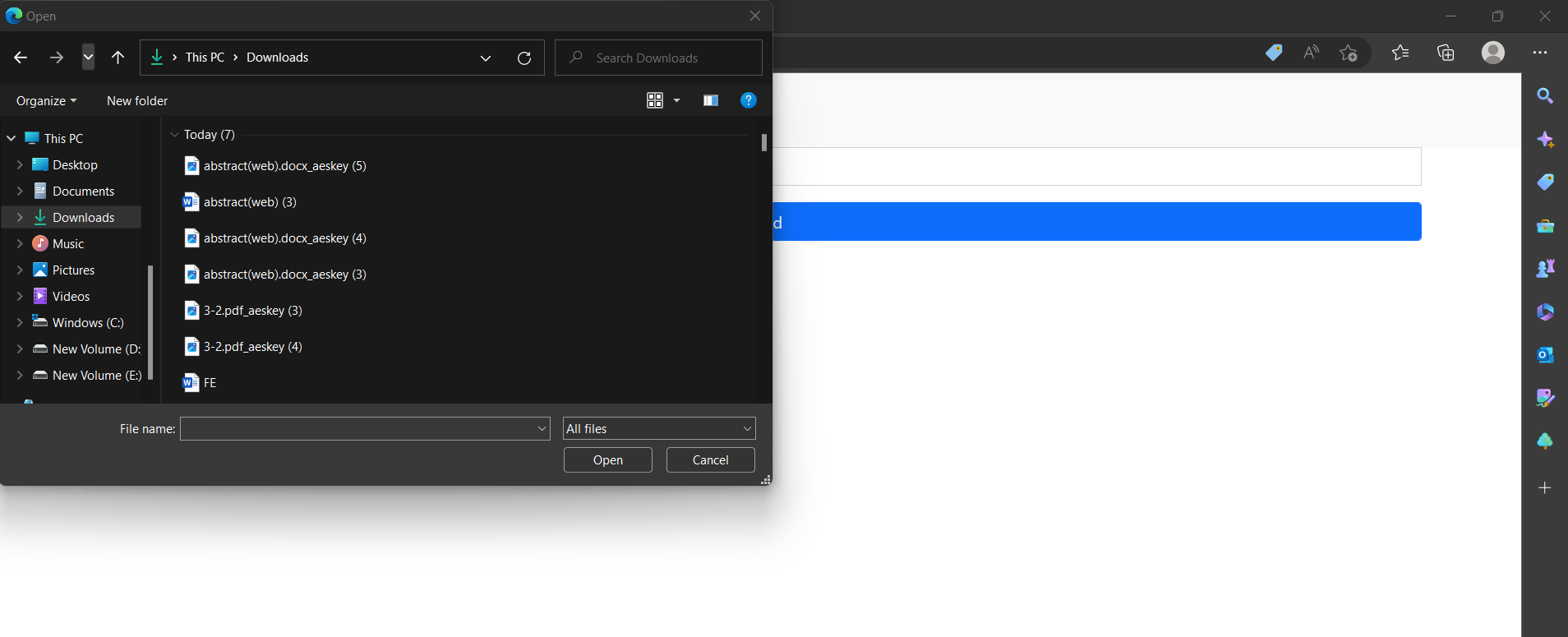


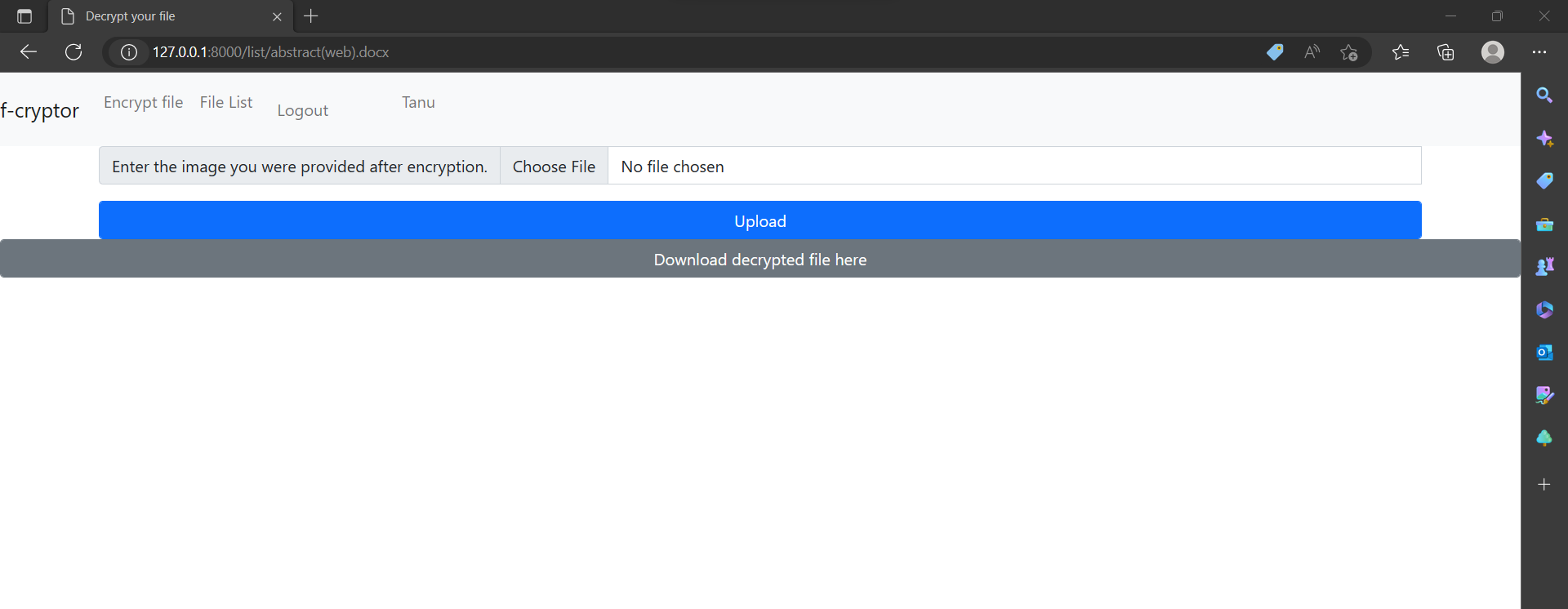
****

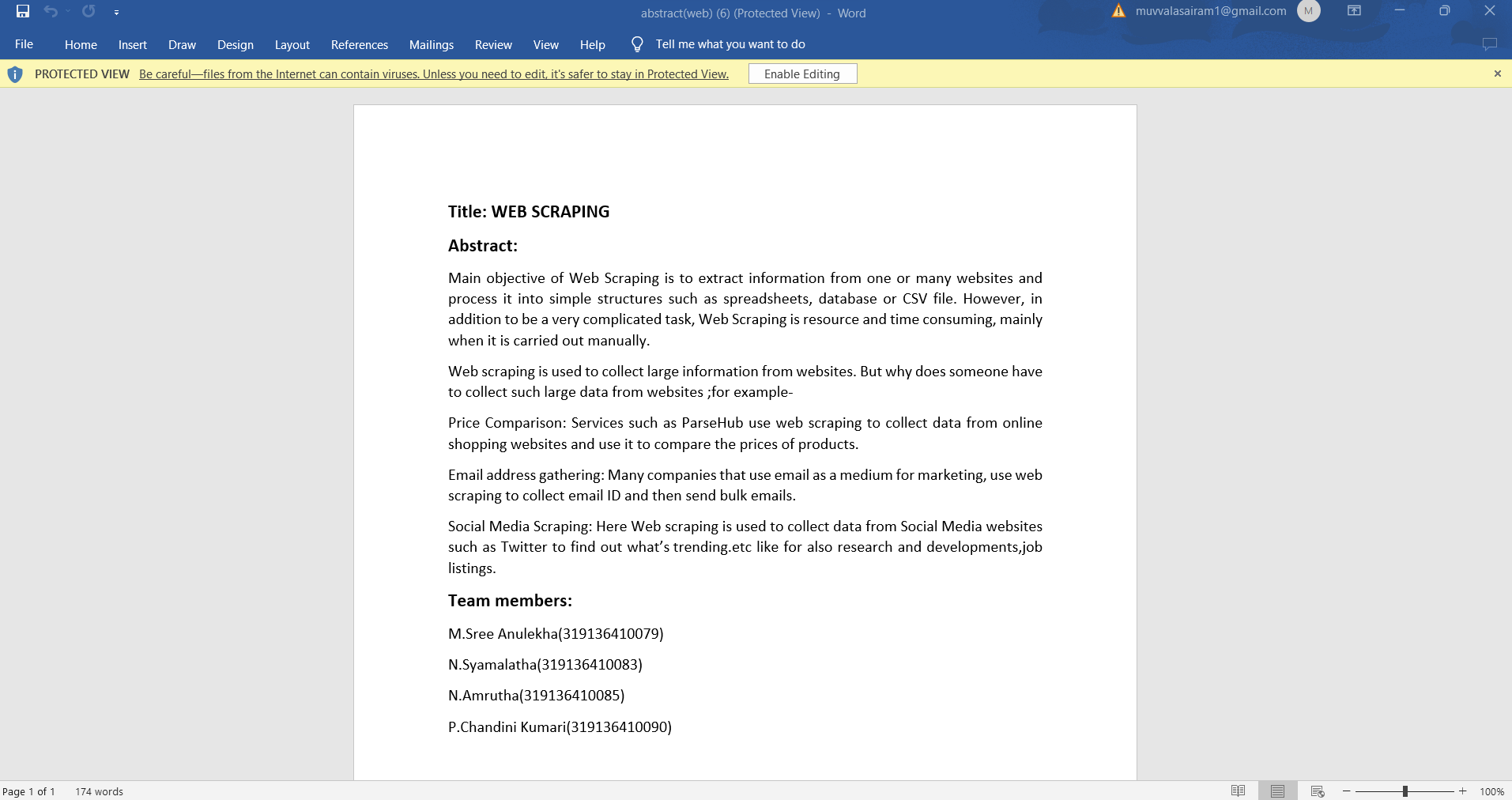
****

****

****

****

****

****

**B-SAMPLE CODE**

from Crypto.Cipher import AES

from Crypto.Random import get\_random\_bytes

import cv2

import numpy as np

def generate\_key(filename):

key = get\_random\_bytes(32)

randomByteArray = bytearray(key)

flatNumpyArray = np.array(randomByteArray)

grayImage = flatNumpyArray.reshape(4,8)

cv2.imwrite(f'media/key/{filename}\_aeskey.png', grayImage)

path = f'/media/key/{filename}\_aeskey.png'

return path

def encaes(data\_file, filename):

img = cv2.imread(f'media/key/{filename}\_aeskey.png', 0)

img = img.ravel()

key = np.ndarray.tobytes(img)

cipher = AES.new(key, AES.MODE\_EAX)

with open(f'{data\_file}', "rb") as f:

data = f.read()

ciphertext, tag = cipher.encrypt\_and\_digest(data)

enc\_file\_path = f'media/files/{filename}.bin'

file\_out = open(f"media/files/{filename}.bin", "wb")

[file\_out.write(x) for x in (cipher.nonce, tag, ciphertext)]

file\_out.close()

return enc\_file\_path

def decaes(data\_file, filename):

img = cv2.imread(f'media/key\_upload/{filename}\_aeskey.png', 0)

img = img.ravel()

key = np.ndarray.tobytes(img)

file\_in = open(f'{data\_file}', 'rb')

nonce, tag, ciphertext = [file\_in.read(x) for x in (16, 16, -1)]

cipher = AES.new(key, AES.MODE\_EAX, nonce)

original = cipher.decrypt\_and\_verify(ciphertext, tag)

with open(f'media/non\_enc\_file/{filename}', 'wb') as f:

f.write(original)

views……………………………………………………

from django.shortcuts import render, redirect

from .models import File\_handler

from algorithm import AES\_algo as aes

from django.http import HttpResponse

from django.contrib.auth.forms import UserCreationForm

from django.contrib.auth import authenticate, login, logout

# Create your views here.

def home(request):

return render(request, 'home.html')

def upload\_and\_encrypt(request):

context = {}

if request.user.is\_authenticated:

if request.method == 'POST':

file = request.FILES['file']

filename = file.name

user = request.user

normalfile = file.read()

with open(f'media/non\_enc\_file/{filename}', 'wb') as f:

f.write(normalfile)

FILE\_PATH = 'media/non\_enc\_file/' + filename

# Now encryption Phase Comes in

path = aes.generate\_key(filename)

encrypted\_file\_path = aes.encaes(FILE\_PATH, filename)

file\_dets = File\_handler(

user=user, filename=filename, enctrypted\_file\_path=encrypted\_file\_path)

file\_dets.save()

context['key\_img'] = path

return render(request, 'encrypt.html', context)

return HttpResponse('<h1>You are not authenticated</h1>')

def decrypt\_and\_download(request, filename):

context = {}

if request.method == 'POST':

img = request.FILES['key\_image']

image\_data = img.read()

with open(f'media/key\_upload/{filename}\_aeskey.png', 'wb') as f:

f.write(image\_data)

file\_path = '/' + f'media/non\_enc\_file/{filename}'

context['file\_path'] = file\_path

return render(request, 'decrypt.html', context)

return render(request, 'decrypt.html', context)

def file\_list(request):

if request.user.is\_authenticated:

context = {}

user = request.user

all\_file\_dets = File\_handler.objects.filter(user=user)

if len(all\_file\_dets) == 0:

return HttpResponse('<h2>No file Can be fetched! Looks like you havent encrypted any file</h2>')

context['all\_files'] = all\_file\_dets

return render(request, 'file\_list.html', context)

return HttpResponse('<h2>Looks like there are some errors!</h2>')

def UserRegister(request):

context = {}

form = UserCreationForm()

context['form'] = form

if request.method == 'POST':

form = UserCreationForm(request.POST)

if form.is\_valid():

user = form.save()

login(request, user)

return redirect('/')

return render(request, 'user\_register.html', context)

def UserLogin(request):

if request.method == 'POST':

username = request.POST.get('username')

password = request.POST.get('password')

user = authenticate(request, username=username, password=password)

if user is not None:

login(request, user)

return redirect('/')

elif user is None:

return HttpResponse('<h2>User not Found Error!!</h2>')

return render(request, 'user\_login.html')

def UserLogout(request):

logout(request)

return redirect('home')

urls……………………………………

from django.urls import path

from .views import \*

urlpatterns = [

path('', home, name='home'),#done

path('login/', UserLogin, name='user\_login'),#done

path('register/', UserRegister, name='user\_register'),#done

path('encrypt/', upload\_and\_encrypt, name='encrypt'),#done

path('list/', file\_list, name= 'fileList'),

path('list/<filename>', decrypt\_and\_download, name= 'decrypt'),

path('logout/', UserLogout, name='user\_logout')#done

modules……………………………………….

from django.urls import path

from .views import \*

urlpatterns = [

path('', home, name='home'),#done

path('login/', UserLogin, name='user\_login'),#done

path('register/', UserRegister, name='user\_register'),#done

path('encrypt/', upload\_and\_encrypt, name='encrypt'),#done

path('list/', file\_list, name= 'fileList'),

path('list/<filename>', decrypt\_and\_download, name= 'decrypt'),

path('logout/', UserLogout, name='user\_logout')#done