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

School of Information Technology & Engineering

Network and Information Security J Component

Winter Semester 2022-23

Course Name: Network and Information Security

ITA6007 VL2022230500484 & D1+TD1

256-bit JPG Format Image Encryption Using Advanced Encryption Standard(AES) Implementation in Java Applet

Submitted by:

Gourav Joshi (22MCA0286)

Under the guidance of

Dr. Navaneethan C

(Associate Professor Senior-SITE)

1.Abstract

The American National Institute of Standards and Technology (NIST) created the popular encryption algorithm known as AES. It is one of the safest encryption algorithms currently in use, and many businesses and governments utilize it globally. The process of image encryption using AES involves several key steps. The first step is key generation, where a key is created based on a password or other input. Larger keys offer higher levels of security, with the key size options being 128, 192, or 256 bits. Once the key is generated, the image is pre-processed to convert it into a format that can be encrypted using the AES algorithm. This may involve dividing the image into blocks of 128,192 or 256 bits, padding the data to ensure it is a multiple of the block size, and other steps to ensure that the input is properly formatted for encryption. The encryption phase then begins, where the AES algorithm is applied to the input data in a block-by-block manner using the key. The AES algorithm consists of several rounds of substitution, permutation, and XOR operations, which are repeated multiple times to produce the encrypted output. The output can then be securely sent or saved and can only be unlocked with the right key. The AES method is performed backward to the encrypted data during the decryption stage, using the same key to create the original image data.In this project we demonstrate Image encryption using AES which provides a high level of security and confidentiality for digital images, making it suitable for a wide range of applications, such as military and government communications, online banking, and digital media distribution.

2. Keywords

Advanced Encryption Standard (AES), Symmetric Block Cipher, Key Generation

Encryption, Decryption, Confidentiality, Digital Images

3.Introduction

This project aims to develop a user-friendly Java Applet that implements the encryption and decryption of JPG format images using the robust 256-bit Advanced Encryption Standard (AES) algorithm. The motivation is to address the need for secure data transmission and storage, especially for sensitive visual information. Existing image encryption systems suffer from limitations such as weak encryption strength, compatibility issues with JPG format, lack of user-friendliness, and inefficiency. To overcome these issues, this project will provide a highly secure AES-based encryption system compatible with JPG format. The system will feature a user-friendly interface that simplifies the encryption and decryption process, making it accessible to users with minimal technical expertise.

Efficiency is also a priority, and the AES algorithm, known for its efficiency and effectiveness, will be employed to ensure optimal performance without compromising security. By addressing these limitations, this project aims to enhance the privacy and integrity of sensitive image data.

The proposed Java Applet will facilitate secure data transmission and storage, benefiting applications such as medical imaging, government communications, and personal image sharing. By providing enhanced security, compatibility, user-friendliness, and efficiency, this project aims to contribute to the advancement of secure image encryption techniques and safeguard sensitive visual information in various domains.

4. Literature Survey:

Ref No.	Title	Publisher	Methodology
1.	Image encryption: Using AES, feature extraction and random no. generation.	2015 4th International Conference on Reliability, Infocom Technologies and Optimization	This paper proposes a technique combining 128-bit AES encryption, feature extraction, and random number generation for secure image data transmission. The system utilizes AES encryption in two levels and feature-based key generation to enhance confidentiality and security against attacks. However, the feature extraction process demands significant computational resources, impacting performance. Dual-level AES encryption may increase encryption time, delaying transmission. Poorly designed feature extraction algorithms may lead to weak keys exploitable by attackers. Additional storage for features and keys may raise implementation costs.
2.	A novel image encryption algorit hm using AES and visual cryptography.	Proceedings on 2016 2nd International Conference on Next Generation Computing Technologies, NGCT 2016,	This paper discusses image encryption approaches: AES, Visual Cryptography, chaotic theory-based algorithms, and a proposed algorithm securing AES key using Visual Secret Sharing. AES and Visual Cryptography lack key security, but proposed algorithm offers better security and hardware compatibility. Challenges include feature extraction computation and algorithm weaknesses. The proposed algorithm ensures confidentiality,

			authenticity, and integrity but may require more storage and longer encryption times, affecting performance.
3.	Image Data Encry ption Using des M ethod.	Conference on Computer Science and Artificial Intelligence, ICCSAI 2021.	This paper explores the usage of Data Encryption Standard (DES) for image encryption, along with the development of a Java-based DES encryption application. DES is a symmetric block cipher algorithm that ensures data confidentiality. However, DES requires an 8-byte or 8-character key length, and weaknesses such as third-party suspicions upon opening encrypted images and key leakage due to key reuse exist. These challenges emphasize the importance of stronger encryption methods and secure key management practices for improved security.
4.	Development of platform using NIOS II soft core processor for image encryption and decryption using AES algorithm	2015 International Conference on Communication and Signal Processing, ICCSP 2015	In order to improve picture security and encryption performance, the W7 key stream generator is added to the AES algorithm in this research study. On a single core NIOS II system with a Cyclone II FPGA board, the suggested AES algorithm is implemented. The article discusses the NIOS II processor's characteristics, platform development, a description of the AES algorithm, and results. Although the suggested technique intends to secure picture data transfer and enhance AES performance, difficulties with hardware implementation and other restrictions exist. Benefits include improved encryption and security performance.
5.	Image Encryption using an Image Pa ttern based on Advanced Encrypt ion Standard	2021 IEEE Colombian Conference on Communications and Computing, COLCOM 2021	This research proposes an enhanced AES algorithm for image encryption using an image as a key. The algorithm generates a state matrix and performs exclusive or sum operations with the key. This method improves security and performance compared to traditional key text methods. The use of image keys provides high-level security, but choosing an appropriate key image is a challenge. The application is for secure image encryption, offering high-level security and improved performance.

6.	Proposing a novel Dynamic AES for image encryption using a chaotic map key management approach	Elsevier GmbH 2021	The Advanced Encryption Standard (AES) and the logistic chaotic map are the foundations of the new method for picture encryption proposed in this study, called Dynamic AES. The proposed algorithm is tested against statistical and differential attacks and found to be superior to many other image encryption algorithms. The challenges may include implementation complexity, while the benefits include improved security. The application is secure transmission and storage of image data.
7.	Improving image e ncryption using tw o-dimensional logistic map and AES	International Conference on Communication and Signal Processing, ICCSP 2016	In this paper, a double encryption method employing the 2D logistic map and AES is proposed. The first round of encryption is performed using a 2D logistic map, while the second round is performed using AES. On a colour image with the red, green, and blue planes individually encrypted, the suggested technique is shown in action. The 2D logistic map's expanded control settings and key spaces make it more challenging for nefarious parties to forecast the secret content. The suggested method offers improved image encryption security;however the twofold encryption procedure may increase computing complexity and length of time needed for encryption and decoding.

8.	Image Encryption and Analysis using Dynamic AES	IEEE, 2019 5th International Conference on Optimization and Applications (ICOA)	A key-dependent dynamic S-Box is incorporated into the proposed dynamic AES algorithm to increase security. A dynamic irreducible polynomial and an affine constant based on a secret key are used to produce a dynamic S-Box. On both grayscale and colour images, the system is assessed using metrics such Image Histogram Analysis, Adjacent Pixel Correlation Analysis, Image Entropy, NPCR, and UACI. The results show that the dynamic AES algorithm performs better than the traditional AES technique. Images encrypted with dynamic AES are very secure, preventing hacking and unauthorised access. However, dynamic AES encryption necessitates more computer power, resulting in slower processing

			times and possible difficulties in image interpretation and transmission.
9.	Digital Image Encryption Based on Advanced Encryption Standard (AES)	IEEE, 2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)	This study implements AES encryption and decryption in MATLAB, defining functions for S-box, byte replacement, displacement, mixed column transformation, and key transformation, along with their inverse transformations. It proposes an image encryption approach using AES and key control in MATLAB, leveraging its computational capabilities. However, the processing demands of AES encryption can slow down encryption and decryption, especially for large images. Proper key management is crucial for secure image encryption, but it may pose challenges in installation and maintenance.
10.	A fast image encryption scheme based on AES	IEEE, 2017 2nd International Conference on Image, Vision and Computing (ICIVC)	This study presents an image cryptosystem based on AES-CBC mode, aiming to enhance security and encryption speed. The AES algorithm is implemented using a look-up table mechanism for faster encryption. The paper analyses the limitations of chaotic systems for image encryption due to data magnitude and redundancy. It introduces an innovative image encryption algorithm combining AES and Visual Cryptography, potentially offering high-level security for network image transmission. However, a comprehensive comparison with other encryption algorithms is lacking for performance evaluation.
11.	Text to image encryption technique using RGB substitution and AES	IEEE, Published in 2017 International Conference on Inventive Computing and Informatics (ICICI)	The suggested system is an Android app that converts text into an image using RGB substitution and then encrypts the resultant image using the AES encryption method. By delivering the secret key and the cypher text together in a single transmission, the technique circumvents the key exchange issue. The combinational number used for text to image transformation is stored in an additional pixel that is appended to the encrypted image. The message is decrypted at the receiver's end by using the reverse process. The system has poor robustness and is susceptible to visual attacks. The quantity of pixels in the image also has an impact on how much text may be included there.

12.	Image Encryption Based on AES Key Expansion	IEEE, 2011 Second International Conference on Emerging Applications of Information Technology	This study presents a new picture encryption technique based on AES Key Expansion, offering high security with reduced processing resources. The encryption process involves bitwise exclusive OR between sets of picture pixels and unique 128-bit keys. Independent key generation using AES Key Expansion is performed at the sender and receiver sides. Evaluation using benchmark images shows its effectiveness. The algorithm requires less computational power, making it suitable for real-time image encryption in applications like video conferencing. However, key management can pose challenges, particularly in large-scale user applications.
13.	Image Encryption Based on AES and RSA Algorithms	IEEE, 2020 3rd International Conference on Computer Applications & Information Security (ICCAIS)	This paper compares AES and RSA algorithms for digital image encryption, evaluating encryption quality, histogram readings, and correlation coefficient using MATLAB. The methodology involved encrypting and decrypting images with both algorithms and analysing the results. AES exhibited faster encryption and decryption times, while RSA showed better image quality and correlation coefficient. The study suggests that the choice of encryption algorithm should be based on user-specific requirements and preferences.
14.	Enhanced Arnold’s Cat Map-AES Encryption Technique for Medical Images	IEEE, 2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)	This paper presents the ECAT-AES-131 encryption technique for medical images, combining a pre-processing phase, modified Arnold Cat map, and AES-128 encryption. A comparative study with three algorithms and recent chaotic-based techniques demonstrates the proposed approach's robustness using medical imaging datasets. The results show enhanced encryption strength and image quality, with lower computational cost compared to AES and CAT-AES methods.
15.	Feature Based Encryption Technique for Securing Forensic Biometric Image	Publisher: IEEE, 2014 2nd International Conference on Artificial	This research presents an image encryption method using visual cryptography and AES-256 algorithm. AES constructs the encryption key based on extracted image features. Rijndael's symmetric block cipher handles data in 128-bit blocks,

	Data Using AES and Visual Cryptography	Intelligence, Modelling and Simulation	supporting key sizes of 128, 192, and 256 bits. The n-share visual cryptography encrypts pixel values without loss. AES ensures diffusion and confusion, maintaining image quality after decryption. However, this technique's applicability may be limited to securing forensic biometric image data, not suitable for other data types or applications.

16.	Light Weight Encryption for Medical Images	IEEE, 2016 26th International Conference on Computer Theory and Applications (ICCTA)	This paper proposes a hybrid technique for efficient encryption and transmission of medical images over networks. The technique involves encrypting the region of interest (ROI) using AES, followed by lossless compression for transmission. Non-ROI regions are lossy compressed for higher compression ratios. At the receiving end, the ROI is decrypted and fused with the decompressed NON-ROI to reconstruct the original image. The paper provides related work, concepts, simulation results, and analysis of the proposed method.
17.	An Advanced Assertion and Refurbishment of Images through AES	IEEE, 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)	This paper addresses the need to secure multimedia content, particularly images, in the context of increasing digital content sharing on social media. Techniques such as cryptography, steganography, and watermarking are employed for multimedia security, with a focus on digital watermarking for copyright protection and authorized usage. The embedded watermark should be imperceptible yet robust to prevent manipulation. Robust and secure watermarking policies are essential to safeguard digital content and prevent unauthorized use.
18.	Web Authentication Security Using Image Steganography and AES Encryption	IEEE, 2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC)	This study presents a highly secure web-based authentication solution using image steganography and the 128-bit AES algorithm. The method pairs a facial identification photo as image with AES encryption, providing robust security. The proposed approach outperforms steganalysis attacks and is recommended for future online applications handling sensitive user information, addressing various security issues.
19.	Comparative	IEEE, 2017	This paper explores the role of cryptography in

	study on AES and RSA algorithm for medical images	International Conference on Communication and Signal Processing (ICCSP)	network security, highlighting the importance of protecting data during transmission. It compares RSA and AES techniques for ensuring data integrity and discusses the hardware devices used in networking. The significance of data integrity in the medical field is emphasized. The proposed method compares RSA and AES, analysis simulation results, and concludes with implications for future research.
20.	Investigation of Fault Propagation in Encryption of Satellite Images Using the AES Algorithm	MILCOM 2006 - 2006 IEEE Military Communications conference	This work extensively explores popular AES modes (ECB, CBC, OFB, CFB, and CTR) along with their advantages and disadvantages. Analysis is conducted on the impact of on-board defects, particularly SEUs, during encryption. The study also investigates transmission failures caused by noise in satellite channels. It is noted that defects during gearbox can propagate to multiple blocks, but OFB and CTR modes limit error propagation, making them more suitable for noisy channels.

5. AES Overview:

The symmetric encryption method known as AES, or Advanced Encryption Standard, is one that is frequently employed to protect sensitive data. The Data Encryption Standard (DES), which had been used for a while, was replaced by it in 2001 by the U.S. National Institute of Standards and Technology (NIST). AES can be used with keys that are 128, 192, or 256 bits long and operates on fixed-size blocks of data, which are commonly 128 bits. The same key is utilised in AES encryption for both encryption and decryption. The algorithm is made up of numerous rounds of transformation that include mixing, permutation, and substitution operations. These rounds make sure that the initial unencrypted data is properly jumbled and hidden, making it challenging for unauthorised parties to identify any patterns and recover the original data. AES uses key exchange for encryption.

5.1 Block Diagram of AES Encryption and Description:

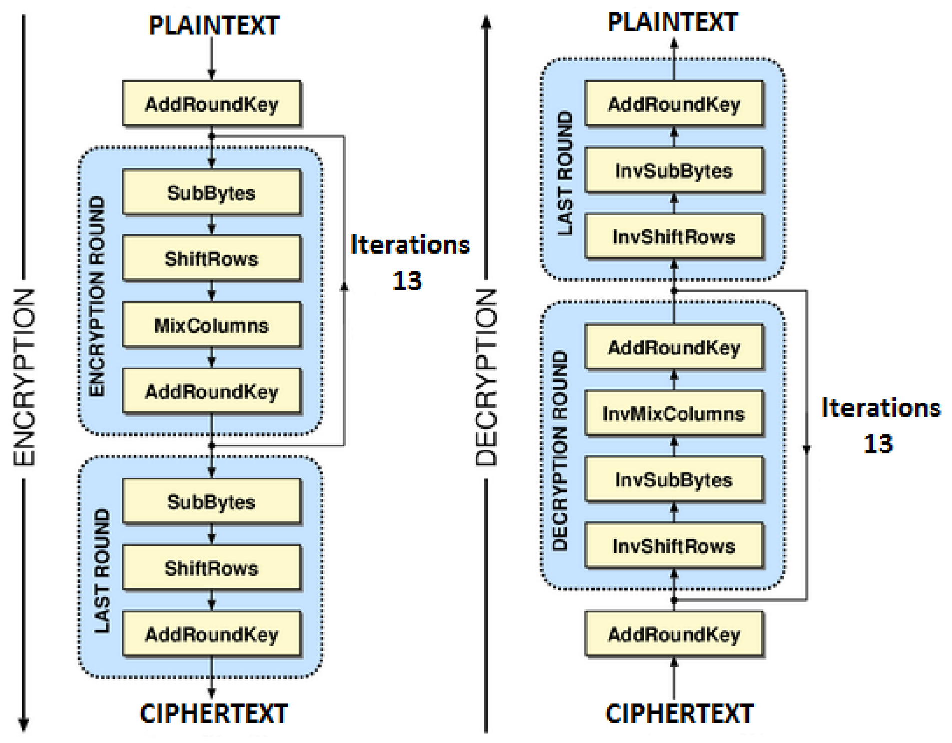


Figure 1

Figure 1: shows the various steps performed in each round.

Step 1: Each byte of the state array is replaced with a corresponding byte from a substitution box (S-Box) during encryption in the procedure known as SubBytes (Byte Substitution). By splitting each bit into two pieces and mapping the row and column values to the S-Box, a new byte value is produced. SubBytes is the name of this procedure. Every byte is replaced with the inverse value from the S-Box during the inverse transformation known as InvSubBytes that is carried out during decryption.

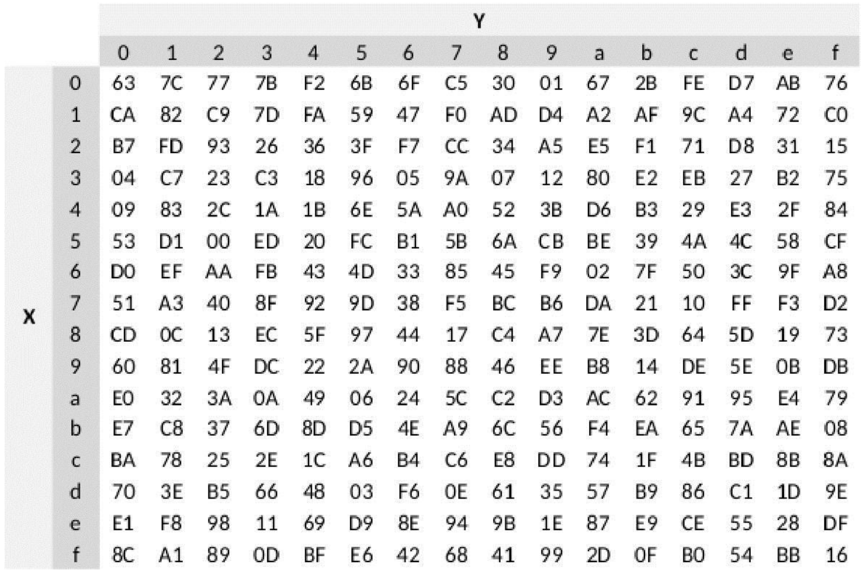
Step 2: ShiftRows (Row Shifting): The bytes in each row of the state array are cycled around in this phase. The second row is moved one place to the left, the third row two positions to the left, and the fourth row three positions to the left during encryption. ShiftRows is the name of this process. To restore the original row positions, the inverse transformation known as InvShiftRows is carried out during decryption.

Step 3: The bytes in each column of the state array are blended in this phase via a mathematical technique known as matrix multiplication. The next column in the state array is created during encryption by multiplying each column by a predetermined matrix. The name of this process is MixColumns. This step is not carried out in the final round. The InvMixColumns inverse transformation is used during decryption to undo column mixing and retrieve the initial state array.

Step 4AddRoundKey (Key Mixing): In this stage, an XOR operation is used to combine a round key and the state array. The round key, which is added to the state array during encryption, is obtained from the primary encryption key. The name of this process is AddRoundKey. In order to reverse the XOR operation and get the original state array during decryption, the inverse transformation known as InvAddRoundKey is carried out.

These steps are repeated for multiple rounds (10, 12, or 14 rounds depending on the key size) in the AES encryption process, with each round involving SubBytes, ShiftRows, MixColumns (except the last round), and AddRoundKey transformations. The order of the steps ensures that the original plaintext is thoroughly mixed and obscured, making it difficult for unauthorized individuals to decrypt the data without the correct key.

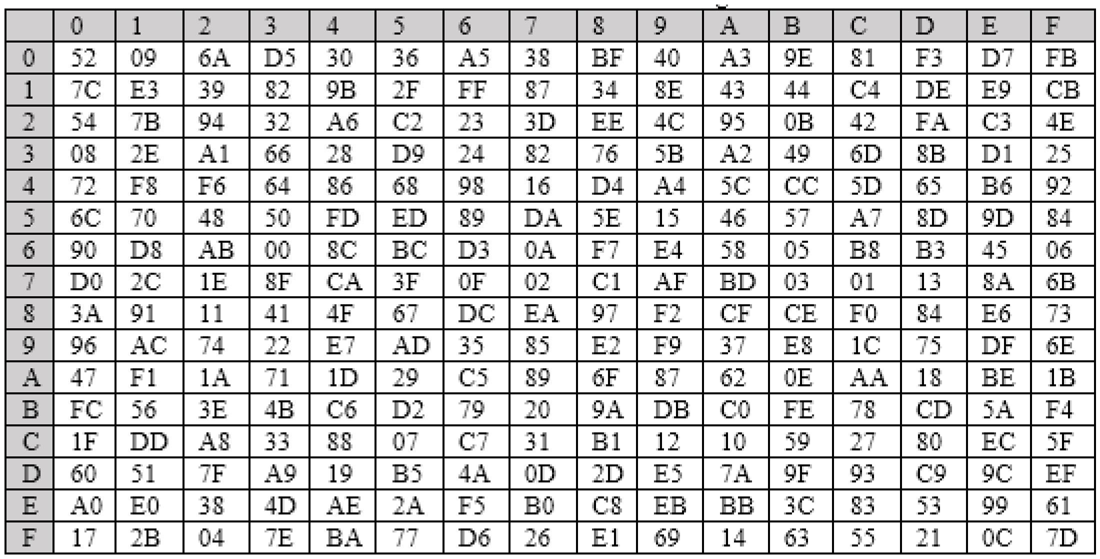
Figure 2:Represent the S-Box table for AES Encryption and Description.



		Y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
X	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	a	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	b	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	c	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	d	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	e	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	f	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Figure 2

Figure3: Represent the Inverse S-Box table for AES Encryption and Decryption.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	82	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	89	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	84	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	87	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	38	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Figure 3

6.Proposed Work:

In this project we perform image encryption using Advanced Encryption Algorithm(256 bit) with the help of java programming.

Our Work Plan divided into following sections:

Image encryption using AES with the help of Java can be achieved using the Java Cryptography Extension (JCE) library. Here are the steps to implement image encryption using AES in Java:

- Import the required libraries:
- Load the image file into a byte array.
- Generate a secret key(256 bit) for AES encryption.
- Initialize the AES cipher with the secret key.
- Encrypt the image data.

- Save the encrypted data to a file.
- To decrypt the image data, initialize the cipher in decrypt mode and use the same secret key.
- Save the decrypted data to a file.

7. Project Architecture:

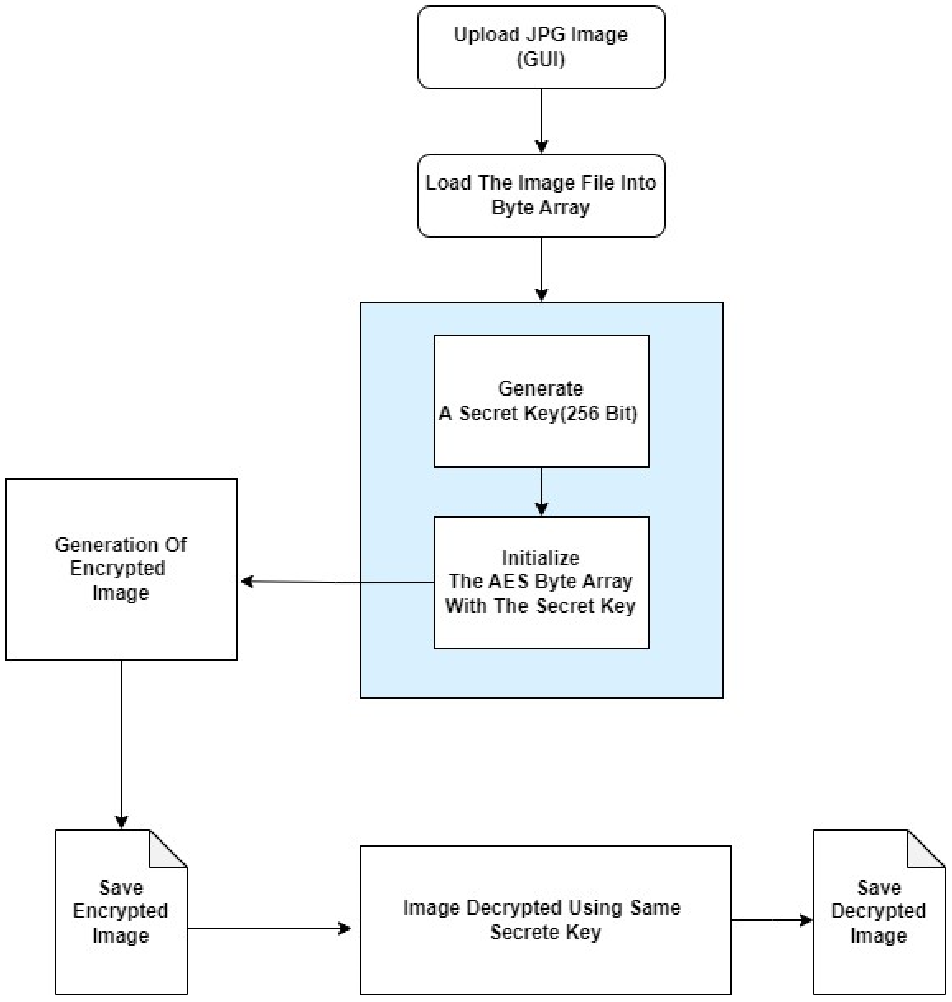

Upload JPG Image (GUI)
Load The Image File Into Byte Array
Generate A Secret Key(256 Bit)
Initialize The AES Byte Array With The Secret Key
Generation Of Encrypted Image
Save Encrypted Image
Image Decrypted Using Same Secrete Key
Save Decrypted Image

Figure 4

8. Implementation Detail:

For our project on image encryption using the AES algorithm, we have chosen the Java programming language for implementation. Java is a highly versatile and powerful language that offers extensive library support, making it well-suited for handling complex projects. To facilitate our development process, we are utilizing Visual Studio Code, a popular integrated development environment (IDE) that provides a user-friendly interface and efficient tools for code execution and debugging. With Java's robust features and Visual Studio Code's capabilities, we have a solid foundation to implement our image encryption project effectively.

8.1 Advantage of Java for this project:

Portability: Java is a platform-independent language, meaning that code written in Java can run on any operating system with a Java Virtual Machine (JVM). This portability ensures that the image encryption software can be easily executed on different platforms without requiring extensive modifications.

Rich Library Support: Java provides a vast array of libraries and APIs that are well-suited for cryptographic operations, including AES encryption. Libraries like Java Cryptography Architecture (JCA) and Java Cryptography Extension (JCE) offer comprehensive support for implementing AES algorithms, generating secure keys, and performing encryption and decryption operations.

Strong Encryption Support: Java supports AES encryption out-of-the-box through its cryptography libraries. It offers various AES modes, including ECB (Electronic Codebook), CBC (Cipher Block Chaining), and CTR (Counter), allowing developers to choose the most suitable mode for image encryption.

Multi-threading and Performance: Java support multi-threading, enabling concurrent execution of image encryption operations. This can significantly improve performance, especially when processing multiple images simultaneously. Additionally, Java's Just-In-Time (JIT) compilation and optimization techniques contribute to efficient execution, ensuring reasonable processing times for image encryption.

8.2 Software Project Modules:

The step of implementation is when the theoretical design is translated into a programmatically based approach. The application will be divided into several components at this point and then coded for deployment. In order to demonstrate the effectiveness of this suggested protocol, we developed the recommended notion using the Java programming language as the preferred language. Three key modules make up the proposed application, including:

I. Data Encryption Module
II. Select the Properties Module.
III. Data Decryption Module.

8.2.1 Data Encryption Module:

In this module of project, the user can select the image file which he wants to encrypt and select the location where the files are stored after encryption.

8.2.2 Select the Properties Module:

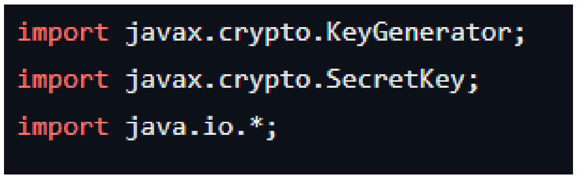
In this module If the image file is not very significant, the user can choose a customised four-round encryption, and if the file is important and contains sensitive information, the user can choose a 14-round encryption method.**8.2.3**

8.2.3 Data Decryption Module:

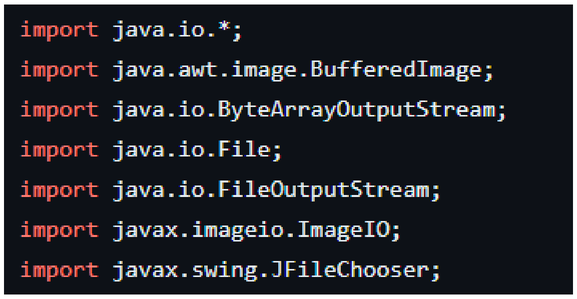
The Data Decryption Module allows users to decrypt encrypted files and retrieve the original image.The module incorporates error handling, optimizing performance for faster decryption. Users can decrypt their files and obtain the original image seamlessly.

8.3 Sample Code:

8.3.1 Import Important Java Libraries :



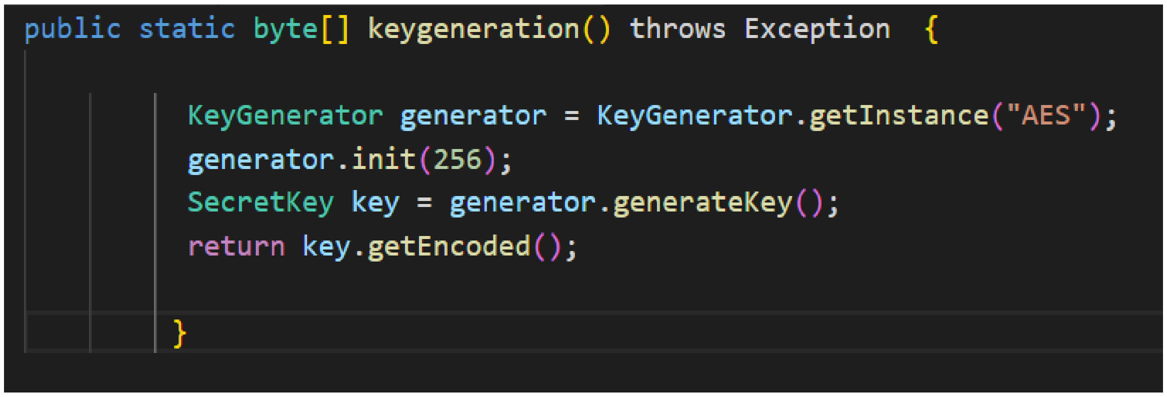
```java
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import java.io.*;
```



```java
import java.io.*;
import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import javax.imageio.ImageIO;
import javax.swing.JFileChooser;
```

Figure 5

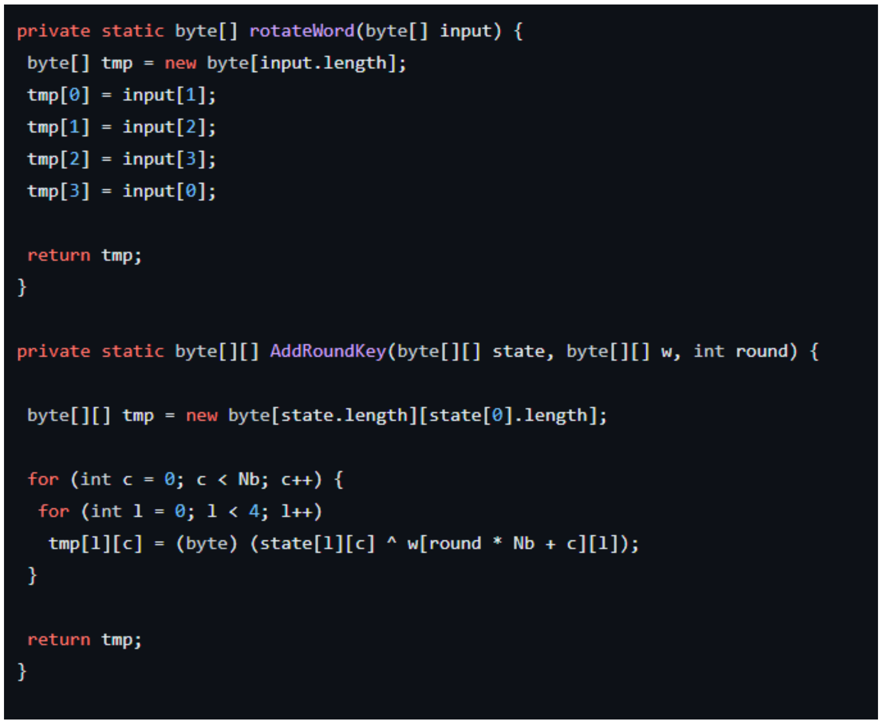
8.3.2 256 Bit Key Generation using Key Generator Class



```java
public static byte[] keygeneration() throws Exception  {

        KeyGenerator generator = KeyGenerator.getInstance("AES");
        generator.init(256);
        SecretKey key = generator.generateKey();
        return key.getEncoded();

    }
```

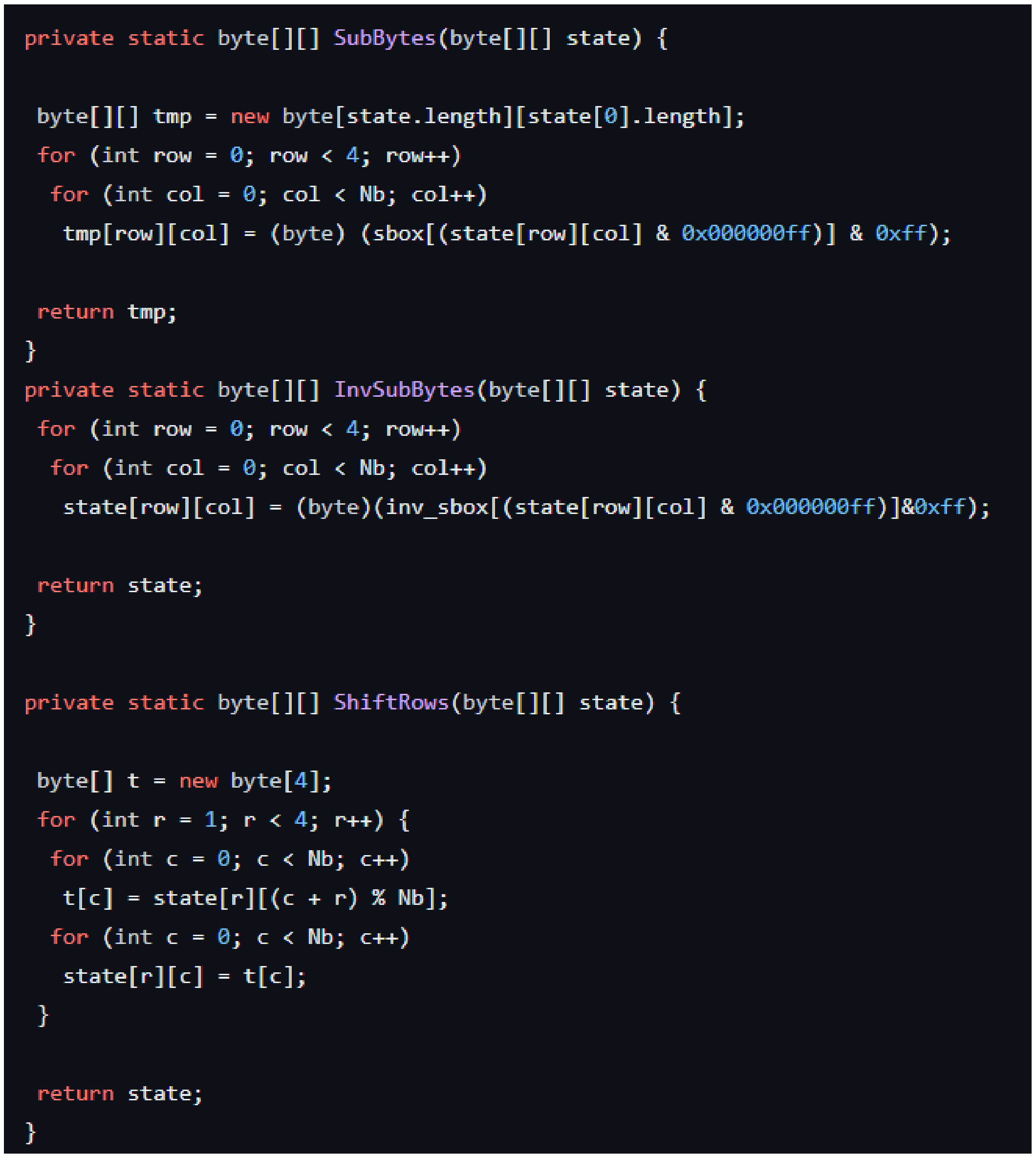
Figure 6

8.3.3 Method forRotateword, AddRoundKey, SubBytes, InverseSubBytes, ShiftRows, MixColumn, InverseMixColumn,



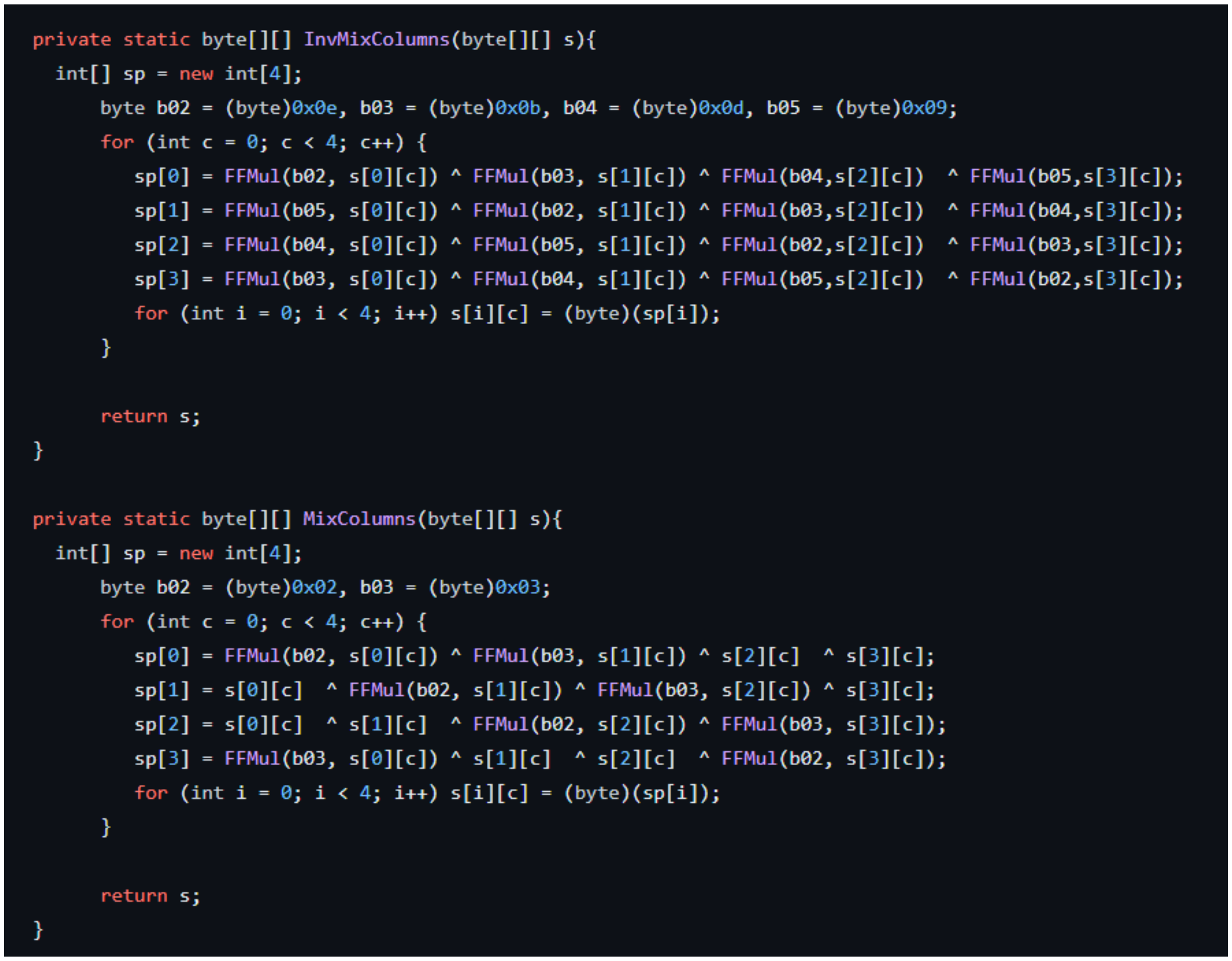
```java
private static byte[] rotateWord(byte[] input) {
 byte[] tmp = new byte[input.length];
 tmp[0] = input[1];
 tmp[1] = input[2];
 tmp[2] = input[3];
 tmp[3] = input[0];

 return tmp;
}

private static byte[][] AddRoundKey(byte[][] state, byte[][] w, int round) {

 byte[][] tmp = new byte[state.length][state[0].length];

 for (int c = 0; c < Nb; c++) {
  for (int l = 0; l < 4; l++)
   tmp[l][c] = (byte) (state[l][c] ^ w[round * Nb + c][l]);
 }

 return tmp;
}
```

Figure 7



```
private static byte[][] SubBytes(byte[][] state) {

 byte[][] tmp = new byte[state.length][state[0].length];
 for (int row = 0; row < 4; row++)
  for (int col = 0; col < Nb; col++)
   tmp[row][col] = (byte) (sbox[(state[row][col] & 0x000000ff)] & 0xff);

 return tmp;
}
private static byte[][] InvSubBytes(byte[][] state) {
 for (int row = 0; row < 4; row++)
  for (int col = 0; col < Nb; col++)
   state[row][col] = (byte)(inv_sbox[(state[row][col] & 0x000000ff)]&0xff);

 return state;
}

private static byte[][] ShiftRows(byte[][] state) {

 byte[] t = new byte[4];
 for (int r = 1; r < 4; r++) {
  for (int c = 0; c < Nb; c++)
   t[c] = state[r][(c + r) % Nb];
  for (int c = 0; c < Nb; c++)
   state[r][c] = t[c];
 }

 return state;
}
```

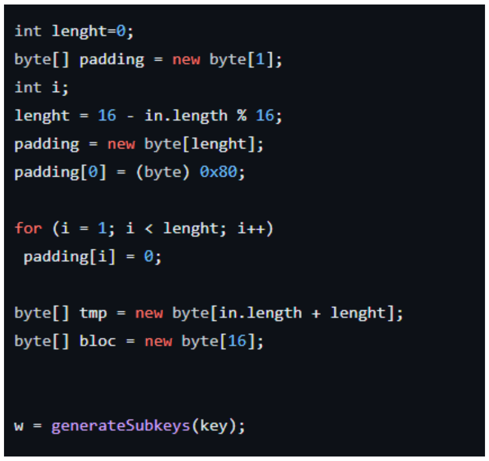
Figure 8



```
private static byte[][] InvMixColumns(byte[][] s){
  int[] sp = new int[4];
      byte b02 = (byte)0x0e, b03 = (byte)0x0b, b04 = (byte)0x0d, b05 = (byte)0x09;
      for (int c = 0; c < 4; c++) {
         sp[0] = FFMul(b02, s[0][c]) ^ FFMul(b03, s[1][c]) ^ FFMul(b04,s[2][c])  ^ FFMul(b05,s[3][c]);
         sp[1] = FFMul(b05, s[0][c]) ^ FFMul(b02, s[1][c]) ^ FFMul(b03,s[2][c])  ^ FFMul(b04,s[3][c]);
         sp[2] = FFMul(b04, s[0][c]) ^ FFMul(b05, s[1][c]) ^ FFMul(b02,s[2][c])  ^ FFMul(b03,s[3][c]);
         sp[3] = FFMul(b03, s[0][c]) ^ FFMul(b04, s[1][c]) ^ FFMul(b05,s[2][c])  ^ FFMul(b02,s[3][c]);
         for (int i = 0; i < 4; i++) s[i][c] = (byte)(sp[i]);
      }

      return s;
}

private static byte[][] MixColumns(byte[][] s){
  int[] sp = new int[4];
      byte b02 = (byte)0x02, b03 = (byte)0x03;
      for (int c = 0; c < 4; c++) {
         sp[0] = FFMul(b02, s[0][c]) ^ FFMul(b03, s[1][c]) ^ s[2][c]  ^ s[3][c];
         sp[1] = s[0][c]  ^ FFMul(b02, s[1][c]) ^ FFMul(b03, s[2][c]) ^ s[3][c];
         sp[2] = s[0][c]  ^ s[1][c]  ^ FFMul(b02, s[2][c]) ^ FFMul(b03, s[3][c]);
         sp[3] = FFMul(b03, s[0][c]) ^ s[1][c]  ^ s[2][c]  ^ FFMul(b02, s[3][c]);
         for (int i = 0; i < 4; i++) s[i][c] = (byte)(sp[i]);
      }

      return s;
}
```

Figure 9

8.3.4 Calculate Number of padding Byte Required and Perform Padding task.



```
int lenght=0;
byte[] padding = new byte[1];
int i;
lenght = 16 - in.length % 16;
padding = new byte[lenght];
padding[0] = (byte) 0x80;

for (i = 1; i < lenght; i++)
 padding[i] = 0;

byte[] tmp = new byte[in.length + lenght];
byte[] bloc = new byte[16];


w = generateSubkeys(key);
```

Figure 10

9. Experimental Results:

9.1 GUI Main Window

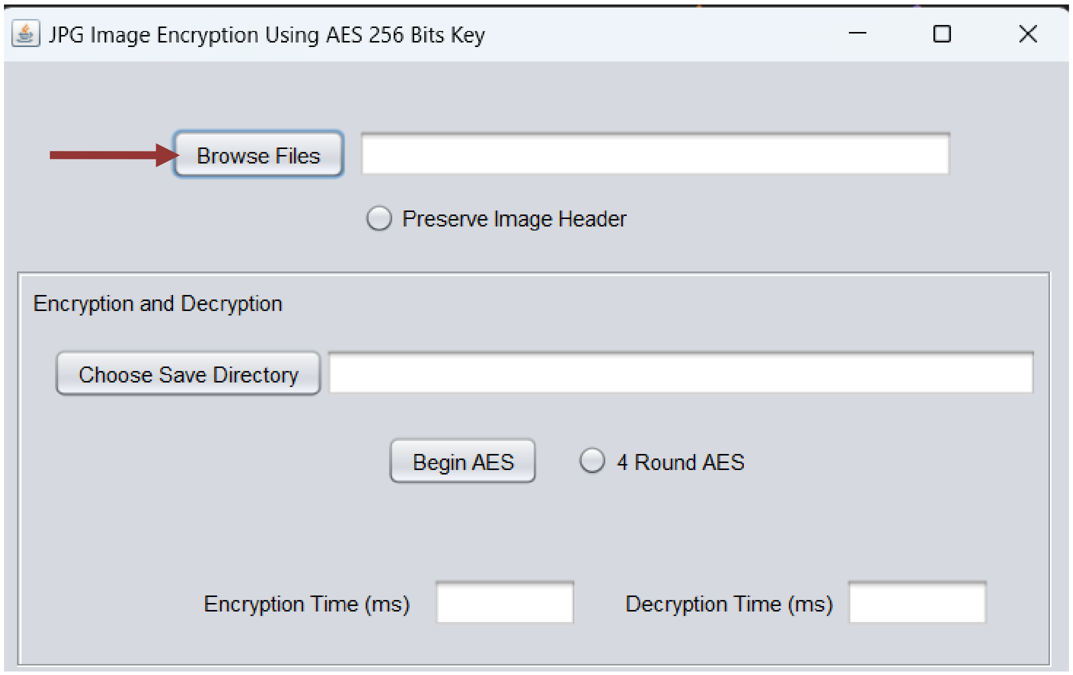


Figure 11

9.2 User Choose Image File for Encryption

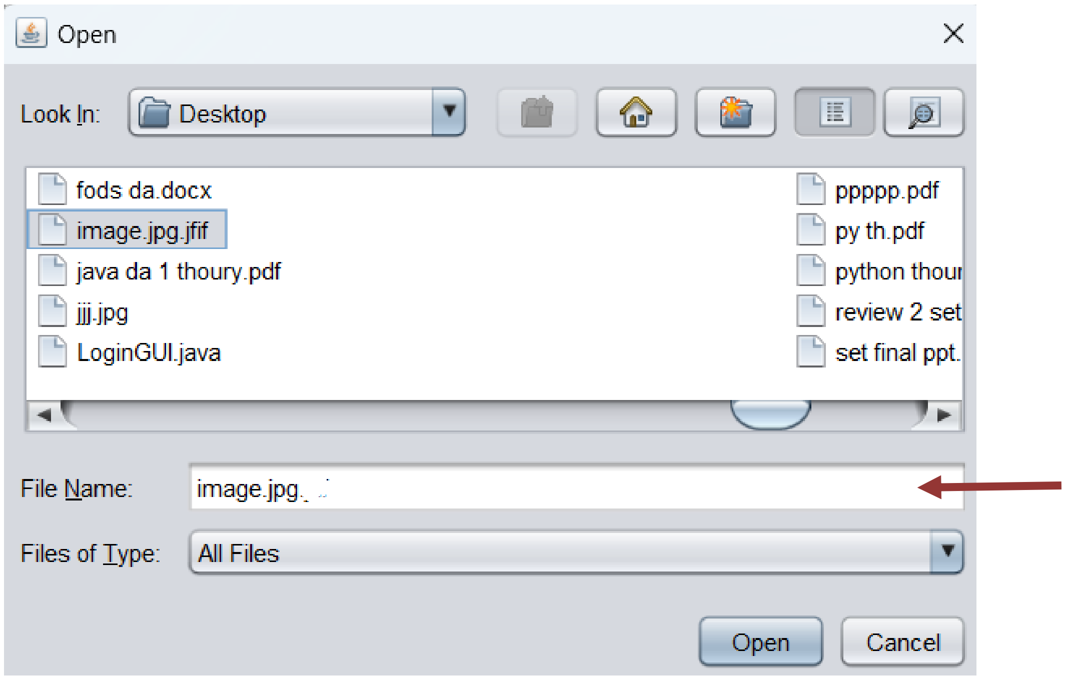
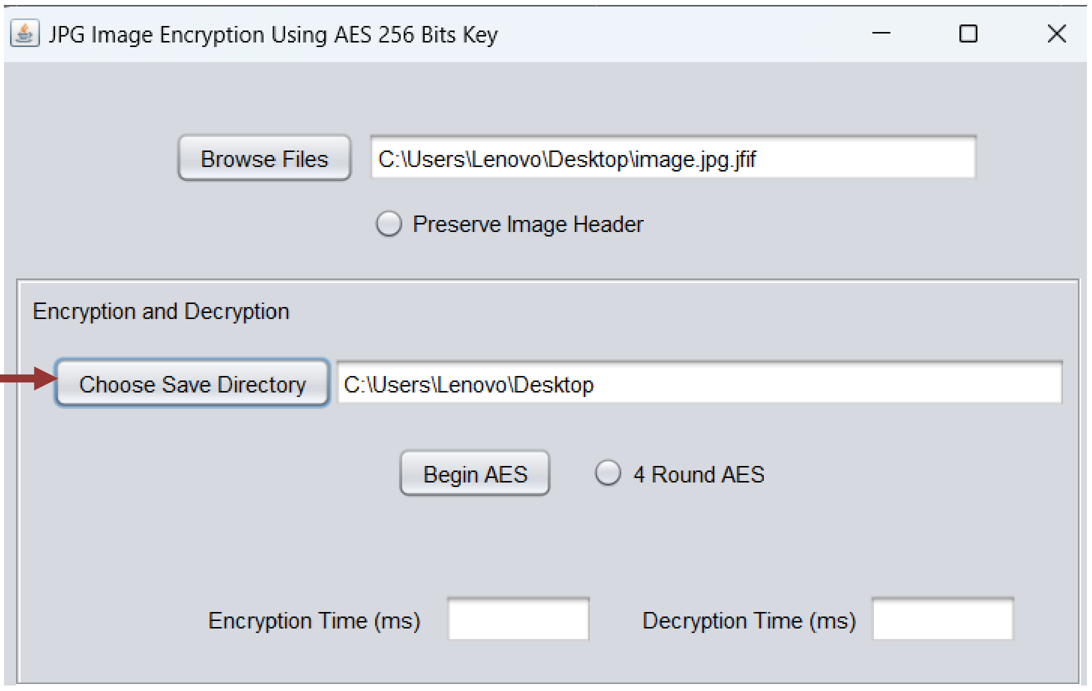


Figure 12

9.3 User Choose Output File Name and Destination After Encryption



C:\Users\Lenovo\Desktop\image.jpg.jfif

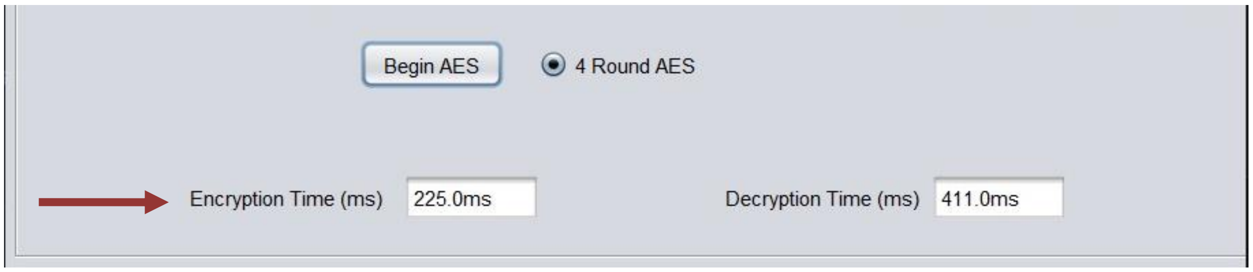
C:\Users\Lenovo\Desktop

Encryption Time (ms)

Decryption Time (ms)

Figure 13

9.4 User Can Choose Number of AES Round 4



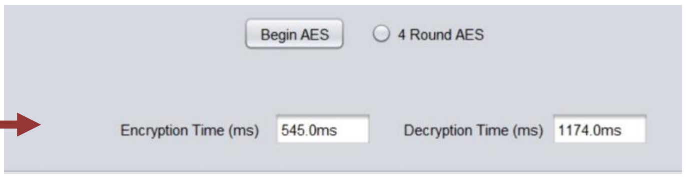
[✓ 4 Round AES]

Encryption Time (ms) 225.0ms

Decryption Time (ms) 411.0ms

Figure 14

9.4.1 Encryption Process 14 Round

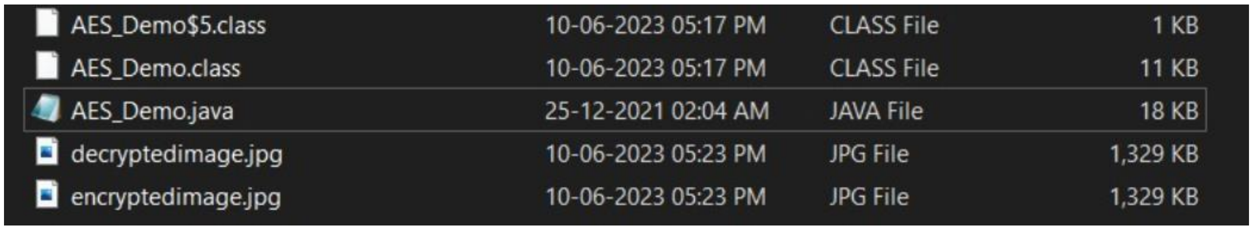


Encryption Time (ms) 545.0ms

Decryption Time (ms) 1174.0ms

Figure 15

9.5 Output:



```
AES_Demo$5.class      10-06-2023 05:17 PM    CLASS File        1 KB
AES_Demo.class        10-06-2023 05:17 PM    CLASS File       11 KB
AES_Demo.java         25-12-2021 02:04 AM    JAVA File        18 KB
decryptedimage.jpg    10-06-2023 05:23 PM    JPG File      1,329 KB
encryptedimage.jpg    10-06-2023 05:23 PM    JPG File      1,329 KB
```

Figure 16

Encrypted Image:

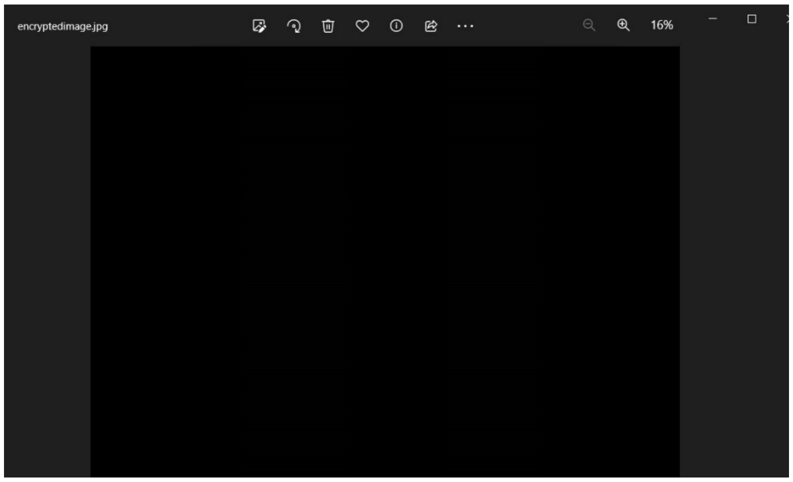


Figure 17

Decrypted Image:

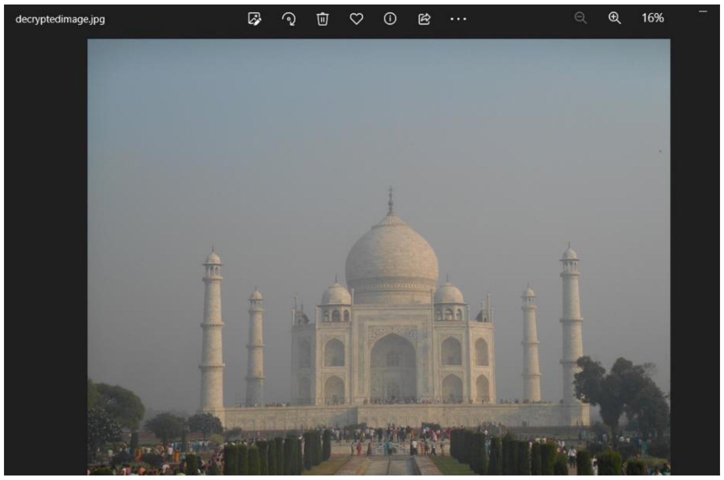


Figure 18

10. Conclusion:

Based on the experimental results and analysis, this project concludes that the choice between 4-round and 14-round AES encryption for encrypting 256-bit JPG format images depends on the specific requirements of the application. If faster processing times and lower computational overhead are of primary concern, the 4-round encryption scheme can be considered. However, if enhanced security is the priority and longer encryption times are acceptable, the 14-round encryption scheme provides a higher level of protection. The findings of this study can guide developers and researchers in selecting the appropriate AES encryption scheme for secure image transmission and storage applications.

11.Future Work:

- Using other types of file formats like a pdf, audio file etc.
- Add user authentication for authenticate valid sender and receiver.
- Explore integrating image compression algorithms to reduce the size of the encrypted image without compromising the encryption strength. This can help optimize storage and transmission requirements while maintaining data security.
- Enhance the key management module to incorporate secure key generation and exchange mechanisms, such as using key derivation functions or public-key cryptography, to strengthen the overall security of the encryption process.
- Refine the user interface module by enhancing the usability, adding additional features like drag-and-drop functionality, and improving the visual feedback provided during the encryption process.

12. Relevant Description About the Figure:

Figure Number	Description
Figure 1	Shows the various steps performed in each round
Figure 2	Represent the s-box table for AES encryption and description
Figure 3	Represent the inverse s-box table for AES encryption and decryption
Figure 4	Project architecture
Figure 5	Java libraries which use for project
Figure 6	2 256-bit key generation using key generator class
Figure 7	Method for add round key
Figure 8	Method for subbytes, inversesubbytes, shiftrows,
Figure 9	Method for mixcolumn
Figure 10	Calculate number of padding byte required and perform padding task.
Figure 11	Gui main window
Figure 12	Window for image file upload for encryption
Figure 13	Userschoose output file name and destination after encryption
Figure 14	User can choose number of aes round 4
Figure 15	Encryption process 14 round
Figure 16	Output file
Figure 17	Encrypted image
Figure 18	Decrypted(original) image

13. References:

[1] Upadhyaya, Akanksha &Shokeen, Vinod & Srivastava, Garima. (2015). Image encryption: Using AES, feature extraction and random no. generation. 1-4. 10.1109/ICRITO.2015.7359286.

[2] Kalubandi, Venkata &Vaddi, Hemanth &Ramineni, Vishnu & Loganathan, Agilandeeswari. (2016). A novel image encryption algorithm using AES and visual cryptography. 808-813. 10.1109/NGCT.2016.7877521.

[3]Bastanta, Artha&Nuryansyah, Ramadhany& Nugroho, Christian &Budiharto, Widodo. (2021). Image Data Encryption Using DES Method. 130-135. 10.1109/ICCSAI53272.2021.9609738.

[4] Hafsa, Amal &Sghaier, Anissa &Wajih, Elhadjyoussef&Machhout, Mohsen & Malek, Jihène. (2017). Image encryption/decryption design using NIOSII soft core processor. 1-5. 10.1109/ICEMIS.2017.8273002.

[5] Quenaya, M. R., Villa-Herrera, A. A., ChambiYtusaca, S. F., YauriItuccayasi, J. E., Velazco-Paredes, Y., & Flores-Quispe, R. (2021). Image Encryption using an Image Pattern based on Advanced Encryption Standard. In C. E. Velasquez-Villada (Ed.), *2021 IEEE Colombian Conference on Communications and Computing, COLCOM 2021* [9486298]

[6] Shariatzadeh, Mahdi & Rostami, Mohamad &Eftekhari, Mahdi. (2021). Proposing a novel Dynamic AES for image encryption using a chaotic map key management approach. Optik. 246. 167779. 10.1016/j.ijleo.2021.167779.

[7] Y. Jha, K. Kaur and C. Pradhan, "Improving image encryption using two-dimensional logistic map and AES," 2016 International Conference on Communication and Signal Processing (ICCSP), Melmaruvathur, India, 2016, pp. 0177-0180, doi: 10.1109/ICCSP.2016.7754116.

[8] Singh, Amandeep & Agarwal, Praveen & Chand, Mehar. (2019). Image Encryption and Analysis using Dynamic AES. 1-6. 10.1109/ICOA.2019.8727711.

[9] Zhang, Qi & Ding, Qun. (2015). Digital Image Encryption Based on Advanced Encryption Standard (AES). 1218-1221. 10.1109/IMCCC.2015.261.

[10] Zhang, Yong & Li, Xueqian& Hou, Wengang. (2017). A fast image encryption scheme based on AES. 624-628. 10.1109/ICIVC.2017.7984631.

[11] Subramanyan, B. &Chhabria, Vivek & Babu, T.G.. (2011). Image encryption based on AES Key Expansion. Emerging Applications of Information Technology, International Conference on. 217-220. 10.1109/EAIT.2011.60.

[12] B. Subramanyan, V. M. Chhabria and T. G. S. Babu, "Image Encryption Based on AES Key Expansion," 2011 Second International Conference on Emerging Applications of Information Technology, Kolkata, 2011, pp. 217-220, doi: 10.1109/EAIT.2011.60.

[13] D. M. Alsaffar et al., "Image Encryption Based on AES and RSA Algorithms," 2020 3rd International Conference on Computer Applications & Information Security (ICCAIS), Riyadh, Saudi Arabia, 2020, pp. 1-5, doi: 10.1109/ICCAIS48893.2020.9096809.

[14] M. A. W. Shalaby, M. T. Saleh and H. N. Elmahdy, "Enhanced Arnold's Cat Map-AES Encryption Technique for Medical Images," 2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES), Giza, Egypt, 2020, pp. 288-295, doi: 10.1109/NILES50944.2020.9257876.

[15] Q. -A. Kester, L. Nana, A. C. Pascu, S. Gire, J. M. Eghan and N. N. Quaynor, "Feature Based Encryption Technique for Securing Forensic Biometric Image Data Using AES and Visual Cryptography," 2014 2nd International Conference on Artificial Intelligence, Modelling and Simulation, Madrid, Spain, 2014, pp. 199-204, doi: 10.1109/AIMS.2014.65.

[16] A. A. A. Laimoon, M. M. A. Elnaby, A. H. Hussein and H. M. A. Kader, "Light Weight Encryption for Medical Images," 2016 26th International Conference on Computer Theory and Applications (ICCTA), Alexandria, Egypt, 2016, pp. 26-31, doi: 10.1109/ICCTA40200.2016.9512947.

[17] C. Chethan, S. N. Raghavendra and N. P. Tejaswini, "An Advanced Assertion and Refurbishment of Images through AES," 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 2018, pp. 570-574, doi: 10.1109/RTEICT42901.2018.9012512.

[18] H. Mogale, M. Esiefarienrhe and L. Letlonkane, "Web Authentication Security Using Image Steganography and AES Encryption," 2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC), Mon Tresor, Mauritius, 2018, pp. 1-7, doi: 10.1109/ICONIC.2018.8601208.

[19] B. J. S. Kumar, V. K. Roshni Raj and A. Nair, "Comparative study on AES and RSA algorithm for medical images," 2017 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 2017, pp. 0501-0504, doi: 10.1109/ICCSP.2017.8286408.

[20] R. Banu and T. Vladimirova, "Investigation of Fault Propagation in Encryption of Satellite Images Using the AES Algorithm," MILCOM 2006 - 2006 IEEE Military Communications conference, Washington, DC, USA, 2006, pp. 1-6, doi: 10.1109/MILCOM.2006.302064.