IMAGE ENCRYPTION

21CSC203P/ADVANCED PROGRAMMING PRACTICE

PROJECT REPORT

Submitted by

S SUJAN (RA2311003020537)

Under the guidance of

Mr. A. Madhu

(Assistant Professor, Department of Computer Science and Engineering)

III SEMESTER/II YEAR

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

RAMAPURAM, CHENNAI-600089.

October 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled **IMAGE ENCRYPTION** is the bonafide work of **S SUJAN (RA2311003020537)** who carried out the project work under my supervision. This project work confirms to 21CSC203P/ADVANCED PROGRAMMING PRACTICE, III Semester, II year, 2024-2025.

SIGNATURE

Mr. A. Madhu

Assistant Professor

Dept. of Computer Science & Engineering

SRM Institute of Science and Technology

Ramapuram, Chennai

SIGNATURE

Dr. K. RAJA

Professor & Head

Dept. of Computer Science & Engineering

SRM Institute of Science and Technology

Ramapuram, Chennai.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

RAMAPURAM, CHENNAI

DECLARATION

We hereby declare that the entire work contained in this project report titled "**IMAGE ENCYPTION"** has been carried out by SUJAN S (RA2311003020537) at SRM Institute of Science and Technology, Ramapuram, Chennai, under the guidance of Mr. Madhu A, Assistant professor, Department of Computer Science and Engineering.

Place: Chennai **SUJAN S**

Date:

ABSTRACT

With the increasing reliance on digital media, ensuring the security of visual data, especially images, has become a priority in numerous applications ranging from social media to confidential data storage. This report delves into the concept of image encryption, which plays a critical role in protecting images from unauthorized access and ensuring data privacy. The primary focus of this project is to implement and analyze various encryption algorithms using Java, a versatile programming language well-suited for such cryptographic tasks.

The report covers both symmetric and asymmetric encryption techniques and evaluates their performance in terms of security and efficiency when applied to image data. We explore commonly used algorithms such as AES (Advanced Encryption Standard) and RSA (Rivest–Shamir–Adleman), comparing their ability to safeguard images while maintaining data integrity. Additionally, we examine the challenges associated with encrypting large image files and discuss potential optimization strategies.

Through this project, we aim to demonstrate the significance of image encryption in modern digital communication and storage systems, providing insights into how encryption can be effectively applied to protect visual content in today's technologically driven world.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	**iv**
	LIST OF FIGURES	**vi**
	LIST OF TABLES	**vi**
1	**INTRODUCTION**	**1**
	1.1 Problem Statement	1
	1.2 Objective	1
	1.3 Scope of Project	2
2	**REQUIREMENTS SPECIFICATION**	**4**
	2.1 Functional Requirements	4
	2.2 User Interface Requirements	5
3	**SYSTEM DESIGN**	**6**
	3.1 Module Description	6
	3.2 Use-Case Diagram	6
	3.3 Class Diagram	7
	3.4 Entity-Relationship Diagram	8
	3.5 System Architecture Diagram	8
	3.6 Software Requirements	8
4	**IMPLEMENTATION**	**9**
	4.1 Java Packages, Classes, and Methods	9
5	**CONCLUSION**	**11**
	5.1 Conclusion	11
A1	**APPENDIX 1 – SAMPLE CODE**	**12**
A2	**APPENDIX 2 – SCREEN SHOTS**	**13**
	REFERENCES	**18**

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
3.1	Use Case Diagram of Image Encryption	7
3.2	Class Diagram of Image Encryption	8
A2.1	Home Screen	13
A2.2	Image Selection for Encryption	14
A2.3	Displays File Name Before Encryption	15
A2.4	Pop-Up Message After Encryption	16
A2.5	Selecting The Encrypted File for Decrypting	16
A2.6	Enter the Decryption Key	17
A2.7	Pop-Up Message After Decryption	17

CHAPTER 1
INTRODUCTION

The introduction chapter sets the stage for the entire report by giving a thorough background on the importance of securing image data and the role of encryption in ensuring data confidentiality. It also outlines the motivation for choosing this project and what the reader can expect.

1.1 Problem Statement

This section identifies and explains the problem that necessitates the use of image encryption. In today's digital landscape, vast amounts of sensitive image data are exchanged over the internet or stored on personal devices, leaving them vulnerable to interception, theft, or unauthorized access. For instance, images may contain confidential business documents, personal photos, or sensitive health information (in the case of medical imaging). Hackers or unauthorized users can exploit vulnerabilities to gain access to such images, leading to potential misuse or privacy violations. This subtopic should thoroughly explain:

- The real-world risks associated with insecure image handling.
- The consequences of image data breaches (personal privacy loss, financial damage, corporate espionage).
- The need for robust encryption methods to protect image data at rest (on storage devices) and in transit (when shared over networks).

1.2 Objective

Here, you describe the primary goals your project aims to achieve. In this case, the main objective is to apply cryptographic techniques to ensure the confidentiality, integrity, and authenticity of image data. Objectives might include:

- **Understanding Encryption Techniques:** The project aims to explore both symmetric (AES) and asymmetric (RSA) encryption algorithms and apply them to images.
- **Implementation Using Java:** To implement encryption and decryption functionalities using Java, a language that provides extensive libraries for cryptographic tasks.
- **Comparative Analysis:** The objective may also include evaluating the performance of different algorithms concerning their encryption speed, key strength, and resource usage, such as CPU and memory.
- **Secure Image Handling:** Ensure that image files can be encrypted and decrypted without losing their visual quality or metadata (like timestamps, image format information).

1.3 Scope of the Project

This subtopic outlines the boundaries of the project to give the reader a clear understanding of what is covered and what is outside the scope. Important points include:

- **Encryption Techniques:** The project may focus on specific cryptographic algorithms like AES (Advanced Encryption Standard) for symmetric encryption and RSA (Rivest–Shamir–Adleman) for asymmetric encryption. More advanced or exotic encryption techniques, such as elliptic curve cryptography, may be beyond the project's scope.
- **Image Formats:** The project might support common image formats such as JPEG, PNG, BMP, etc. Other formats, such as RAW or TIFF, may not be handled due to complexity or file size.
- **File Size Limitations:** The project may place limits on the file size that can be encrypted or decrypted due to performance concerns (large files might require more sophisticated optimization techniques that are beyond the scope).

- **Java Libraries:** The project scope could be limited to the use of specific Java libraries like the Java Cryptography Extension (JCE) and exclude advanced third-party libraries that require more complex integration.
- **Real-time vs Batch Processing:** The project might focus on batch processing of files rather than real-time encryption (e.g., during video streaming).
- **Error Handling:** Basic error handling will be provided (like file not found, incorrect decryption key), but advanced scenarios like handling partial corruption during transmission might be outside the scope.

CHAPTER 2
SYSTEM ANALYSIS

2. REQUIREMENTS SPECIFICATION

The requirements specification defines what the system is supposed to do. It specifies all the functionalities, constraints, and technical requirements that the project must meet.

2.1 Functional Requirements

This section defines the core functions or operations that the system must perform. Each functional requirement outlines a distinct feature of the project. Some key functional requirements for an image encryption system include:

- **Encryption:** The system must take an input image and apply the specified encryption algorithm (e.g., AES) to produce an encrypted image file that cannot be viewed without decryption.
    - **Input:** Image file (JPEG, PNG, etc.), encryption key.
    - **Output:** Encrypted file (cipher image that appears as random noise).
- **Decryption:** The system must reverse the encryption process by taking the encrypted image and decrypting it using the appropriate decryption key. The output should be the original image.
    - **Input:** Encrypted image file, decryption key.
    - **Output:** Original image file restored in its original format.
- **Key Generation:** For symmetric encryption like AES, the system should provide functionality for generating a pair of cryptographic keys (public and private keys). For symmetric encryption, it should generate secure keys of a specified length (e.g., 128-bit or 256-bit keys for AES).
- **File Handling:** The system should support reading and writing image files in various formats. It must maintain file integrity, ensuring no data is lost during encryption and decryption.
- **Error Handling:** The system should be capable of handling errors, such as:
    - Incorrect file format.
    - Corrupted or unreadable image files.
    - Incorrect decryption key.
    - File path issues (e.g., file not found).

2.2 User Interface Requirements

This section focuses on how users will interact with the system. Even if the system uses a command-line interface (CLI), the design must still consider user experience. Requirements for a GUI, if applicable, might include:

- **Command-line Interface (CLI):** Users should be able to perform encryption and decryption by entering commands and specifying image paths and keys. The interface should provide clear prompts and feedback.
    - **Encryption Command:** User provides the image path and key. The system returns a confirmation of successful encryption and provides the path to the encrypted file.
    - **Decryption Command:** User provides the encrypted image path and the correct decryption key. The system restores the original image and provides confirmation.
- **Graphical User Interface (GUI):** (If implemented) The interface should allow users to select files, input keys, and click buttons to encrypt or decrypt images. Additional features may include drag-and-drop functionality, status bars for progress, and clear error messages.

CHAPTER 3
SYSTEM DESIGN

System design outlines the technical structure and architecture of the system. It explains how different modules will interact with one another and how the system is organized.

3.1 Module Description

This section breaks down the project into several logical modules, each responsible for a different part of the system's functionality. Each module will be described in detail:

- **Encryption Module:** This module will handle the encryption logic, using algorithms like AES or RSA. It will take the image as input and transform it into an unreadable format.
- **Decryption Module:** This module will reverse the encryption, restoring the original image. It uses the appropriate decryption algorithm and key.
- **File Handling Module:** Responsible for reading and writing files from/to the disk. It ensures that the correct file formats (JPEG, PNG) are supported, and that file integrity is maintained.
- **Key Management Module:** This module generates and manages cryptographic keys. For symmetric encryption, it might generate a single key. For asymmetric encryption, it generates both public and private keys.

Each module will be detailed in terms of:

- Inputs (e.g., image files, encryption keys).
- Outputs (e.g., encrypted images, decrypted originals).
- Internal operations (e.g., how the encryption algorithm is applied).

3.2 Use-Case Diagram

A **Use-Case Diagram** visually represents how the system interacts with users. It identifies the main actions or use cases and the external entities (users) interacting with the system. Example use cases include:

- **Encrypt Image:** The user selects an image and provides an encryption key to encrypt the image.

- **Decrypt Image:** The user selects an encrypted image and provides the correct decryption key to restore the original.
- **Generate Key (Optional):** The user can request the system to generate a cryptographic key for encryption or decryption. The diagram will show the relationship between the user and each of these actions, making it easier to understand the system's functionality at a glance.

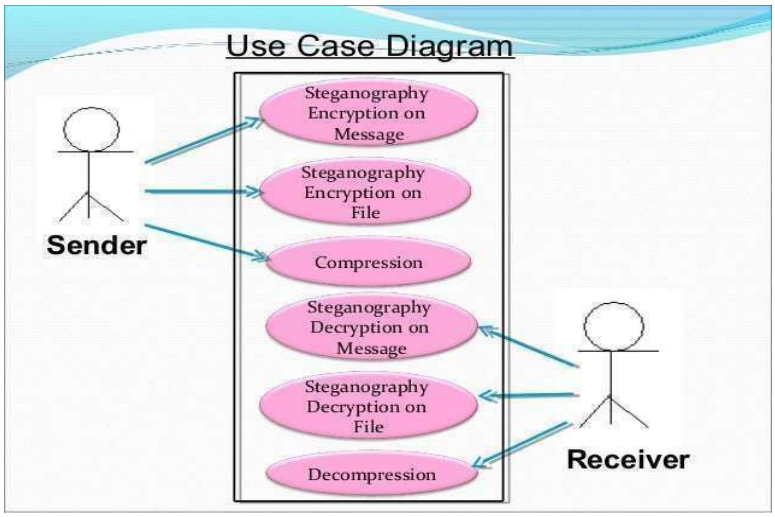

Fig 3.1: Use Case Diagram of Image Encryption

3.3 Class Diagram

A **Class Diagram** shows the object-oriented structure of the project, detailing the system's classes, attributes, methods, and relationships. It provides a blueprint for how the software will be implemented:

- **Classes:** Main classes might include:
  - ImageEncryptor: Responsible for encrypting images.
  - ImageDecryptor: Responsible for decrypting images.
  - FileHandler: Manages reading and writing images to/from disk.
  - KeyManager: Handles cryptographic key generation and management.
- **Attributes:** Each class will have attributes like file paths, encryption keys, and image formats.
- **Methods:** Methods might include encryptImage(), decryptImage(), readFile(), writeFile(), generateKey().

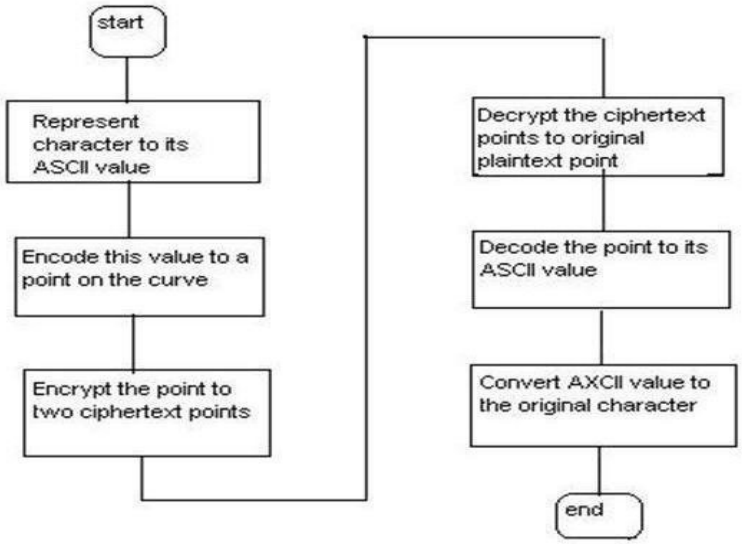

Fig 3.2: Class Diagram of Image Encrypt

3.4 Entity-Relationship Diagram (ERD)

An **Entity-Relationship Diagram** is particularly useful if your project uses a database to store information about encrypted images, users, or keys. The ERD defines the relationships between different entities:

- **Entities:** These might include Image, Key, User.
- **Relationships:** You would specify how these entities relate to one another. For instance, an Image might be encrypted by a Key, and a User might own several encrypted Images.

3.5 System Architecture Diagram

This diagram provides a high-level view of how the system components interact and how data flows through the system. It shows:

- **User Interface (CLI/GUI):** How the user interacts with the system.
- **Encryption Engine:** The component responsible for performing encryption and decryption.

3.6 Software Requirements

- Programming Language : JAVA,
- DBMS : MySQL
- JAVA IDE : Eclipse IDE

CHAPTER 4
IMPLEMENTATION

The **Implementation** chapter provides a detailed explanation of how the system was built. It covers the actual code structure, libraries used, and the logic behind how different functionalities were programmed. This section should cover the following key elements:

4.1 Java Packages, Classes, and Methods

This subtopic discusses the code organization in terms of Java packages, the classes that make up the project, and the methods implemented within each class. Each package and class serves a specific role in the project, and this section dives into the details of their responsibilities:

- **Java Packages**:

  Java encourages modular programming by grouping related classes into packages. Your image encryption project might have packages such as:
  - encryption: Handles all encryption-related tasks.
  - decryption: Focuses on decrypting images.
  - filehandler: Manages file operations like reading and writing.
  - keymanagement: Contains logic for key generation and storage.

These packages help maintain clean code and logical separation of concerns.

- **Classes**:

  Each package contains multiple classes that serve specific functions:
  - **Encryptor Class**: This class is responsible for encrypting images. It might have methods for taking an image as input and returning the encrypted image using algorithms like AES or RSA.
    - **Methods**:
      - public byte[] encryptImage(File imageFile, SecretKey key): Encrypts the image using AES encryption.
      - private byte[] applyAESAlgorithm(byte[] imageBytes, SecretKey key): Internal method that implements AES.

- **Decryptor Class**: This class focuses on reversing the encryption process to retrieve the original image.
  - **Methods**:
    - public byte[] decryptImage(File encryptedImage, SecretKey key): Decrypts the image back to its original state.
    - private byte[] applyAESDecryption(byte[] encryptedBytes, SecretKey key): Implements AES decryption.
- **FileHandler Class**: This class manages reading from and writing to files. It ensures that the image data is correctly read as byte arrays for processing.
  - **Methods**:
    - public byte[] readFile(File imageFile): Reads the image file and converts it into a byte array for encryption.
    - public void writeFile(byte[] imageData, String filePath): Writes the byte array back into an image file after encryption or decryption.
- **KeyManager Class**: This class manages key generation and storage. In the case of symmetric encryption, it generates a single key; for asymmetric encryption, it generates public-private key pairs.
  - **Methods**:
    - public SecretKey generateAESKey(): Generates an AES key for symmetric encryption.
    - public KeyPair generateRSAKeyPair(): Generates an RSA public-private key pair.

- **Methods**:

  Each method within these classes should be carefully explained in terms of:
  - **Input parameters**: What data does the method require (e.g., image file, encryption key)?
  - **Output**: What does the method return (e.g., encrypted byte array, decrypted image)?
  - **Logic**: What steps does the method take to perform its task (e.g., converting image data to bytes, applying encryption)?

CHAPTER 5
CONCLUSION

5.1 Conclusion

Reflects on the project's outcome, highlighting the success of implementing image encryption and the effectiveness of the selected algorithms. It discusses the challenges faced during development (e.g., handling large images, performance issues), and how these were addressed. It may also offer suggestions for future improvements, such as enhancing performance, adding more encryption algorithms, or optimizing the system for larger images.

APPENDIX 1

SAMPLE CODE

```
import java.sql.*;

public class DatabaseHelper {

   private static final String JDBC_URL =
"jdbc:mysql://localhost:3306/imageEncryptionDB";
   private static final String JDBC_USER = "root";
   private static final String JDBC_PASSWORD = "password";

   // Save metadata (filename and encryption key) into the database
   public static void saveMetadata(String filename, String encryptionKey) throws
SQLException {
      Connection connection = DriverManager.getConnection(JDBC_URL, JDBC_USER,
JDBC_PASSWORD);
      String query = "INSERT INTO image_metadata (filename, encryption_key) VALUES
(?, ?)";
      PreparedStatement statement = connection.prepareStatement(query);
      statement.setString(1, filename);
      statement.setString(2, encryptionKey);
      statement.executeUpdate();
      statement.close();
      connection.close();
      System.out.println("Metadata saved to the database.");
   }

   // Retrieve encryption key for a given file
   public static String getEncryptionKey(String filename) throws SQLException {
      Connection connection = DriverManager.getConnection(JDBC_URL, JDBC_USER,
JDBC_PASSWORD);
      String query = "SELECT encryption_key FROM image_metadata WHERE filename =
?";
      PreparedStatement statement = connection.prepareStatement(query);
      statement.setString(1, filename);
      ResultSet resultSet = statement.executeQuery();

      String encryptionKey = null;
      if (resultSet.next()) {
         encryptionKey = resultSet.getString("encryption_key");
      }

      resultSet.close();
      statement.close();
      connection.close();
      return encryptionKey;
   }}
```

APPENDIX 2

SCREENSHOTS

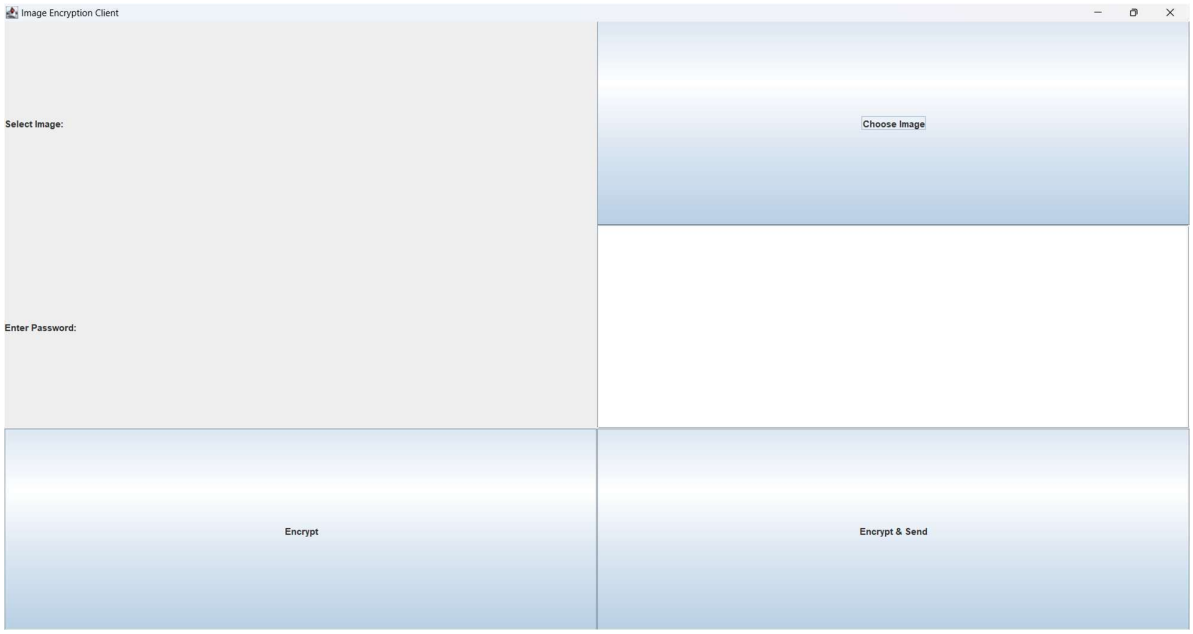

Fig A2.1: Home Screen

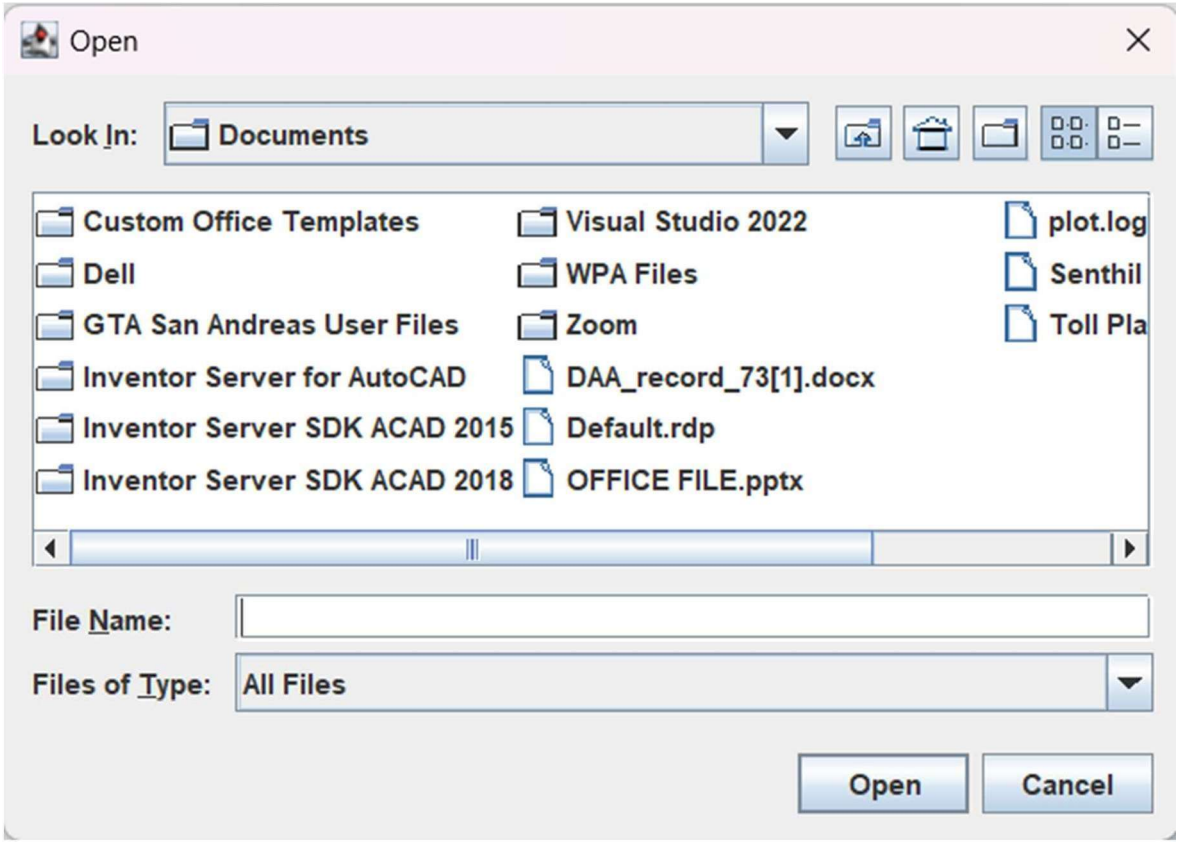

Fig A2.2: Image Selection for Encryption

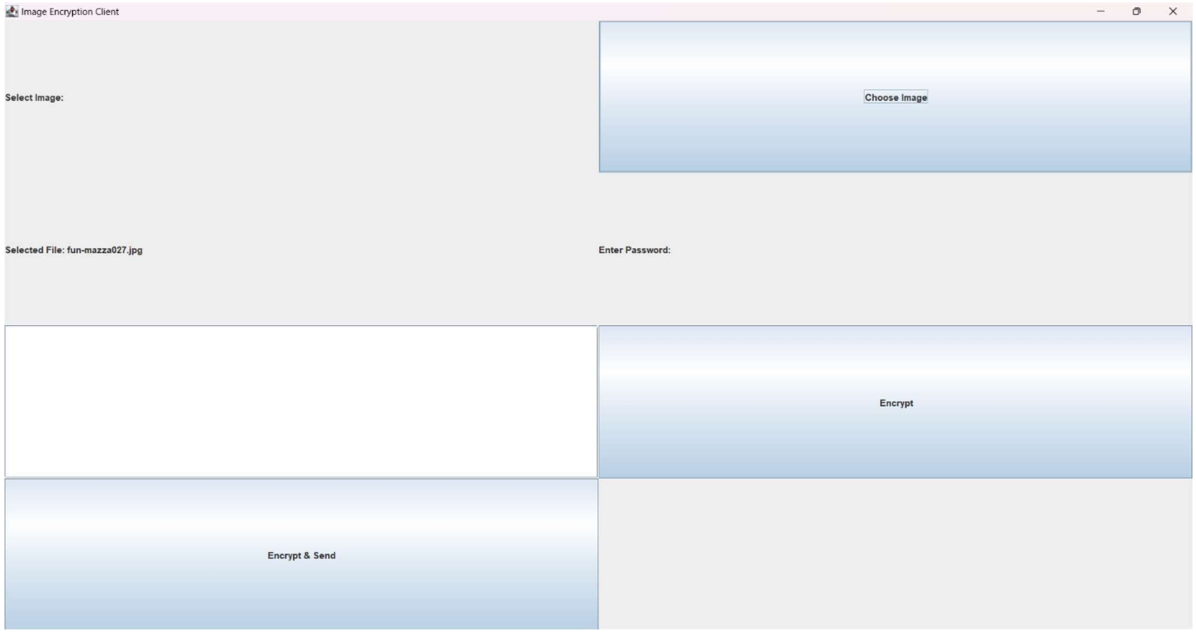

Fig A2.3: Displays File Name Before Encryption

Fig A2.4: Pop-Up Message After Encryption

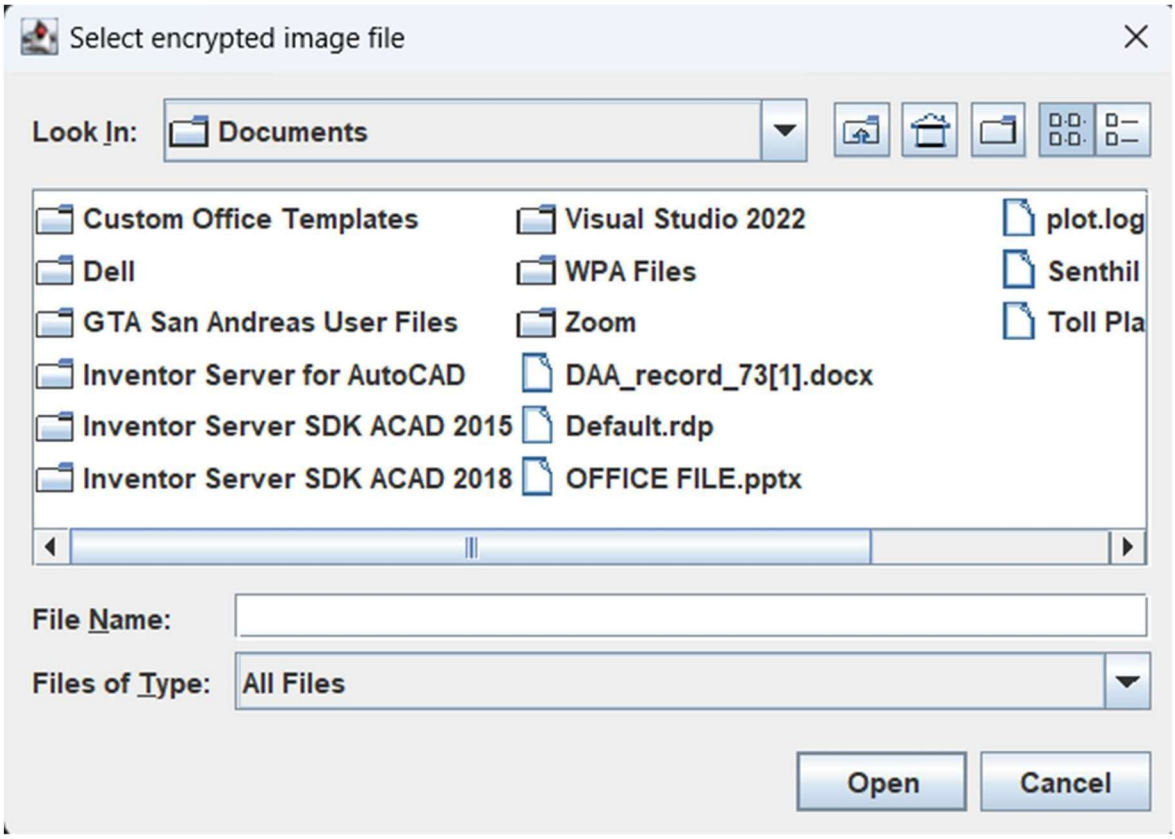

Fig A2.5: Selecting the Encrypted file For Decrypting

Fig A2.6: Enter the Decryption Key

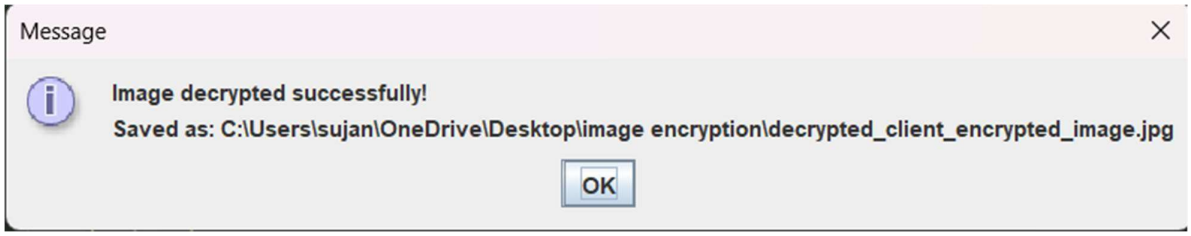

Fig A2.7: Pop-Up Message After Decryption

REFERENCES

1. Yen J. C. and Guo J. I., “A new chaotic image encryp-tion algorithm,” Proceeding of National Symposium on Telecommunications, pp. 358-362, December 1998.
2. Jui-Cheng Yen and J. I. Guo, “A New Chaotic Mirror-Like Image Encryption Algorithm and its VLSI Archi-tecture”, Pattern Recognition and Image Analysis, vol.10, no.2, pp.236-247, 2000.
3. Jui-Cheng Yen and J. I. Guo, “Efficient Hierarchical Chaotic Image Encryption Algorithm and Its VLSI Re-alization”. IEEE Proceeding Vis. Image Signal Process, vol. 147, no. 2, pp. 167-175,2000