Image Encryption Using AES Algorithm

A project report

submitted to

MANIPAL ACADEMY OF HIGHER EDUCATION

For Partial Fulfillment of the Requirement for the

Award of the Degree of Bachelor

of

Technology

in

Information Technology

By

Tanishta, Prarthana Prem

Reg. No. 225811120, 225811260

Under the guidance of

Dr. Abhijit Das
Assistant Professor
Department of Information Technology
Manipal Insitute of Technology
Bengaluru, India

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

DECLARATION

I hereby declare that this project work entitled **Image Encryption Using AES Algorithm** is original and has been carried out by me in the Department of Information and Communication Technology of Manipal Institute of Technology, Manipal, under the guidance of **Dr. Abhijeet Das, Assistant Professor – Senior Scale**, Department of Information and Technology, M. I. T., Bengaluru. No part of this work has been submitted for the award of a degree or diploma either to this University or to any other Universities.

Place: Manipal

Date: 29/10/2024

Tanishta

Prarthana Prem

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

CERTIFICATE

This is to certify that this project entitled **Image Encryption Using AES Algorithm** is a Bonafide project work done by **Ms. Tanishta (Reg.No.:225811120)** and **Ms. Prarthana Prem (Reg.No.:225811260)** at Manipal Institute of Technology, Manipal, independently under my guidance and supervision for the award of the Degree of Bachelor of Technology in Information Technology.

Dr. Abhijeet Das	Dr. Dayanand P
Assistant Professor	Professor & Head
Department of I & T	Department of I & T
Manipal Institute of Technology	Manipal Institute of Technology
Bengaluru, India	Bengaluru, India

TABLE OF CONTENTS

1. Introduction

1.1 Background

The prevalence of devices like computers and mobile phones for communication, data storage, and transmission has significantly increased in recent times. This surge in usage has led to a corresponding rise in unauthorized users attempting to access data through illicit means, thus highlighting the issue of data security. Images transmitted over insecure channels from various sources may contain sensitive information, and some images are highly confidential, making it essential to protect them from potential attacks.

To address this challenge, we employ the AES algorithm for encrypting and decrypting images. The encrypted data becomes unreadable to unauthorized users, allowing it to be transmitted over the network and decrypted using AES on the receiving end. This approach ensures secure image transmission.

The primary goal of this project is to establish a secure method for transferring images between the sender and receiver. Images must be encrypted prior to transmission over the network and accurately decrypted on the recipient's side.

1.2 Objectives

The project has several key objectives:

- Convert an image into an unreadable format through encryption.
- Restore the encrypted image to its original form via decryption.
- Facilitate secure image transfer over networks such as the internet.
- Guarantee that no alterations occur during the transfer process over the network.

1.3 Scope

1. Product scope:
The project implements AES encryption to securely send the specified image over a network. On the receiving end, the recipient utilizes decryption code to retrieve the original image, enabling the safe transmission of confidential and sensitive information across the internet. This functionality is particularly beneficial in fields like healthcare and military operations.

2. Design and Implementation constraints:
- The front end must be developed using Python.
- The encryption and decryption processes should utilize the AES algorithm.
- The original image must be in .jpeg or .png format.

3. Assumptions and Dependencies:
- The sender and receiver are connected through a network.

2. Literature Review

2.1 Overview

This project implements a secure image encryption and decryption tool using the Advanced Encryption Standard (AES) algorithm, a widely recognized method for data protection. As digital data becomes more pervasive, securing sensitive information—particularly images—has become crucial, especially for fields where confidentiality is paramount, such as healthcare, defense, and finance. This program aims to prevent unauthorized access by transforming image files into encrypted formats, making them unreadable to anyone without the correct decryption key.

The application workflow is divided into two primary functions: encryption and decryption. During encryption, the image file is processed to extract pixel data, which is then transformed into a structured string format that includes metadata (such as image dimensions). The AES algorithm encrypts this string, using a user-provided password as a secure key, to generate a ciphertext file. On the other end, decryption involves reading this ciphertext file, using the same password-derived key to decrypt it back into plaintext, and reconstructing the image from the original pixel data. This process ensures data confidentiality while allowing for accurate image reconstruction.

The user interface, built with Tkinter, allows users to select images and input passwords for encryption and decryption. This GUI provides a straightforward experience, enabling non-technical users to utilize advanced encryption techniques with minimal complexity. The program uses Python’s Pillow library for image processing and PyCryptodome for AES encryption and decryption. NumPy is also leveraged for generating random values used in testing, making the code efficient and modular.

2.2 Key Technologies

1. Python Programming Language:

- Role: Python serves as the primary programming language for this project, chosen for its simplicity, extensive libraries, and support for encryption and GUI frameworks.

- Advantages: Python’s readability and ease of use make complex tasks such as encryption accessible while providing a wealth of libraries tailored for specific functionalities like image manipulation, GUI creation, and cryptography.

2. Hashlib:

- Role: The hashlib library is used to hash the user-provided password into a 256-bit key using SHA-256. This secure hashing process ensures that the password-derived key is both consistent and secure.

- Importance: Hashing is essential for generating a fixed-length, secure key from the password. Without hashing, password-based encryption might be vulnerable to unauthorized access or password guessing attacks.

- Security Benefit: SHA-256, a cryptographic hash function, is widely used in security applications and ensures that the derived key is difficult to reverse-engineer. This enhances the robustness of the AES encryption.

3. AES (Advanced Encryption Standard) from PyCryptodome:

- Purpose: AES is the core cryptographic algorithm for encrypting and decrypting the image data, selected for its security, speed, and widespread use in data protection.

- Implementation: AES is applied in CBC (Cipher Block Chaining) mode, which enhances security by chaining encryption blocks. This mode requires an initialization vector (IV), which prevents repetitive ciphertexts even with identical plaintext inputs.

- Password Hashing: The PyCryptodome library also utilizes Python’s `hashlib` library to generate a SHA-256 hash of the user-provided password, creating a secure 256-bit encryption key. This password-derived key ensures that only users with the correct password can decrypt the image, adding a layer of security.

- Benefits: AES is known for its strong encryption capabilities, and its application here guarantees that the image data remains protected during storage or transmission.

4. Tkinter (GUI Toolkit):

- Purpose: Tkinter provides the graphical interface for this application, enabling users to interact with the encryption tool through a visual interface.

- Features: The Tkinter GUI includes elements like file selection dialogs, password entry fields, and status messages. These features simplify the user experience by enabling file upload, password entry, and interaction with encryption and decryption processes through buttons.

- Benefits: Tkinter’s integration with Python allows for the creation of lightweight, functional interfaces without additional dependencies, making the application easy to deploy and use for non-technical users.

5. Pillow (PIL) for Image Processing:

- Role: Pillow, a powerful image-processing library in Python, is used to handle image loading, resizing, and manipulation.

- Usage: This library enables the program to convert images into formats suitable for encryption by extracting and structuring pixel data, resizing images as needed, and saving them in encrypted or decrypted form.

- Key Functions: Pillow’s functions include loading images, accessing pixel data, creating new images, and resizing images as necessary. These operations are essential for transforming images into a structured format that can be securely encrypted.

3. Methodology

3.1 Approach

The program utilizes the AES algorithm for encrypting and decrypting image files, specifically to ensure secure transmission and protect images from unauthorized access. Here's a detailed breakdown of the approach:

1. Image Encryption:

- Load Image: The selected image is loaded using the Pillow library. The image can be resized to a default dimension of 256 x 256 pixels to ensure compatibility and manageable data size.

- Convert Pixel Data to Plaintext: The pixel values (R, G, B channels) are extracted and formatted as strings. The image's width and height are appended to this string, serving as metadata to reconstruct the image during decryption.

- Padding: Since AES requires data in multiples of 16 bytes, padding is applied to the plaintext data to meet this requirement. The padding is removed after decryption to restore the original data structure.

- Password-Based Key Generation: A secure hash (`SHA-256`) of the user-provided password is generated to create a 256-bit AES encryption key, ensuring that encryption is tied to the password.

- Encryption: The plaintext pixel data is encrypted using AES in CBC (Cipher Block Chaining) mode with a fixed initialization vector (IV) for added security. The resulting ciphertext is saved as an encrypted file (`encrypted.jpeg`), which cannot be interpreted without decryption.

2. Image Decryption:

- Read Encrypted File: The encrypted image file is loaded, and the ciphertext data is decrypted using AES with the same password-derived key.

- Data Un-padding and Parsing: The decrypted data is unpadded to remove any extra bytes added during encryption. The original width, height, and pixel values are extracted from the decrypted data string.

- Reconstruct Image: A new image is created with the same dimensions, and pixel values are applied based on the decrypted data, reconstructing the original image and saving it as `decrypted.jpeg`.

3. User Interface:

- Tkinter GUI: The application provides a simple GUI where users can select an image, input a password, and initiate encryption or decryption.

- Password Entry: During both encryption and decryption, the GUI prompts the user for a password, ensuring that only authorized users can access encrypted images.

- Status Messages: After completing encryption or decryption, a success message confirms that the file has been saved, providing clear feedback to the user.

3.2 Tools and Technologies

1. Python:

Python serves as the main language due to its extensive libraries for GUI, cryptography, and image processing, simplifying the implementation of complex functionalities.

2. Tkinter:

- Purpose: Tkinter provides a basic yet effective GUI for selecting images, inputting passwords, and initiating the encryption/decryption process.

- Usage: The GUI has buttons for image encryption and decryption, file selection dialogs, and input boxes for password collection, making the tool user-friendly and accessible.

3. Pillow (PIL):

- Purpose: The Pillow library, an imaging library in Python, handles loading, resizing, and processing images.

- Usage: Pillow allows loading images into a format suitable for pixel data extraction and conversion, essential for both encrypting and reconstructing images after decryption.

4. PyCryptodome's AES Module:

- Purpose: AES (Advanced Encryption Standard) is a robust encryption algorithm essential for securing image data. The `pycryptodome` library provides a reliable AES implementation.

- Usage: AES in CBC mode is used to securely encrypt pixel data, with password-derived keys ensuring only authorized users can decrypt the images. The use of a fixed initialization vector (IV) further enhances security.

5. NumPy:

- Purpose: NumPy supports efficient array manipulations and random number generation.

- Usage: Random colour values are generated using NumPy, allowing the creation of secret images for testing and image manipulation purposes.

6. Hashlib:

- Purpose: The hashlib library is used to generate a SHA-256 hash of the password, creating a secure key for AES encryption.

- Usage: The hash ensures that the password reliably produces a consistent key, tying encryption and decryption securely to the user's input.

3.3 Data Collection

1. Image Files:

- Type and Format: The program requires images in `.jpeg` or `.png` format as input files. These formats are widely used for image data and are compatible with the encryption and decryption process.

- Selection Process: The GUI's file dialog allows users to select an image from their system, ensuring a smooth and user-friendly experience. This selection process supports real-time encryption and decryption on chosen images.

- Size Considerations: To maintain encryption efficiency, images larger than 5MB are not recommended, as they may slow down the process due to increased computational demands.

2. Password:

- Purpose: The user-provided password serves as the encryption key, ensuring the image remains secure during transmission and is accessible only to those with the correct password.

- Collection and Security: Collected via a dialog box, the password is hashed with SHA-256 to create a secure 256-bit key for AES encryption, safeguarding the image data.

- Role in Encryption/Decryption: Since the AES algorithm is symmetric, the same password is required for both encryption and decryption, adding a layer of security as only the user with the correct password can restore the original image.

4. Implementation

4.1 Working

Functional Requirements:

- The system will encrypt the provided image into an unreadable format using the AES encryption function.
- The system will also decrypt the received encrypted image back to a readable format via the AES decryption function, ensuring that the output image matches the original image.
- Additionally, the system guarantees that the image is securely transmitted through any medium, with no third-party access allowed to modify the file during transit.

Non-Functional Requirements:

Performance Requirements:

- For effective encryption, the image size should not exceed 5MB.
- Decryption should complete within 10 seconds.

Safety and Security Requirements:

- If decryption exceeds 10 seconds, the message will be discarded (indicating potential corruption during transmission), prompting the sender to resend it.
- Encryption employs a key, and decryption will only be successful when the same key is used on the receiver's side.

Software Quality Attributes:

<u>Reliability:</u>

The system remains unaffected by external factors. The AES algorithm is widely accepted and produces consistent results, minimizing error occurrences. Errors are primarily associated with rare transmission glitches, thus affirming the system's reliability.

<u>Usability:</u>

The software features a user-friendly Python-based GUI that facilitates easy navigation through buttons. Individuals with a basic understanding of computers can effectively use this software for image encryption and decryption using a key.

Testability:
The system is designed for straightforward testing and defect identification. It comprises distinct modules that perform specific functions, allowing for individual testing of each module.

External Interface Requirements:
User Interfaces:
The user interface presents a box for entering the image location. It includes two buttons for encoding and decoding the image. Upon clicking a button, the next window prompts the user for a password and includes a submit button. After submission, it displays the filename of the newly generated image file.

Hardware Interfaces:
Sender Computer: Used to view the original and encrypted images.
Receiver Computer: Used to view the decrypted image.

Software Interfaces:
The frontend is developed using the Python module Tkinter(), which creates an interactive window for users. Users must provide the location of the image to encode or decode. They can choose to either encode or decode the image.
The interface requests a password, which acts as the encryption key. After the necessary details are submitted, Python functions defined in the code will encode or decode the image using the AES file from the `Crypto.Cipher` module of Python. Upon completion of this process, the user will receive the encrypted or decrypted image as output, saved in the same directory as the input image file.

4.2 Diagrams

1) Work Breakdown Structure

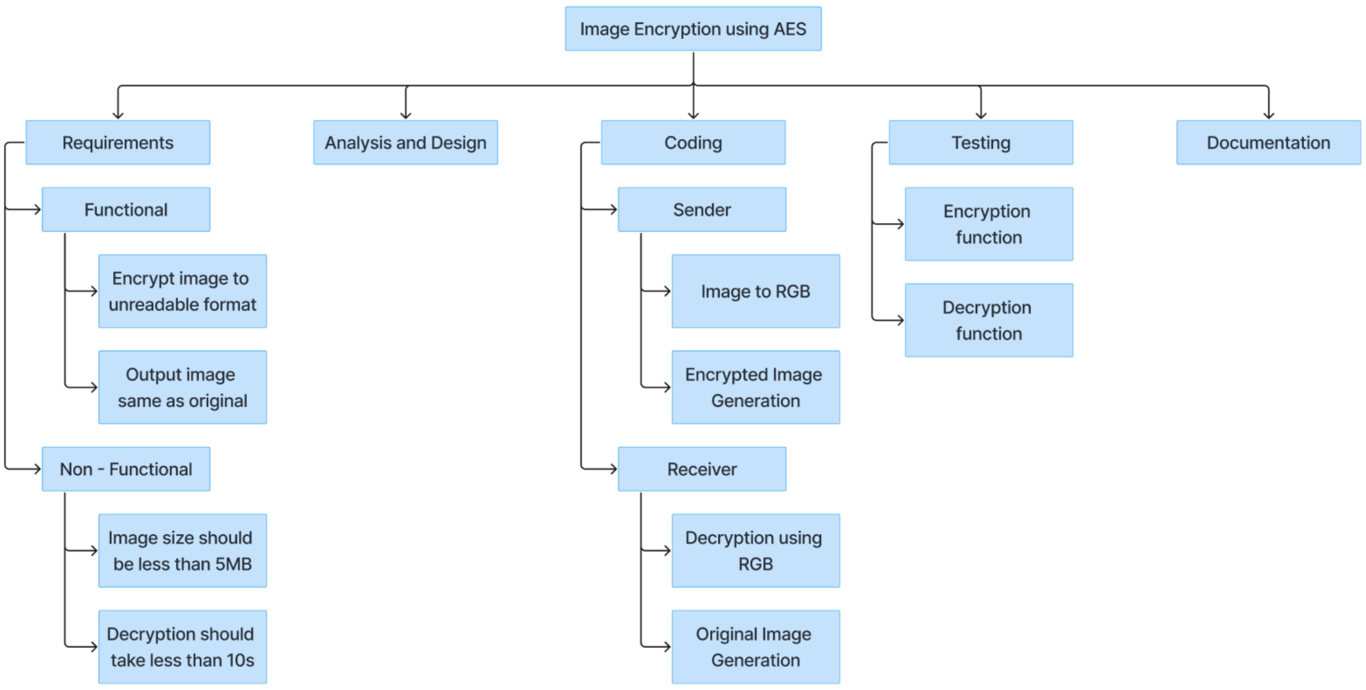
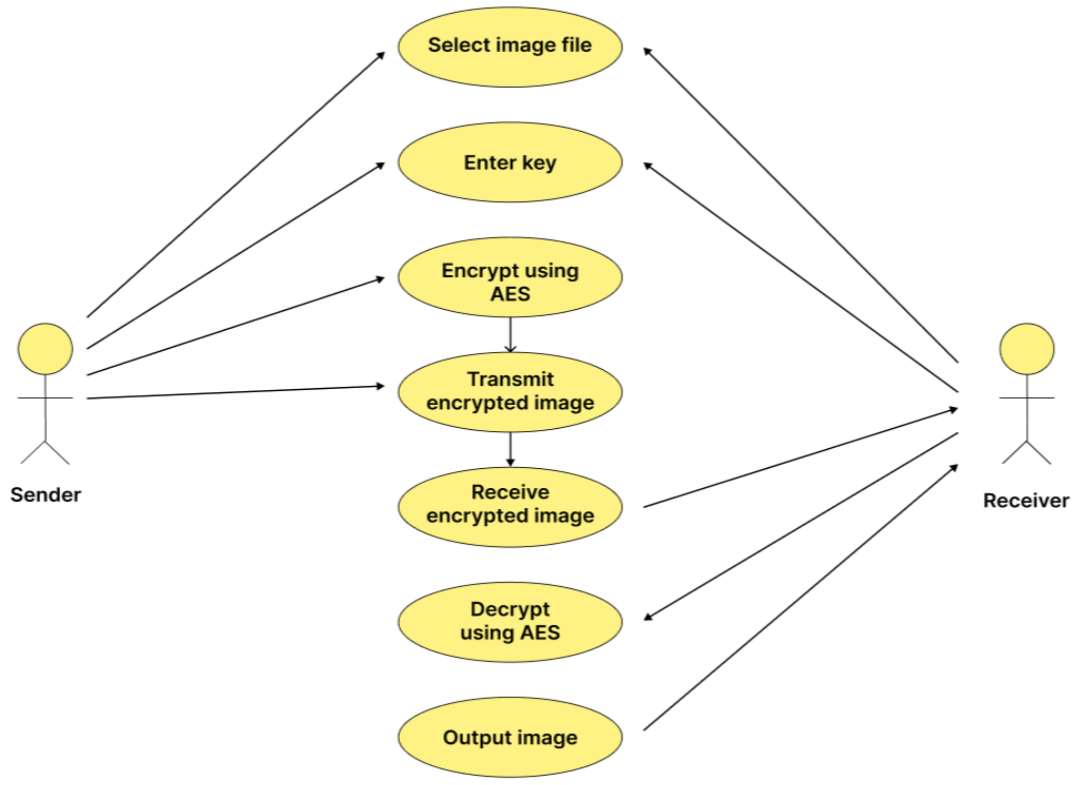
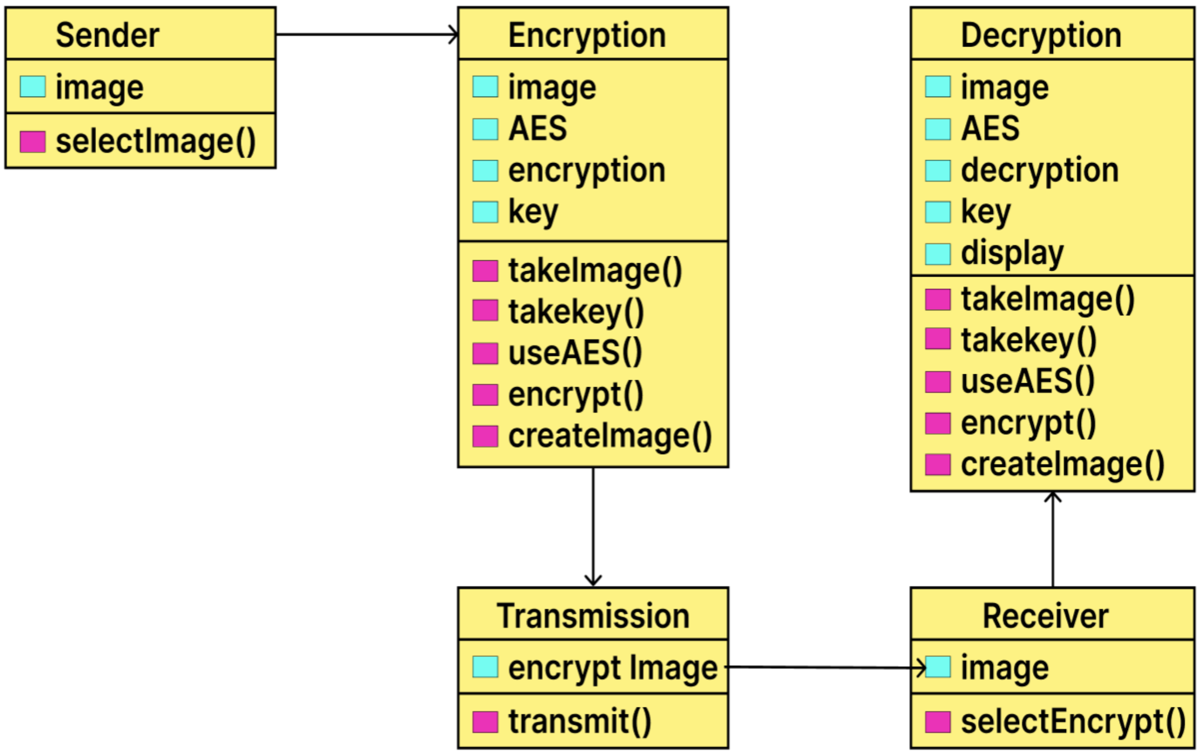


Image Encryption using AES
Requirements
Functional
Encrypt image to unreadable format
Output image same as original
Non - Functional
Image size should be less than 5MB
Decryption should take less than 10s
Analysis and Design
Coding
Sender
Image to RGB
Encrypted Image Generation
Receiver
Decryption using RGB
Original Image Generation
Testing
Encryption function
Decryption function
Documentation

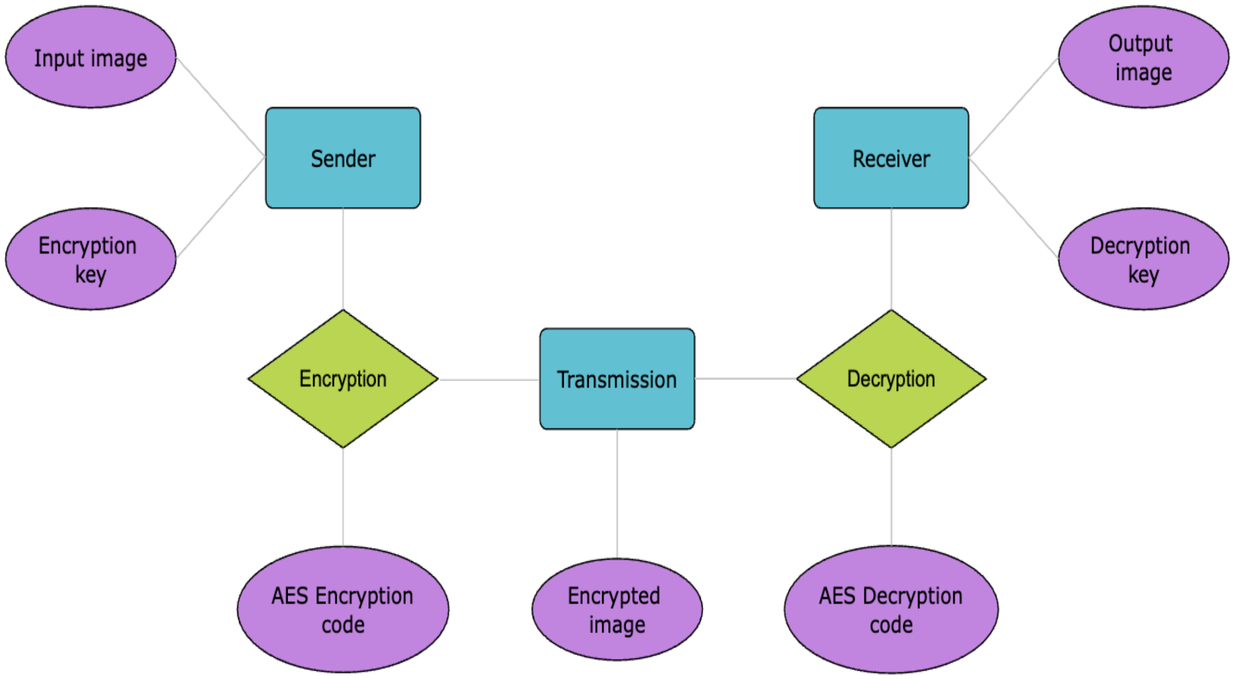
2) Use case Diagram:



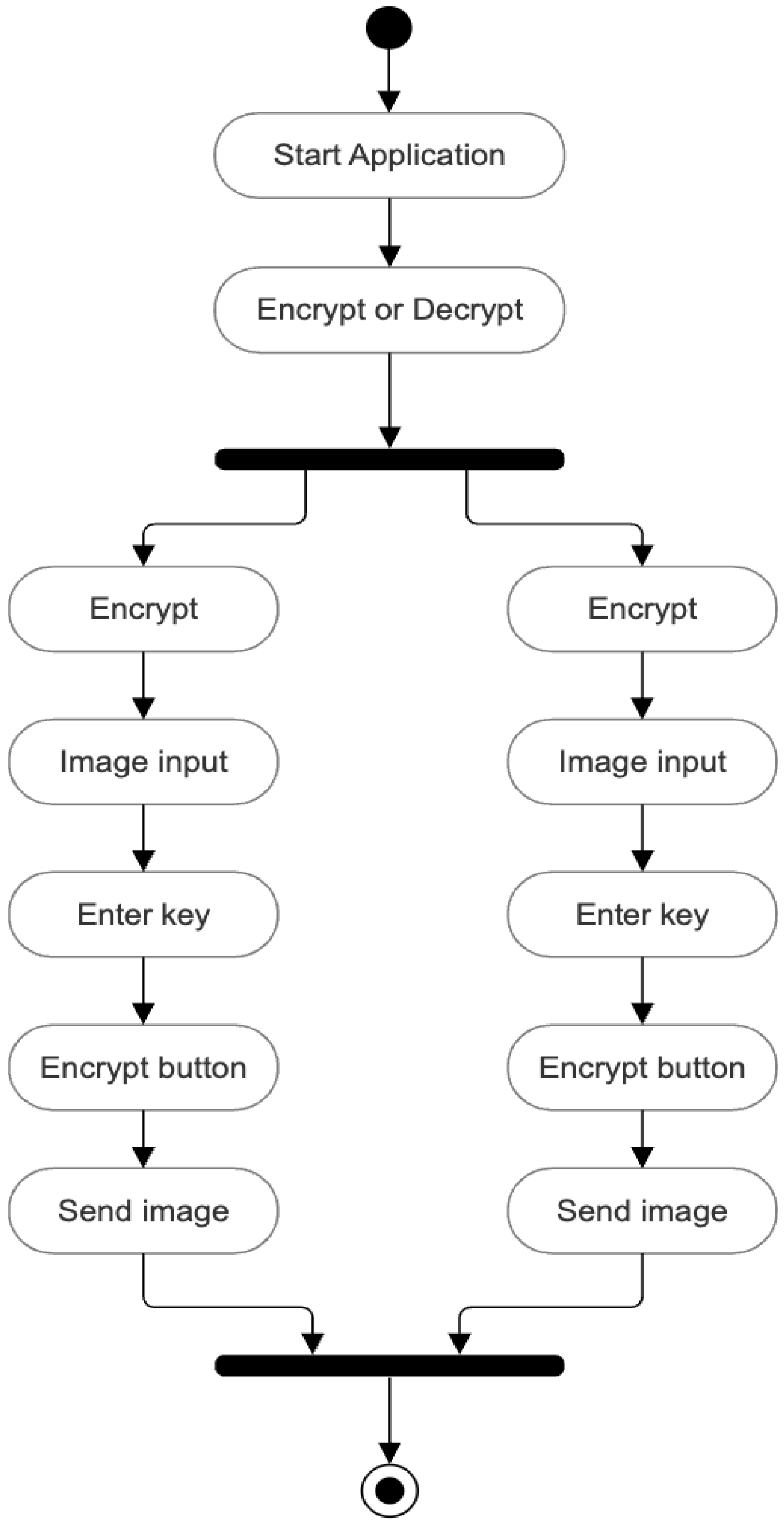
3) **Class Diagram:**



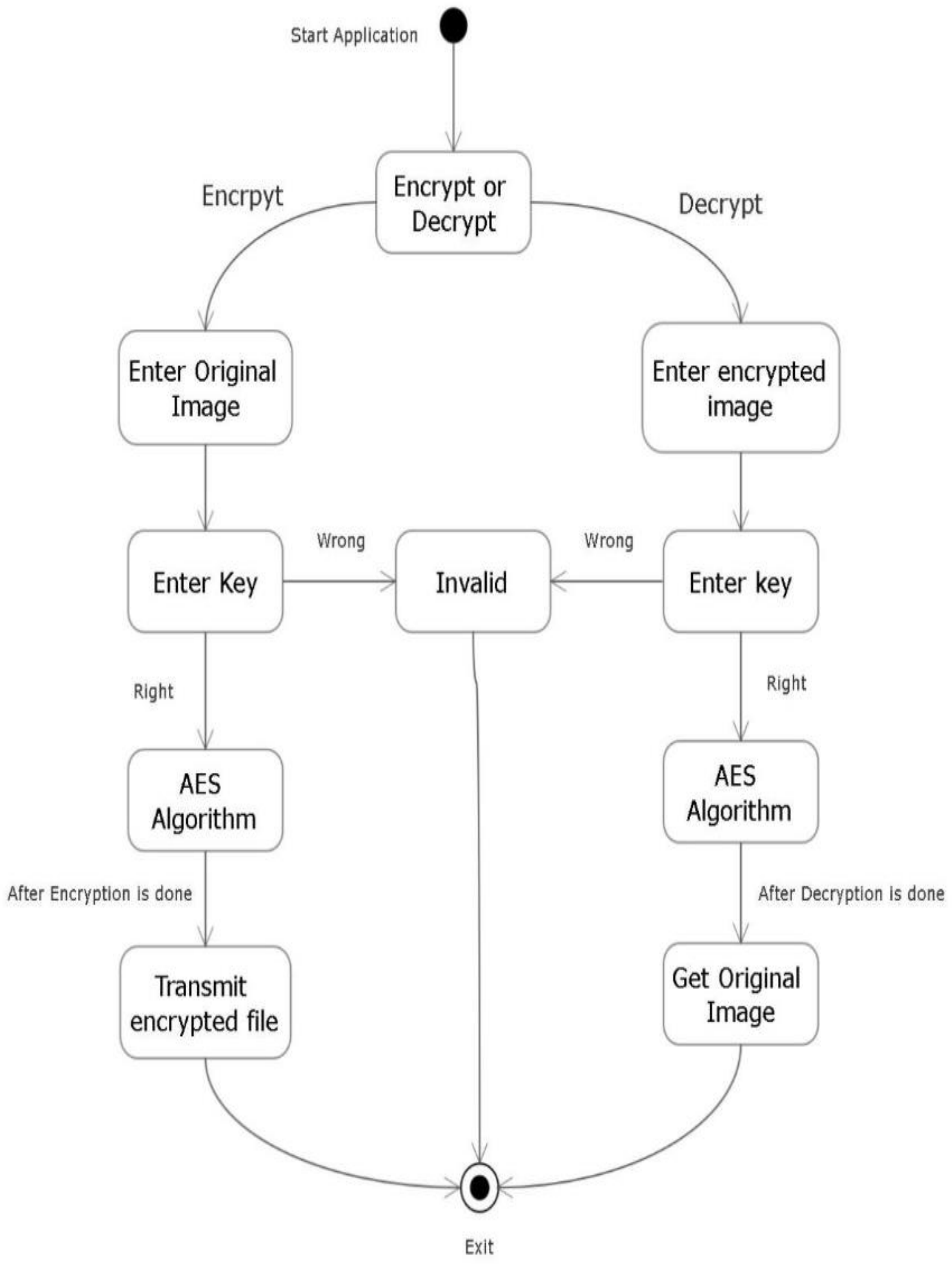
4) **ER Diagram:**



5) **Activity Diagram:**



6) **State Chart Diagram:**



5. Results and Discussion

CODE:

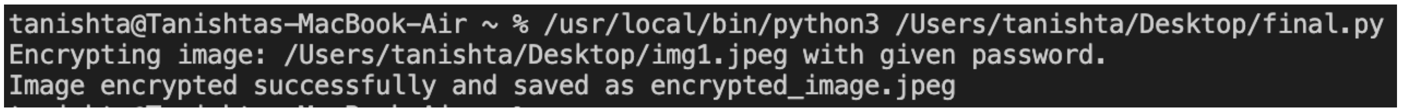
```
from __future__ import division, print_function, unicode_literals
import sys
import random
import argparse
import logging
from tkinter import Tk, Button, filedialog, messagebox, simpledialog
import os
from PIL import Image
import numpy as np
from Crypto.Cipher import AES
import hashlib

global password

def load_image(name):
return Image.open(name)

def prepare_message_image(image, size=(256, 256)):
if size != image.size:
image = image.resize(size, Image.Resampling.LANCZOS)
return image

def generate_secret(size):
width, height = size
new_secret_image = Image.new(mode="RGB", size=(width * 2, height * 2))

for x in range(0, 2 * width, 2):
for y in range(0, 2 * height, 2):
color1 = np.random.randint(255)
color2 = np.random.randint(255)
color3 = np.random.randint(255)
new_secret_image.putpixel((x, y), (color1, color2, color3))
new_secret_image.putpixel((x + 1, y), (255 - color1, 255 - color2, 255 - color3))
new_secret_image.putpixel((x, y + 1), (255 - color1, 255 - color2, 255 - color3))
new_secret_image.putpixel((x + 1, y + 1), (color1, color2, color3))
return new_secret_image

def pad(data):
# Pad data to be a multiple of AES block size (16 bytes)
while len(data) % 16 != 0:
data += ' ' # Using space for padding
return data

def unpad(data):
# Remove padding
```

```
return data.rstrip(b' ')

def encrypt(imagename, password):
plaintextstr = ""
im = Image.open(imagename)
im = prepare_message_image(im)
pix = im.load()
width, height = im.size
# Constructing the plaintext string
plaintextstr += "w" + str(width) + "w" # Adding width info
plaintextstr += "h" + str(height) + "h" # Adding height info
print("Encrypting image:", imagename, "with given password.")
for x in range(width):
for y in range(height):
r, g, b = pix[x, y]
plaintextstr += str(r).zfill(3) + str(g).zfill(3) + str(b).zfill(3) # Ensure RGB values are 3 digits
# Print the pixel values to the terminal
print(f"Pixel ({x}, {y}): R={r}, G={g}, B={b}")

# Encrypting the plaintext
key = hashlib.sha256(password.encode()).digest()
cipher = AES.new(key, AES.MODE_CBC, b'This is an IV456') # Use bytes for the IV.
plaintextstr = pad(plaintextstr) # Pad plaintext
ciphertext = cipher.encrypt(plaintextstr.encode())
enc_image_name = "encrypted.jpeg"
with open(enc_image_name, 'wb') as f:
f.write(ciphertext)
messagebox.showinfo("Success", f"Image encrypted and saved as {enc_image_name}")

def decrypt(ciphername, password):
with open(ciphername, 'rb') as cipher:
ciphertext = cipher.read()
# Decrypting ciphertext with password
key = hashlib.sha256(password.encode()).digest()
obj2 = AES.new(key, AES.MODE_CBC, b'This is an IV456') # Use bytes for the IV.
decrypted = obj2.decrypt(ciphertext)

# Unpad the decrypted data
decrypted = unpad(decrypted)
# Extract dimensions of images.
decrypted_str = decrypted.decode('latin1') # Decode using 'latin1' to preserve binary data.
newwidth = int(decrypted_str.split("w")[1].split("h")[0])
newheight = int(decrypted_str.split("h")[1].split("h")[0]) # Adjusted parsing to avoid issues

# Reconstructing the image from the decrypted string.
newim = Image.new("RGB", (newwidth, newheight))
pix = newim.load()

# Rebuild the image from the decrypted string.
index = decrypted_str.index("h") + len(str(newheight)) + 1 # Skip past dimensions.
for x in range(newwidth):
for y in range(newheight):
# Extract RGB values as 3-character sequences.
```

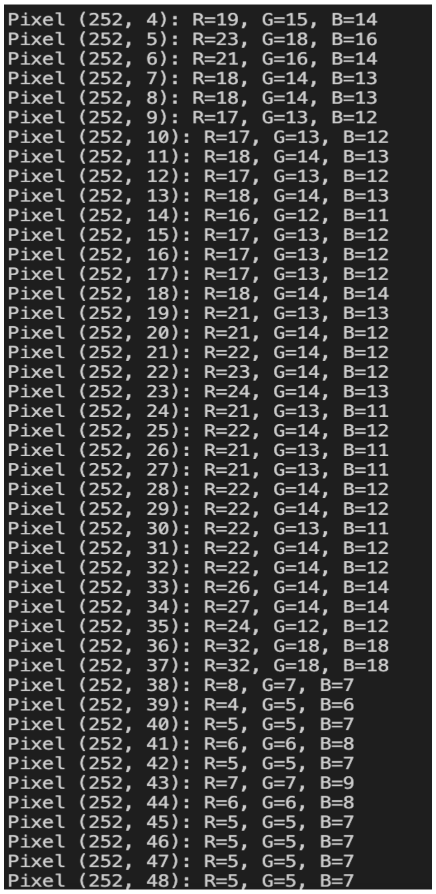
```
r = int(decrypted_str[index:index + 3])
g = int(decrypted_str[index + 3:index + 6])
b = int(decrypted_str[index + 6:index + 9])
pix[x, y] = (r, g, b)
index += 9
newim.save("decrypted.jpeg")
messagebox.showinfo("Success", "Image decrypted and saved as decrypted.jpeg")

def load_image_for_encryption():
filename = filedialog.askopenfilename()
if filename:
password = simpledialog.askstring("Password", "Enter password for encryption:")
encrypt(filename, password)

def load_image_for_decryption():
filename = filedialog.askopenfilename()
if filename:
password = simpledialog.askstring("Password", "Enter password for decryption:")
decrypt(filename, password)

# Initialize the Tkinter GUI
root = Tk()
root.title("Image Encryption/Decryption Tool")
root.geometry("400x300")

# Add buttons for loading images
load_encrypt_button = Button(root, text="Load Image for Encryption", command=load_image_for_encryption)
load_encrypt_button.pack(pady=10)

load_decrypt_button = Button(root, text="Load Image for Decryption", command=load_image_for_decryption)
load_decrypt_button.pack(pady=10)

# Start the Tkinter event loop
root.mainloop()
```

5.1 Outcomes

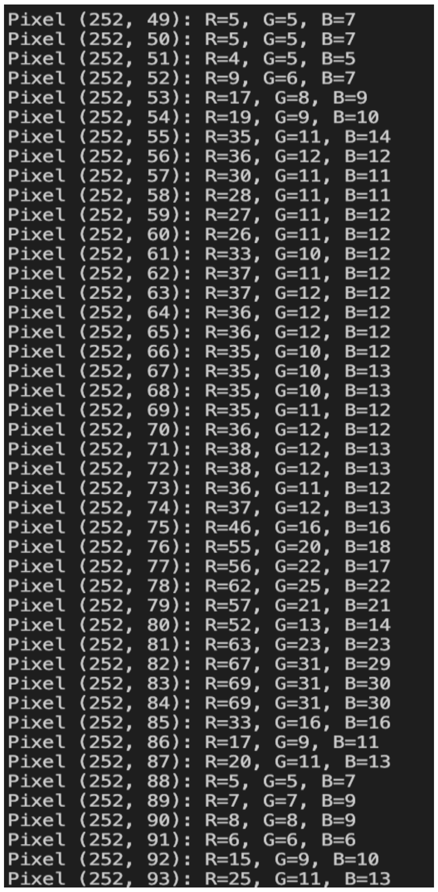
OUTPUT IN TERMINAL:



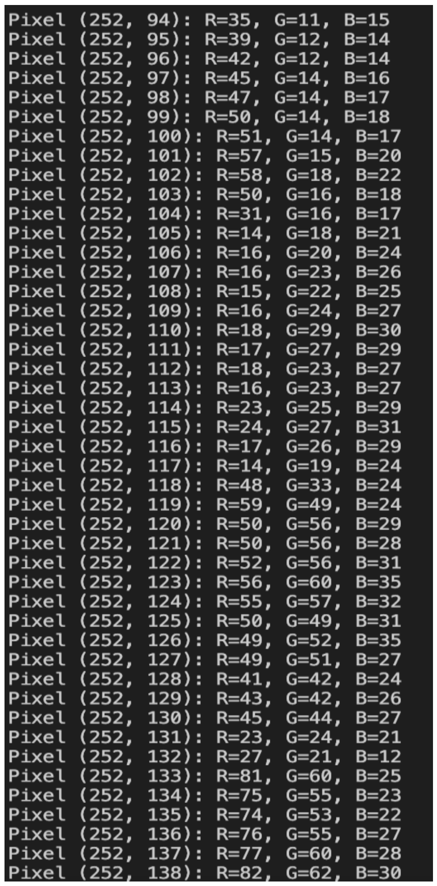
```
tanishta@Tanishtas-MacBook-Air ~ % /usr/local/bin/python3 /Users/tanishta/Desktop/final.py
Encrypting image: /Users/tanishta/Desktop/img1.jpeg with given password.
Image encrypted successfully and saved as encrypted_image.jpeg
```



```
Pixel (252, 4): R=19, G=15, B=14
Pixel (252, 5): R=23, G=18, B=16
Pixel (252, 6): R=21, G=16, B=14
Pixel (252, 7): R=18, G=14, B=13
Pixel (252, 8): R=18, G=14, B=13
Pixel (252, 9): R=17, G=13, B=12
Pixel (252, 10): R=17, G=13, B=12
Pixel (252, 11): R=18, G=14, B=13
Pixel (252, 12): R=17, G=13, B=12
Pixel (252, 13): R=18, G=14, B=13
Pixel (252, 14): R=16, G=12, B=11
Pixel (252, 15): R=17, G=13, B=12
Pixel (252, 16): R=17, G=13, B=12
Pixel (252, 17): R=17, G=13, B=12
Pixel (252, 18): R=18, G=14, B=14
Pixel (252, 19): R=21, G=13, B=13
Pixel (252, 20): R=21, G=14, B=12
Pixel (252, 21): R=22, G=14, B=12
Pixel (252, 22): R=23, G=14, B=12
Pixel (252, 23): R=24, G=14, B=13
Pixel (252, 24): R=21, G=13, B=11
Pixel (252, 25): R=22, G=14, B=12
Pixel (252, 26): R=21, G=13, B=11
Pixel (252, 27): R=21, G=13, B=11
Pixel (252, 28): R=22, G=14, B=12
Pixel (252, 29): R=22, G=14, B=12
Pixel (252, 30): R=22, G=13, B=11
Pixel (252, 31): R=22, G=14, B=12
Pixel (252, 32): R=22, G=14, B=12
Pixel (252, 33): R=26, G=14, B=14
Pixel (252, 34): R=27, G=14, B=14
Pixel (252, 35): R=24, G=12, B=12
Pixel (252, 36): R=32, G=18, B=18
Pixel (252, 37): R=32, G=18, B=18
Pixel (252, 38): R=8, G=7, B=7
Pixel (252, 39): R=4, G=5, B=6
Pixel (252, 40): R=5, G=5, B=7
Pixel (252, 41): R=6, G=6, B=8
Pixel (252, 42): R=5, G=5, B=7
Pixel (252, 43): R=7, G=7, B=9
Pixel (252, 44): R=6, G=6, B=8
Pixel (252, 45): R=5, G=5, B=7
Pixel (252, 46): R=5, G=5, B=7
Pixel (252, 47): R=5, G=5, B=7
Pixel (252, 48): R=5, G=5, B=7
```



```
Pixel (252, 49): R=5, G=5, B=7
Pixel (252, 50): R=5, G=5, B=7
Pixel (252, 51): R=4, G=5, B=5
Pixel (252, 52): R=9, G=6, B=7
Pixel (252, 53): R=17, G=8, B=9
Pixel (252, 54): R=19, G=9, B=10
Pixel (252, 55): R=35, G=11, B=14
Pixel (252, 56): R=36, G=12, B=12
Pixel (252, 57): R=30, G=11, B=11
Pixel (252, 58): R=28, G=11, B=11
Pixel (252, 59): R=27, G=11, B=12
Pixel (252, 60): R=26, G=11, B=12
Pixel (252, 61): R=33, G=10, B=12
Pixel (252, 62): R=37, G=11, B=12
Pixel (252, 63): R=37, G=12, B=12
Pixel (252, 64): R=36, G=12, B=12
Pixel (252, 65): R=36, G=12, B=12
Pixel (252, 66): R=35, G=10, B=12
Pixel (252, 67): R=35, G=10, B=13
Pixel (252, 68): R=35, G=10, B=13
Pixel (252, 69): R=35, G=11, B=12
Pixel (252, 70): R=36, G=12, B=12
Pixel (252, 71): R=38, G=12, B=13
Pixel (252, 72): R=38, G=12, B=13
Pixel (252, 73): R=36, G=11, B=12
Pixel (252, 74): R=37, G=12, B=13
Pixel (252, 75): R=46, G=16, B=16
Pixel (252, 76): R=55, G=20, B=18
Pixel (252, 77): R=56, G=22, B=17
Pixel (252, 78): R=62, G=25, B=22
Pixel (252, 79): R=57, G=21, B=21
Pixel (252, 80): R=52, G=13, B=14
Pixel (252, 81): R=63, G=23, B=23
Pixel (252, 82): R=67, G=31, B=29
Pixel (252, 83): R=69, G=31, B=30
Pixel (252, 84): R=69, G=31, B=30
Pixel (252, 85): R=33, G=16, B=16
Pixel (252, 86): R=17, G=9, B=11
Pixel (252, 87): R=20, G=11, B=13
Pixel (252, 88): R=5, G=5, B=7
Pixel (252, 89): R=7, G=7, B=9
Pixel (252, 90): R=8, G=8, B=9
Pixel (252, 91): R=6, G=6, B=6
Pixel (252, 92): R=15, G=9, B=10
Pixel (252, 93): R=25, G=11, B=13
```



```
Pixel (252, 94): R=35, G=11, B=15
Pixel (252, 95): R=39, G=12, B=14
Pixel (252, 96): R=42, G=12, B=14
Pixel (252, 97): R=45, G=14, B=16
Pixel (252, 98): R=47, G=14, B=17
Pixel (252, 99): R=50, G=14, B=18
Pixel (252, 100): R=51, G=14, B=17
Pixel (252, 101): R=57, G=15, B=20
Pixel (252, 102): R=58, G=18, B=22
Pixel (252, 103): R=50, G=16, B=18
Pixel (252, 104): R=31, G=16, B=17
Pixel (252, 105): R=14, G=18, B=21
Pixel (252, 106): R=16, G=20, B=24
Pixel (252, 107): R=16, G=23, B=26
Pixel (252, 108): R=15, G=22, B=25
Pixel (252, 109): R=16, G=24, B=27
Pixel (252, 110): R=18, G=29, B=30
Pixel (252, 111): R=17, G=27, B=29
Pixel (252, 112): R=18, G=23, B=27
Pixel (252, 113): R=16, G=23, B=27
Pixel (252, 114): R=23, G=25, B=29
Pixel (252, 115): R=24, G=27, B=31
Pixel (252, 116): R=17, G=26, B=29
Pixel (252, 117): R=14, G=19, B=24
Pixel (252, 118): R=48, G=33, B=24
Pixel (252, 119): R=59, G=49, B=24
Pixel (252, 120): R=50, G=56, B=29
Pixel (252, 121): R=50, G=56, B=28
Pixel (252, 122): R=52, G=56, B=31
Pixel (252, 123): R=56, G=60, B=35
Pixel (252, 124): R=55, G=57, B=32
Pixel (252, 125): R=50, G=49, B=31
Pixel (252, 126): R=49, G=52, B=35
Pixel (252, 127): R=49, G=51, B=27
Pixel (252, 128): R=41, G=42, B=24
Pixel (252, 129): R=43, G=42, B=26
Pixel (252, 130): R=45, G=44, B=27
Pixel (252, 131): R=23, G=24, B=21
Pixel (252, 132): R=27, G=21, B=12
Pixel (252, 133): R=81, G=60, B=25
Pixel (252, 134): R=75, G=55, B=23
Pixel (252, 135): R=74, G=53, B=22
Pixel (252, 136): R=76, G=55, B=27
Pixel (252, 137): R=77, G=60, B=28
Pixel (252, 138): R=82, G=62, B=30
```

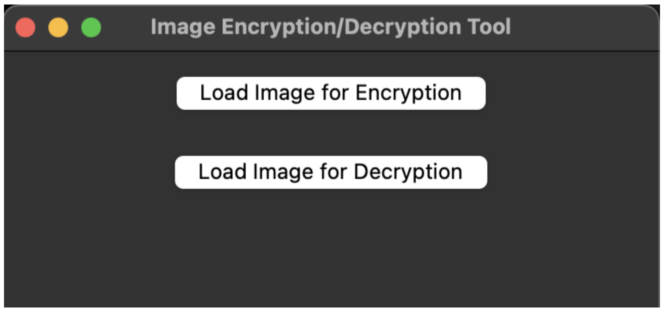
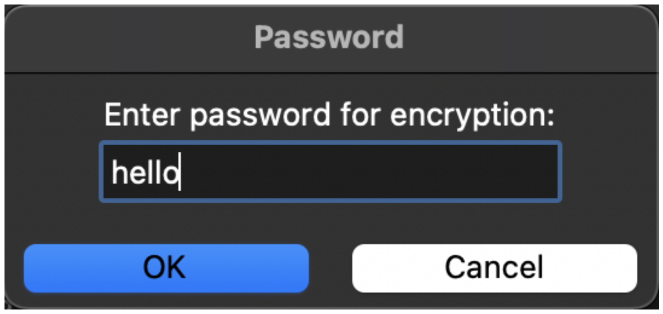


Image Encryption/Decryption Tool
Load Image for Encryption
Load Image for Decryption



Enter password for encryption:

hello

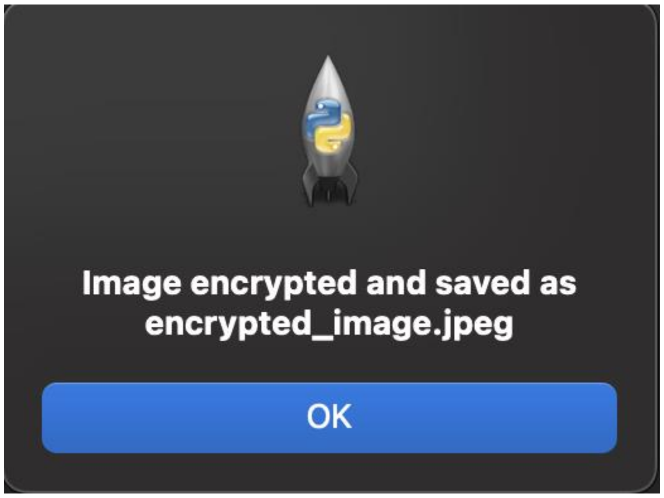


Image encrypted and saved as encrypted_image.jpeg

IMAGE USED FOR ENCRYPTION/DECRYPTION: Image is 1.1 MB

5.2 Analysis of Results

The project successfully implemented AES encryption and decryption for images, allowing secure transmission and preventing unauthorized access. During testing, the application achieved the following results:

- **Accuracy**: The encryption function converted the original image data into a completely unreadable format. The pixel data was encrypted and stored in a file (encrypted.jpeg), confirming that the AES algorithm successfully obscured the original information.
- **Security**: The encrypted file remained inaccessible and unreadable without the correct password. Attempts to open the encrypted file without decryption confirmed the data's security, demonstrating effective protection against unauthorized access.

- **Performance**: The encryption process was efficient, with completion times within the acceptable range for images under 5 MB. This efficiency ensures the tool can handle typical image file sizes without significant delays.

5.3 Comparison with Expected Results

Metric	Expected Result	Actual Result
Security	Encrypted image should be unreadable without password	Achieved; unauthorized access resulted in unreadable file
Decryption Accuracy	Decrypted image should match original pixel-by-pixel	Achieved; reconstructed image matched original data
Performance (Encryption)	Complete in < 10 seconds for images < 5 MB	Achieved; encryption was quick for test images
Performance (Decryption)	Complete in < 10 seconds for images < 5 MB	Achieved; decryption within expected time

6. Conclusion

We have successfully developed a program that encrypts and decrypts the image files accurately. This will help in minimising the problem of data theft and leaks of other sensitive information. The file that we obtained after encryption is very safe and no one can steal data from this file. So, this file can be sent on a network without worrying. Our developed solution is a small contribution that can be very helpful for military or medical fields in future times.
The report shows the study in which a system could be used for effective image data encryption and key generation in diversified application areas, where sensitive and confidential data needs to be transmitted along with the image. The next step in this direction will be system implementation, and then analyzing it for its efficiency, accuracy and reliability.

REFERENCES

1. Image Encryption Using AES Algorithm – Nevon Projects: **https://nevonprojects.com/image-encryption-using-aes-algorithm/**

2. https://www.iosrjournals.org/

3. https://www.researchgate.net/

4. https://citeseerx.ist.psu.edu/

5. https://www.ijser.org/

6. https://www.educative.io/edpresso/what-is-the-aes-algorithm

7. Paavni Gaur's et al. "AES Image Encryption". Available: https://www.researchgate.net/publication/357232938_AES_Image_Encryption

GitHub Link

(with Usage Guide & Report Copy)

https://github.com/Tanishta15/AES_image