CRYPTOGRAPHY

Encryption & Decryption

Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

Name	Registration No
Harsh Ruhela	12105121
Aryan Raj	12113281
Marvin Mehta	12109909

Supervisor

Dr. Shruti

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

November 2022

TABLE OF CONTENTS

CHAPTER1: INTRODUCTION

Cryptography, the science of encoding and decoding information to protect its confidentiality, integrity, and authenticity, is a fundamental aspect of modern information security. With the increasing reliance on digital communication and data exchange, ensuring the secure transmission and storage of sensitive information has become paramount. Cryptographic techniques, such as encryption and decryption, play a crucial role in safeguarding data from unauthorized access and tampering.

In this project, we have developed a cryptography encryption and decryption system using Java, one of the popular programming languages known for its versatility and strong support for object-oriented programming.

The goal of our project is to provide a comprehensive and user-friendly cryptography system that can be used to protect sensitive information in different applications, such as secure communication, data storage, and authentication. We have implemented a user-friendly interface that allows users to input data to be encrypted or decrypted, and display the encrypted or decrypted output. Our project also includes robust error handling and security measures to protect against various attacks, such as brute force attacks and manin-the-middle attacks.

Throughout this report, we will provide an overview of the fundamental concepts of cryptography, including encryption, decryption, and attacks on cryptographic systems. We will then discuss the implementation details of our cryptography encryption and decryption system using Java, including the design and functionality of our user interface, the cryptographic algorithms used, key management techniques employed, and security measures implemented. We will also evaluate the performance, security, and usability of our system, and highlight any challenges encountered during the development process. Finally, we will discuss the implications and future directions of our project, including potential enhancements and applications of our cryptography system.

CHAPTER2: FEATURE

Features Supported by our Project: -

1.) **File Encryption and Decryption**: Extend the functionality to allow for encryption and decryption of files, enabling users to secure files with sensitive information.

2.) **Security Analysis**: Provide an analysis of the security of the implemented cryptographic algorithms, including their strengths, weaknesses, and potential vulnerabilities.

3.) **User-Friendly Interface**: Design a user-friendly interface that allows users to input data to be encrypted or decrypted, select cryptographic algorithms and key sizes, and view the encrypted or decrypted output.

4.) **Input and Output Options**: Provide options for users to input data to be encrypted or decrypted via text input or file input, and output the encrypted or decrypted data as text or save it to a file.

5.) **Easy To Use**: Because of the GUI Made using Java Swing the interface provided is easy to use.

CHAPTER3: ADVANTAGES / DISADVANTAGES

FEW ADVANTAGES: -

1.) Enhanced Security: Cryptography is a vital technique for securing sensitive information and communications. Implementing encryption and decryption in your project can help protect data from unauthorized access, ensuring confidentiality, integrity, and authenticity.

2.) Learning Opportunity: Developing a cryptography project in Java provides an opportunity to gain in-depth knowledge and understanding of various encryption and decryption techniques, cryptographic algorithms, key management, and other security concepts. This can enhance your understanding of cryptography and security principles, which can be valuable in future projects or professional endeavors.

3.) Usability and Flexibility: A cryptography project developed using Java can be easily integrated into various applications, systems, or networks, making it usable in a wide range of contexts. The flexibility of Java as a programming language allows for easy integration with other Java-based applications or systems.

FEW DISADVANTAGES: -

1.) Complexity: Cryptography is a complex field that requires a solid understanding of mathematical concepts, algorithms, and security principles. Developing a cryptography project may involve implementing complex encryption and decryption algorithms, key management techniques, and other security measures, which can be challenging and time-consuming.

2.) Maintenance and Updates: Cryptography is a constantly evolving field, with new algorithms, attacks, and vulnerabilities discovered over time. Maintaining and updating the cryptography project to stay current with the latest developments and security best practices may require ongoing effort and resources.

CHAPTER4: SOFTWARE / HARDWARE REQUIREMENTS

SOFTWARE REQUIREMENTS: -

1.) Java Development Kit (JDK): This is the essential software development environment for Java projects, including the Java compiler, runtime libraries, and tools for building and running Java applications.

2.) Integrated Development Environment (IDE): An IDE such as Eclipse, IntelliJ IDEA, or NetBeans can provide a comprehensive development environment with features like code editing, debugging, and project management to streamline the development process.

3.) Cryptography Libraries: Java provides built-in cryptographic libraries, such as Java Cryptography Extension (JCE) and Bouncy Castle, that offer various encryption and decryption algorithms, key management functions, and other security features.

4.) Additional Libraries: Depending on the specific requirements of the project, additional libraries may be needed for tasks such as input/output operations, user interface development, or other functionality beyond the builtin Java libraries.

HARDWARE REQUIREMENTS: -

1.) Computer System: A computer system with sufficient processing power, memory, and storage to support the development and execution of Java applications. The specific hardware requirements may depend on the size and complexity of the project, as well as any performance considerations.

2.) Operating System: Java is platform-independent and can be run on various operating systems, including Windows, macOS, Linux, or others, depending on the chosen development environment and deployment environment.

CHAPTER5: METHODOLOGY

ALGO FOR TEXT ENCRYPTION

1.) Read the plaintext from the input text field labeled "t1".

2.) Create a SecretKeySpec object using the secret key (KEY) converted to bytes and the specified algorithm (ALGORITHM).

3.) Create a Cipher object instance using the specified algorithm.

4.) Initialize the cipher in encryption mode with the secret key.

5.) Convert the plaintext string into bytes using the UTF-8-character encoding.

6.) Call the doFinal method on the cipher, passing in the plaintext bytes to encrypt.

7.) Retrieve the resulting encrypted bytes.

8.) Encode the encrypted bytes into a Base64 string representation.

9.) Set the encrypted text in the output text field "t2" using the setText method.

ALGO FOR TEXT DECRYPTION

1.) Read the ciphertext from the input text field labeled "t1".

2.) Create a SecretKeySpec object using the secret key (KEY) converted to bytes and the specified algorithm (ALGORITHM).

3.) Create a Cipher object instance using the specified algorithm.

4.) Initialize the cipher in decryption mode with the secret key.

5.) Decode the ciphertext from Base64 representation using Base64.getDecoder().

6.) Perform the decryption operation using cipher. doFinal () to obtain the decrypted byte array.

7.) Convert the decrypted byte array to a string using new String (decryptedValue, StandardCharsets.UTF_8).

8.) Set the decrypted text to the output text field labeled "t2" using t2. setText(decryptedText).

9.) Handle any exceptions that may occur during the decryption process, such as InvalidKeyException, NoSuchAlgorithmException, NoSuchPaddingException, IllegalBlockSizeException, BadPaddingException, and IOException, by printing the stack trace for debugging purposes.

ALGO FOR FILE AND IMAGE ENCRYPTION

1. Read the content of the input file as a byte array using Files.readAllBytes(inputFile).

2. Create a SecretKeySpec object using the secret key (keyBytes) and the "AES" encryption algorithm.
3. Create a Cipher object instance using the "AES" encryption algorithm.
4. Initialize the cipher in encryption mode with the secret key using cipher. init (Cipher.ENCRYPT_MODE, secretKeySpec).
5. Encrypt the file content using cipher. doFinal(fileContent) and store the result in a byte array (encryptedFileContent).
6. Write the encrypted content to the output file using Files.write(outputFile, encryptedFileContent).
7. Set the label (l3) to display the name of the input file.
8. Set the status of the encryption process in the text field (t1) to "Encrypted".
9. Set the path of the output file in the text field (t2) using t2. setText(outputFilePath).
10. Handle any exceptions that may occur during the encryption process, such as InvalidKeyException, NoSuchAlgorithmException, NoSuchPaddingException, IllegalBlockSizeException, BadPaddingException, and IOException, by printing the stack trace for debugging purposes.

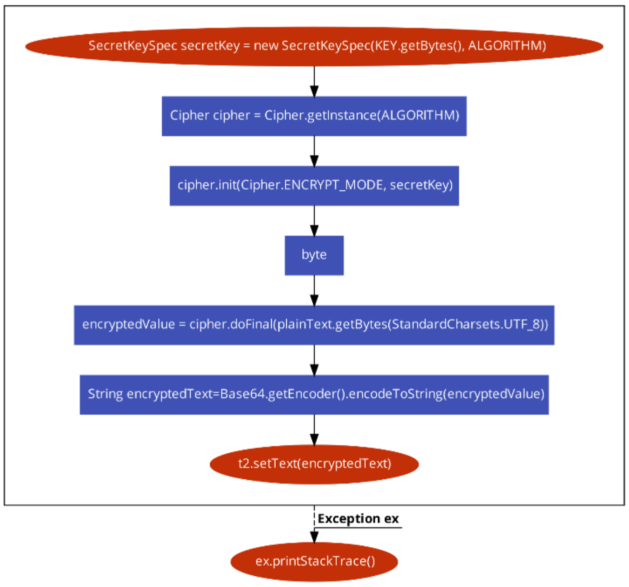
ALGO FOR FILE AND IMAGE ENCRYPTION

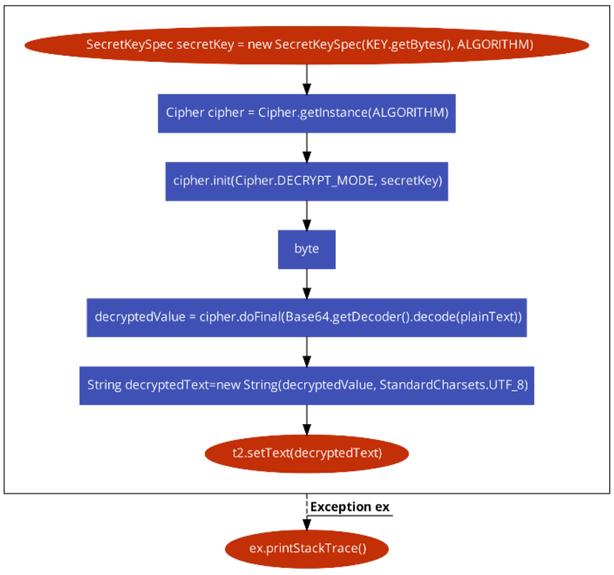
1. Read the content of the input file as a byte array using Files.readAllBytes(inputFile).
2. Create a SecretKeySpec object using the secret key (keyBytes) and the "AES" encryption algorithm.
3. Create a Cipher object instance using the "AES" encryption algorithm.
4. Initialize the cipher in decryption mode with the secret key using cipher. init (Cipher.DECRYPT_MODE, secretKeySpec).
5. Decrypt the file content using cipher. doFinal(encryptedFileContent) and store the result in a byte array (decryptedFileContent).
6. Write the decrypted content to the output file using Files.write(outputFile, decryptedFileContent).
7. Set the label (l3) to display the name of the input file.
8. Set the status of the decryption process in the text field (t1) to "Decrypted".
9. Set the path of the output file in the text field (t2) using t2. setText(outputFilePath).
10. Handle any exceptions that may occur during the decryption process, such as InvalidKeyException, NoSuchAlgorithmException, NoSuchPaddingException, IllegalBlockSizeException, BadPaddingException, and IOException, by printing the stack trace for debugging purposes.

CHAPTER6: FLOW CHART

Flowchart For Text Encryption

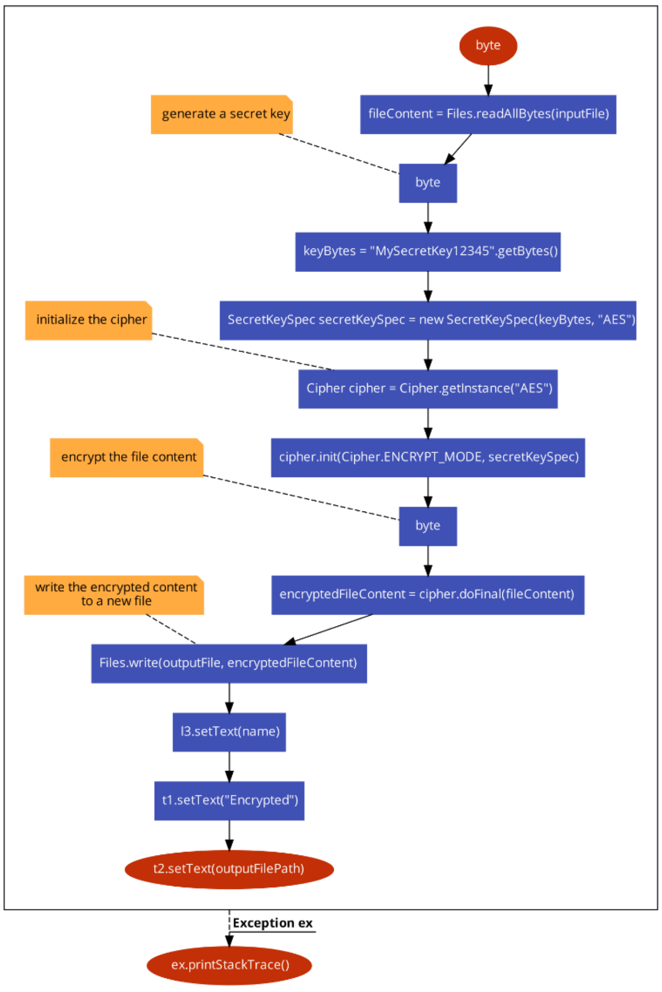
Flowchart For Text Decryption

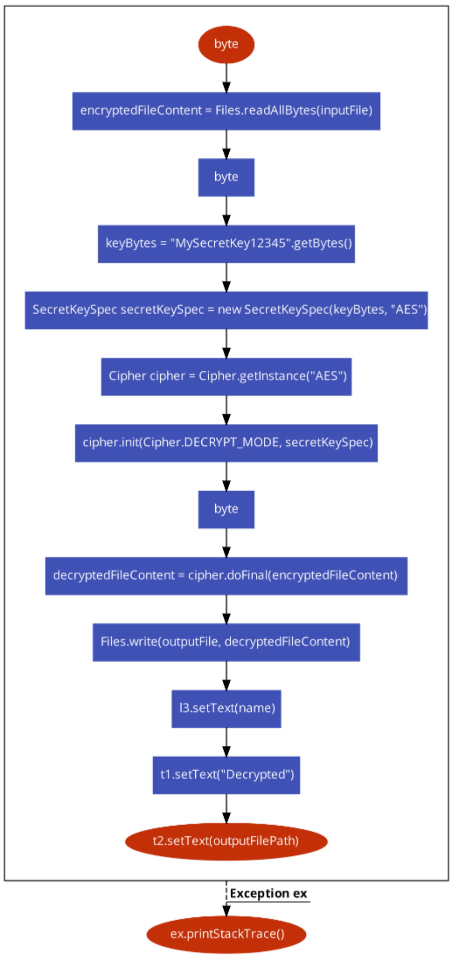

SecretKeySpec secretKey = new SecretKeySpec(KEY.getBytes(), ALGORITHM)
Cipher cipher = Cipher.getInstance(ALGORITHM)
cipher.init(Cipher.ENCRYPT_MODE, secretKey)
byte
encryptedValue = cipher.doFinal(plainText.getBytes(StandardCharsets.UTF_8))
String encryptedText=Base64.getEncoder().encodeToString(encryptedValue)
t2.setText(encryptedText)
Exception ex
ex.printStackTrace()


SecretKeySpec secretKey = new SecretKeySpec(KEY.getBytes(), ALGORITHM)
Cipher cipher = Cipher.getInstance(ALGORITHM)
cipher.init(Cipher.DECRYPT_MODE, secretKey)
byte
decryptedValue = cipher.doFinal(Base64.getDecoder().decode(plainText))
String decryptedText=new String(decryptedValue, StandardCharsets.UTF_8)
t2.setText(decryptedText)
Exception ex
ex.printStackTrace()

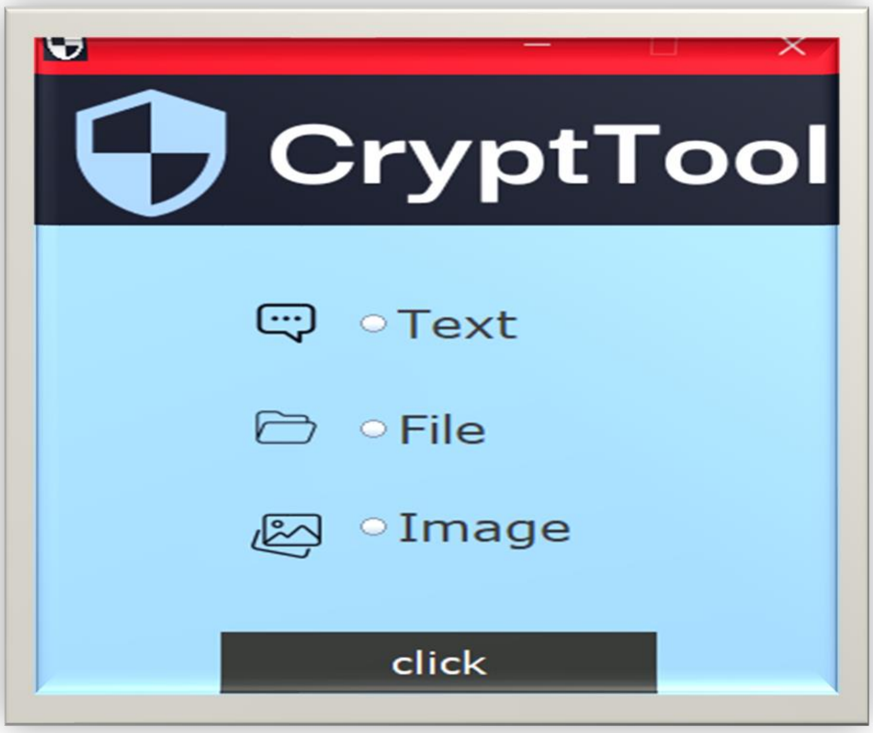
Flowchart For Image and File Encryption

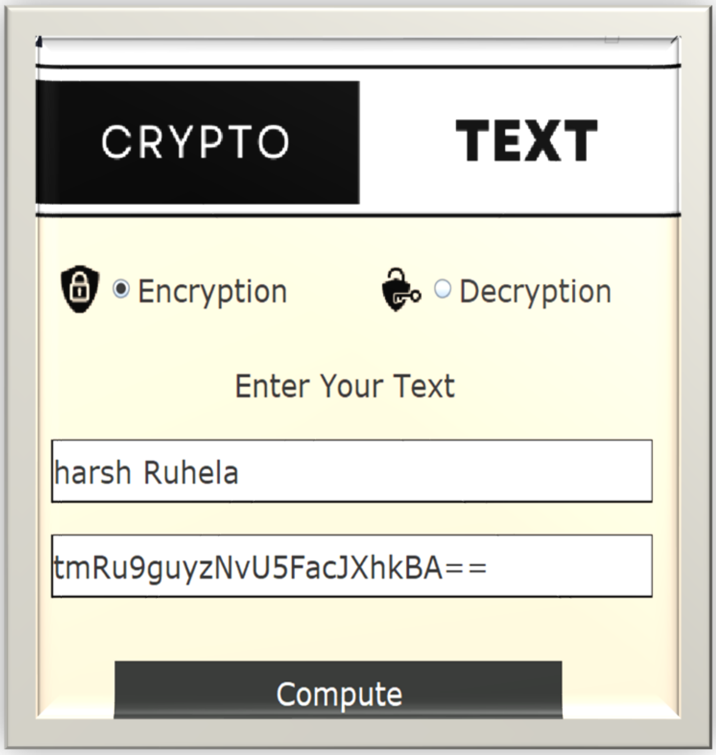
Flowchart For Image and File Decryption


byte
generate a secret key
fileContent = Files.readAllBytes(inputFile)
byte
keyBytes = "MySecretKey12345".getBytes()
initialize the cipher
SecretKeySpec secretKeySpec = new SecretKeySpec(keyBytes, "AES")
Cipher cipher = Cipher.getInstance("AES")
encrypt the file content
cipher.init(Cipher.ENCRYPT_MODE, secretKeySpec)
byte
write the encrypted content to a new file
encryptedFileContent = cipher.doFinal(fileContent)
Files.write(outputFile, encryptedFileContent)
l3.setText(name)
t1.setText("Encrypted")
t2.setText(outputFilePath)
Exception ex
ex.printStackTrace()


byte
encryptedFileContent = Files.readAllBytes(inputFile)
byte
keyBytes = "MySecretKey12345".getBytes()
SecretKeySpec secretKeySpec = new SecretKeySpec(keyBytes, "AES")
Cipher cipher = Cipher.getInstance("AES")
cipher.init(Cipher.DECRYPT_MODE, secretKeySpec)
byte
decryptedFileContent = cipher.doFinal(encryptedFileContent)
Files.write(outputFile, decryptedFileContent)
l3.setText(name)
t1.setText("Decrypted")
t2.setText(outputFilePath)
Exception ex
ex.printStackTrace()

CHAPTER7: RESULTS




CRYPTO TEXT

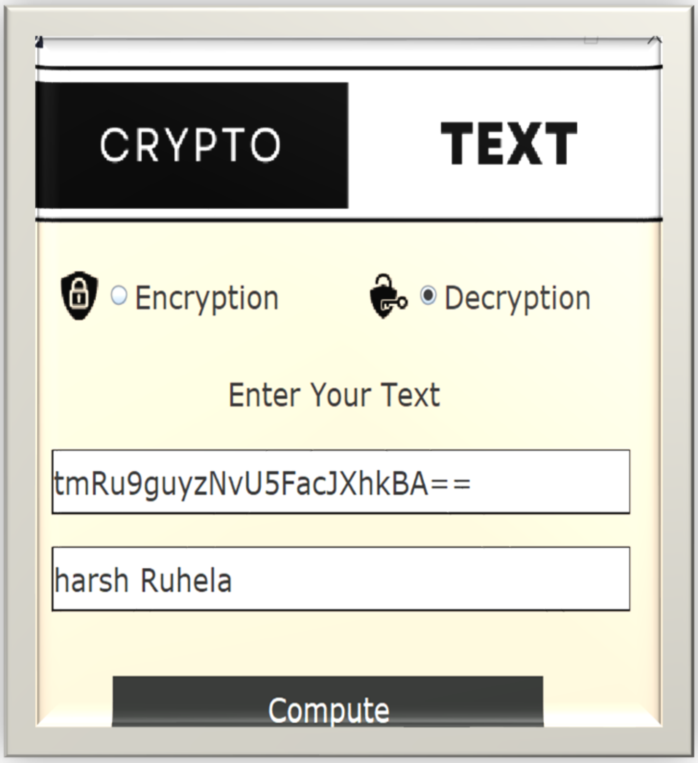
[✓ Encryption] Decryption

Enter Your Text

harsh Ruhela

tmRu9guyzNvU5FacJXhkBA==

Compute


CRYPTO TEXT

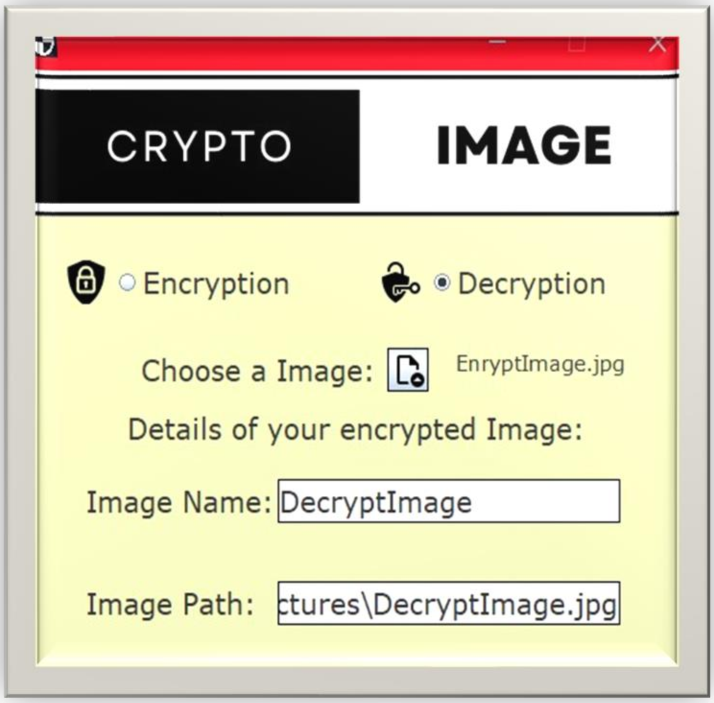
Encryption [✓ Decryption]

Enter Your Text

tmRu9guyzNvU5FacJXhkBA==

harsh Ruhela

Compute


CRYPTO IMAGE

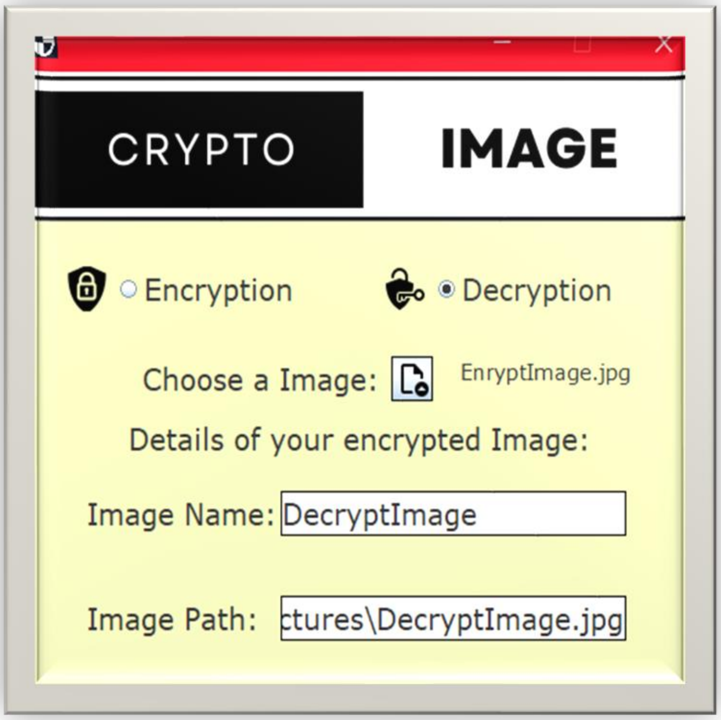
Encryption [✓ Decryption]

Choose a Image: EnryptImage.jpg

Details of your encrypted Image:

Image Name: DecryptImage

Image Path: ctures\DecryptImage.jpg


CRYPTO IMAGE

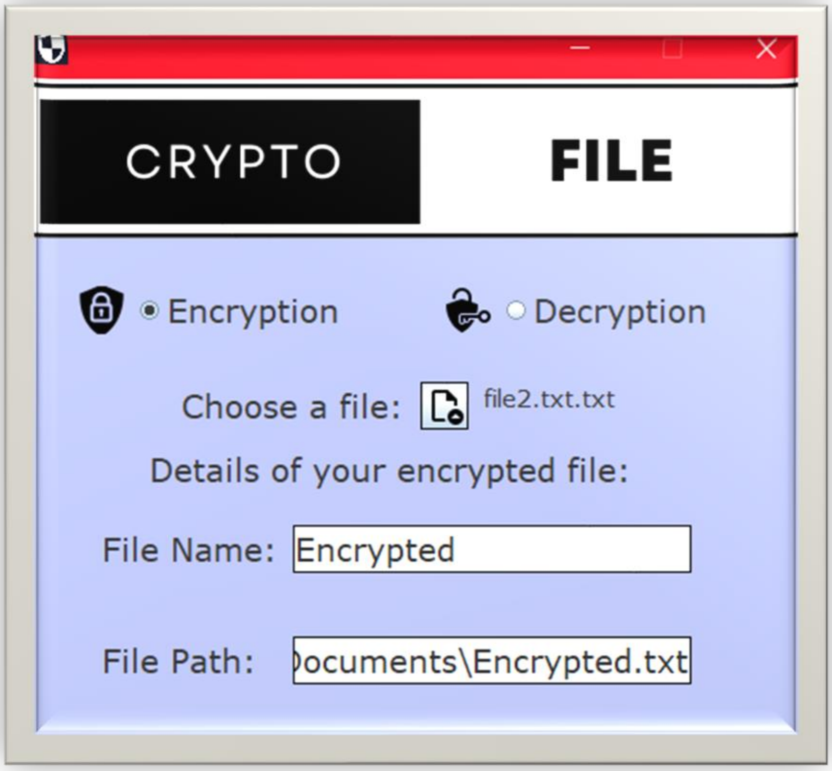
Encryption [✓ Decryption]

Choose a Image: EnryptImage.jpg

Details of your encrypted Image:

Image Name: DecryptImage

Image Path: ctures\DecryptImage.jpg


CRYPTO FILE

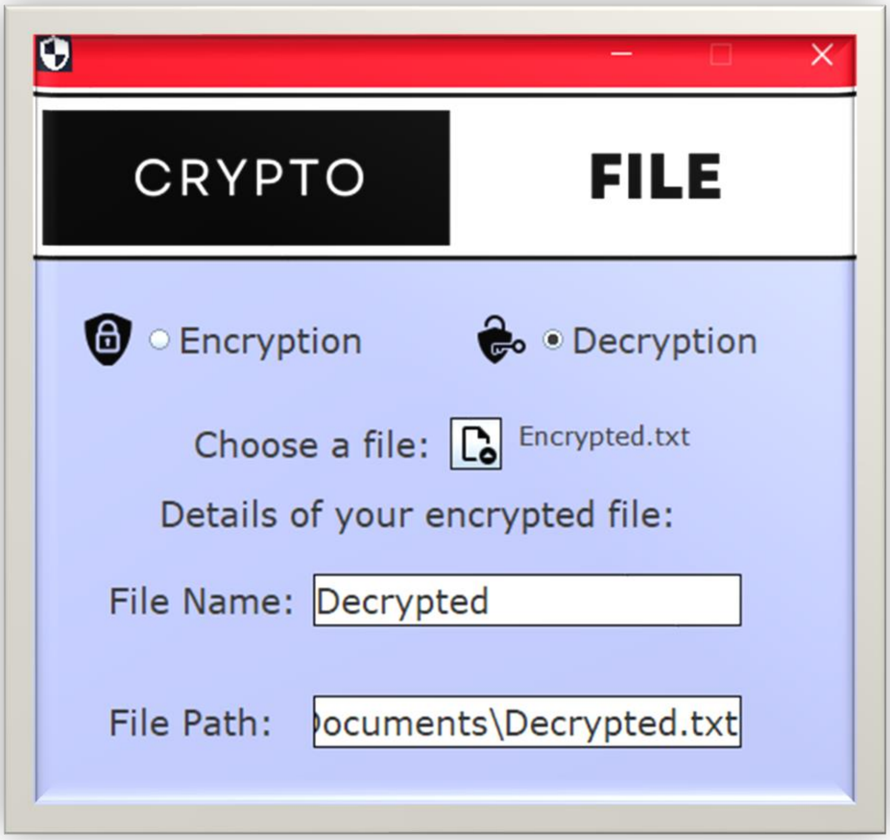
● Encryption ○ Decryption

Choose a file: file2.txt.txt

Details of your encrypted file:

File Name: Encrypted

File Path:)ocuments\Encrypted.txt


CRYPTO FILE

○ Encryption ● Decryption

Choose a file: Encrypted.txt

Details of your encrypted file:

File Name: Decrypted

File Path:)ocuments\Decrypted.txt

CHAPTER8: CONCLUSION & FUTURE SCOPE

CONCLUSION

In conclusion, our cryptography project in Java has been a successful endeavour. We have developed a robust and secure system that employs various encryption and decryption techniques to protect sensitive data and ensure secure communication. Throughout the development process, we have gained a deep understanding of cryptographic concepts, Java programming, and best practices for secure coding.

Our project has demonstrated the importance of cryptography in modern-day communication, where data privacy and security are critical concerns.

During the project, we have encountered challenges such as key management, performance optimization, and handling large data sets. We have addressed these challenges by implementing proper key management techniques, optimizing code for performance, and handling data streams efficiently.

Overall, our cryptography project has provided us with invaluable practical experience in designing and implementing secure systems using Java. We have gained a solid understanding of cryptographic concepts, Java libraries, and best practices for secure coding. We are proud of our achievements in creating a functional and secure cryptography project, and we believe that our work can serve as a strong foundation for future cryptography applications or further enhancements.

We acknowledge that cryptography is an ever-evolving field, and there may be future developments and advancements that could enhance the security of our project further. Nevertheless, we are confident that our project is a significant step towards ensuring secure communication and data protection in today's digital world. We are proud of our accomplishments and look forward to further exploring the fascinating field of cryptography in our future endeavours.

FUTURE SCOPE

1.) **Incomplete Security Measures: -** While this Project provides Encryption and Decryption ,it does not include other essential security measures such as access control, authentication and authorization.

2.) **Lack Some Feature: -** Here we have done Encryption and Decryption of a text, text file, image. But addition to that a file can be of multiple pages and can be present in any format.