IMAGE ENCRYPTION AND DECRYPTION USING AES ALGORITHM

PROJECT REPORT

Submitted by

Parth Galhotra(RA2111032010029)

Kumar Aniruddh Singh(RA2111032010032)

Under the Guidance of

Dr. Soumiya

Assistant Professor, Department of Networking and Communications

In partial Satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND TECHNOLOGY

with specialization in Internet of Things

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TCEHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203

NOVEMBER 2023

BONAFIDE

This is to certify that this project titled "**Image Encryption and Decryption using AES Algorithm**" is the bonafide work of **Parth Galhotra(RA2111032010029)** and **Kumar Aniruddh Singh(RA2111032010032)** who undertook the task of completing the project within the allotted time.

Signature of the Guide
Dr. Soumiya
Assistant Professor
Department of NWC
SRM Institute of Science and Technology

Signature of the II Year Academic Advisor
____
Professor and Head
Department of NWC
SRM Institute of Science and Technology

Abstract

Now a day the use of devices such as computer, mobile and many more other devices for communication as well as for data storage and transmission has increased. As a result, there is increase in number of users. Along with these users, there is also increase in number of unauthorized users which are trying to access a data by unfair means. This arises the problem of data security. Images are sent over an insecure transmission channel from different sources, some image data contains secret data, some images itself are highly confidential hence, securing them from any attack is essentially required.

To solve this problem, we are using AES algorithm for encrypting and decrypting image. This encrypted data is unreadable to the unauthorized user. This encrypted data can be sent over network and can be decrypted using AES at the receiving end. Hence it ensures secure transmission of image.

Aim of the project

The project aims to develop a secure transfer of images between sender and receiver. Image should be encrypted before it is sent on a network and it should be correctly decrypted on the receiver side.

Objective of the project

- Encryption of an Image to unreadable format
- Decryption of encrypted image to original image
- Secure transfer of an image over the network such as internet
- Ensure no modifications are made while transferring over the network.

Scope of the project

Product scope

The project works by encrypting the given image using AES algorithm so that this image can be sent securely over the network. At the receiver side, the receiver has code for decrypting the image so that he can get the original image. This helps in sending confidential and sensitive information securely over the internet. Main application of this can be very helpful in medical and military fields.

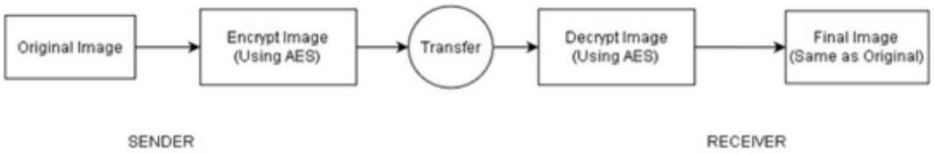

Original Image
Encrypt Image (Using AES)
Transfer
Decrypt Image (Using AES)
Final Image (Same as Original)
SENDER
RECEIVER

Figure 1

Design and Implementation constraints

• Python must be used for front end

• Encryption and Decryption should be done using AES algorithm

• Original Image must be in .jpeg/.png format.

Assumptions and Dependencies

• Sender and Receiver are connected on a network

Functional Requirements

- The system shall encrypt the given image to an unreadable format. This is done using AES encryption function.
- The system shall decrypt the received encrypted image to a readable format. This is done using AES decryption function. The output image should be same as the original image.
- The system ensures that the image is securely sent over any transmission medium. Third party system cannot make modifications to the file being sent since unauthorised access is not supported.

Non-Functional Requirements

Performance Requirements

- For smooth & efficient encryption, image size must be less than 5MB.
- Decryption should not take more than 10 seconds.

Safety and Security Requirements

- If the decryption takes more than 10 seconds, then discard the message (because the message might have been corrupted during transmission) and ask sender to re-send it.
- Encryption is done using encryption key. Decryption will happen only when same encryption key is used at the receiver side.

Software Quality Attributes

Reliability

External factors do not affect the system. AES algorithm is universally accepted and generates consistent results therefore there are very less chances of errors. Error can occur only if there is a transmission glitch (the probability of which is very rare). So, the system is reliable.

Usability

It uses python based GUI which is user friendly and provides buttons for easy navigation. A person with basic understanding of computer can easily use this software for encrypting/decrypting image using key.

Testability

The system is easy to test and find defects. The system is divided into different modules performing specific functions that can be tested individually.

External Interface Requirements

User Interfaces

The interface window gives us a box for entering the location of image. Along with this box, it also contains two buttons for encoding and decoding the image. After clicking on one of these button, next window has a textbox to enter the password and a submit button. After submitting, it shows the filename of new image file generated.

Hardware Interfaces

- Sender Computer: for seeing the original and encrypted image
- Receiver Computer: for seeing the decrypted image

Software Interfaces

We have frontend made using python module named Tkinter(). This provides an interactive window for the user. The user need to provide the location of the image to be encoded/decoded. After this, user can either go for encoding or decoding the image. It asks for a password, which is basically your encryption key. After these details are entered, we use the python functions declared in the code to encode/decode the image using AES file from crypto.cypher module of python. When this process is done, the user will get encrypted/decrypted image as output which will be saved in the same directory as input image file.

Work Breakdown Structure

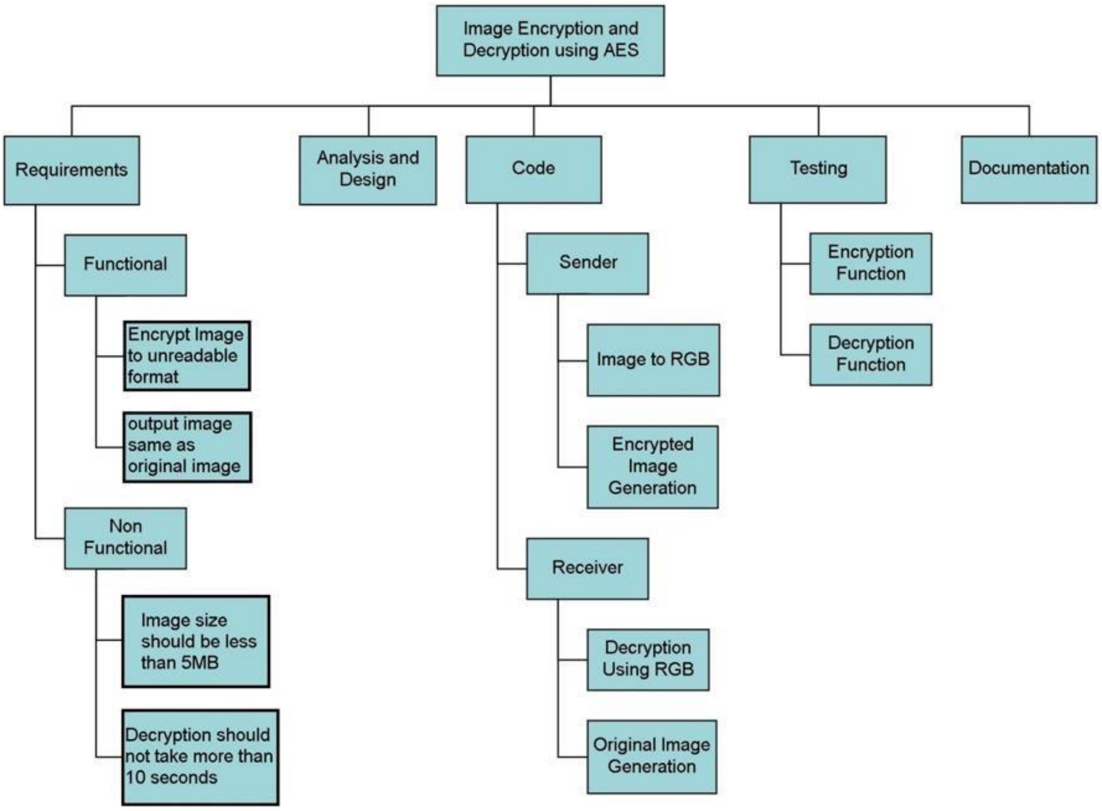
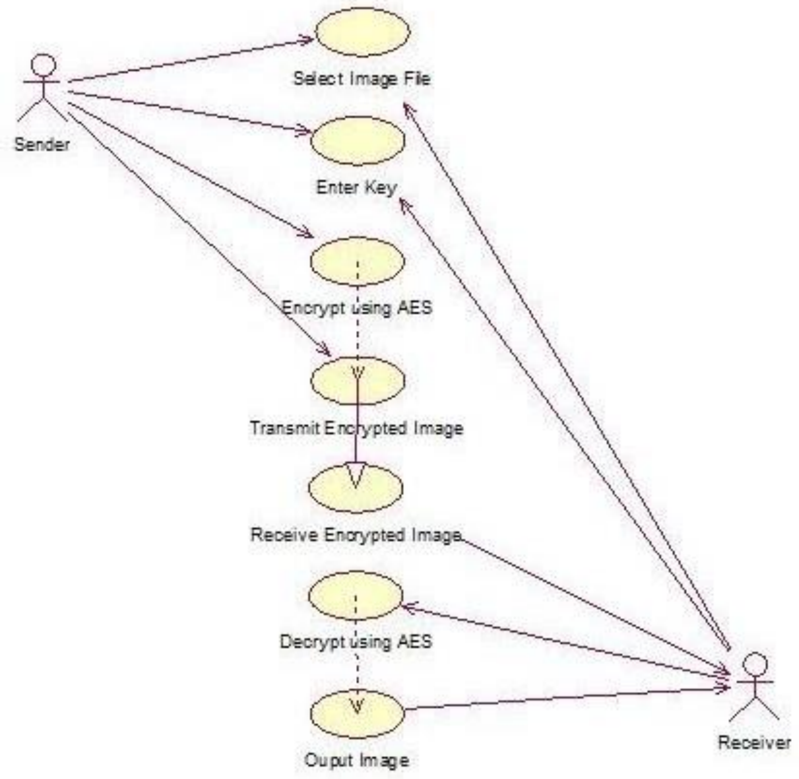


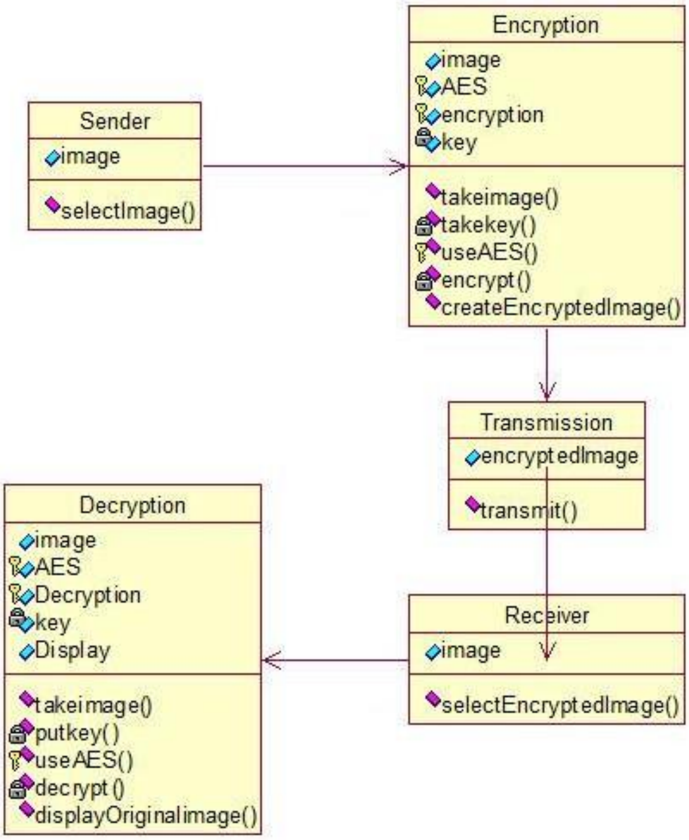
Image Encryption and Decryption using AES
Requirements
Analysis and Design
Code
Testing
Documentation
Functional
Encrypt Image to unreadable format
output image same as original image
Non Functional
Image size should be less than 5MB
Decryption should not take more than 10 seconds
Sender
Image to RGB
Encrypted Image Generation
Receiver
Decryption Using RGB
Original Image Generation
Encryption Function
Decryption Function

Design Diagrams

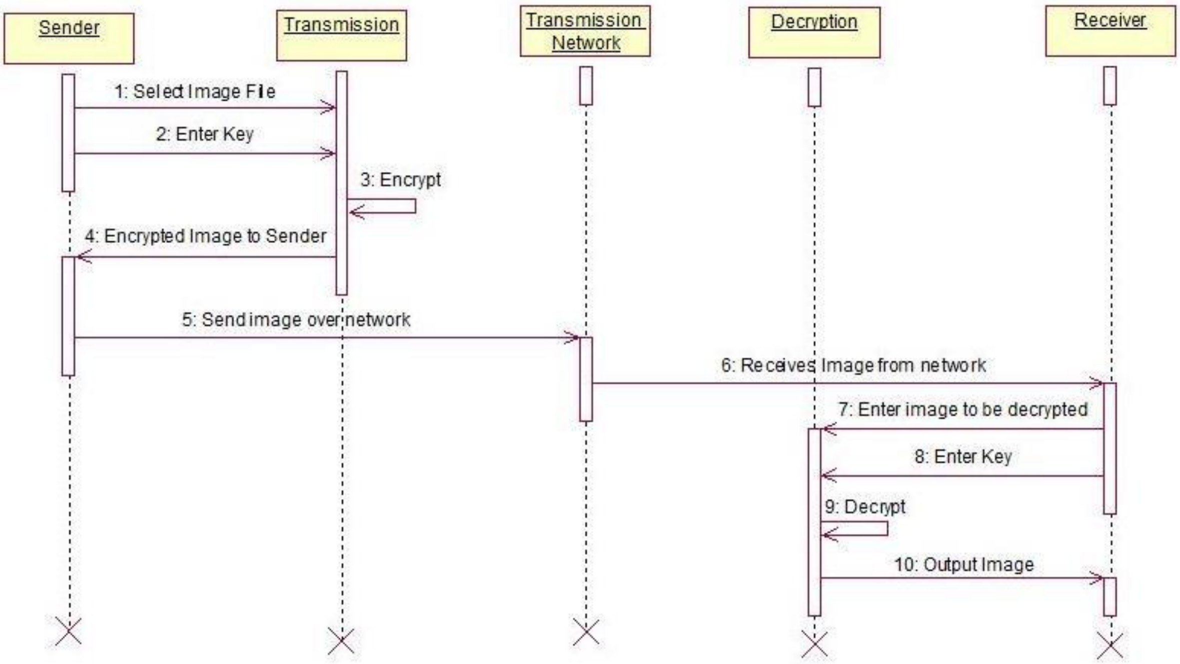
Usecase Diagram:



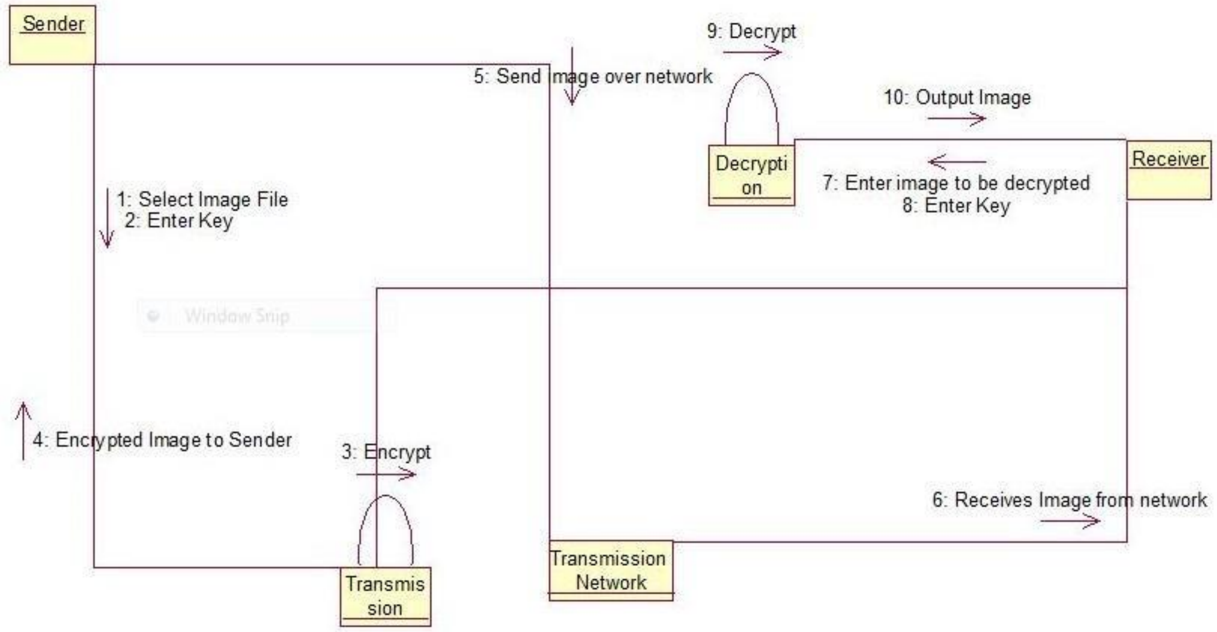
Class Diagram:



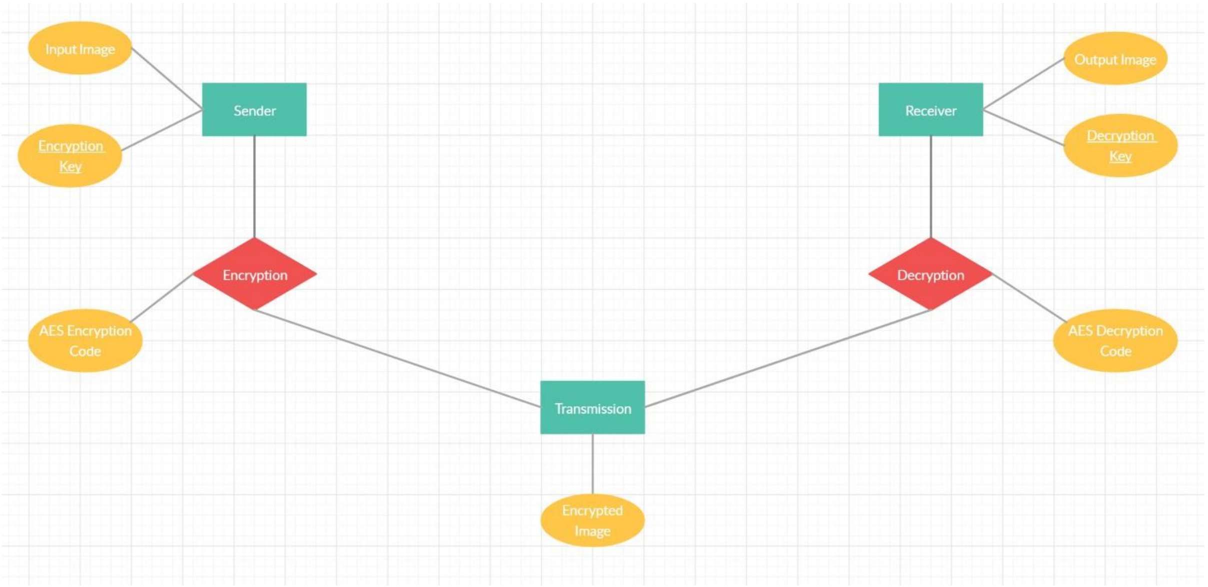
Sequence Diagram:



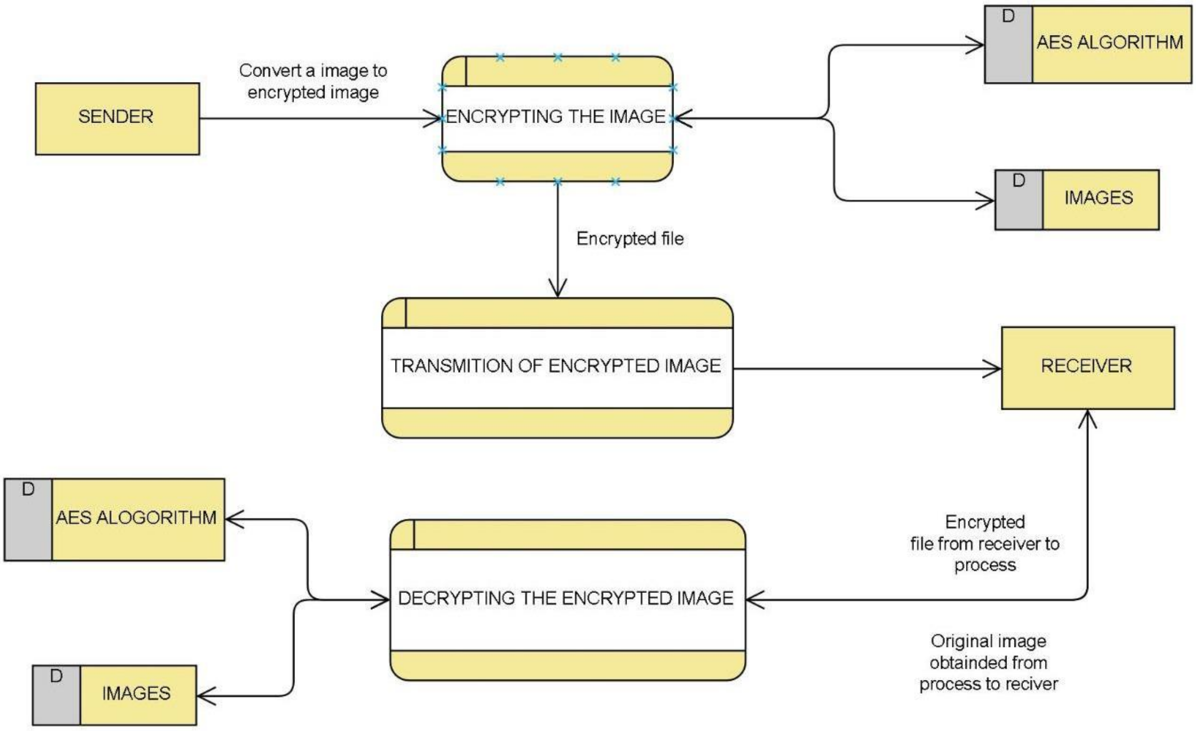
Collaboration Diagram:



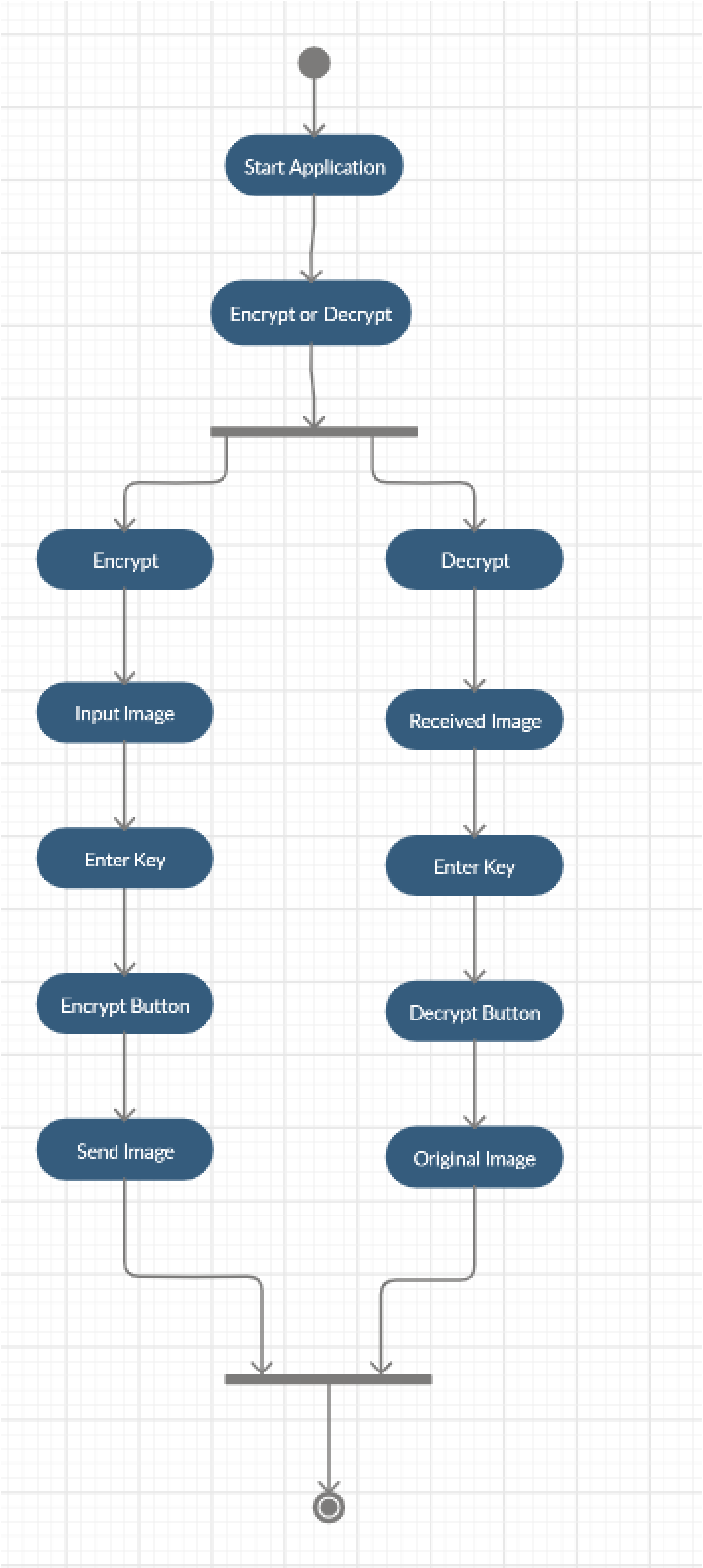
ER Diagram:



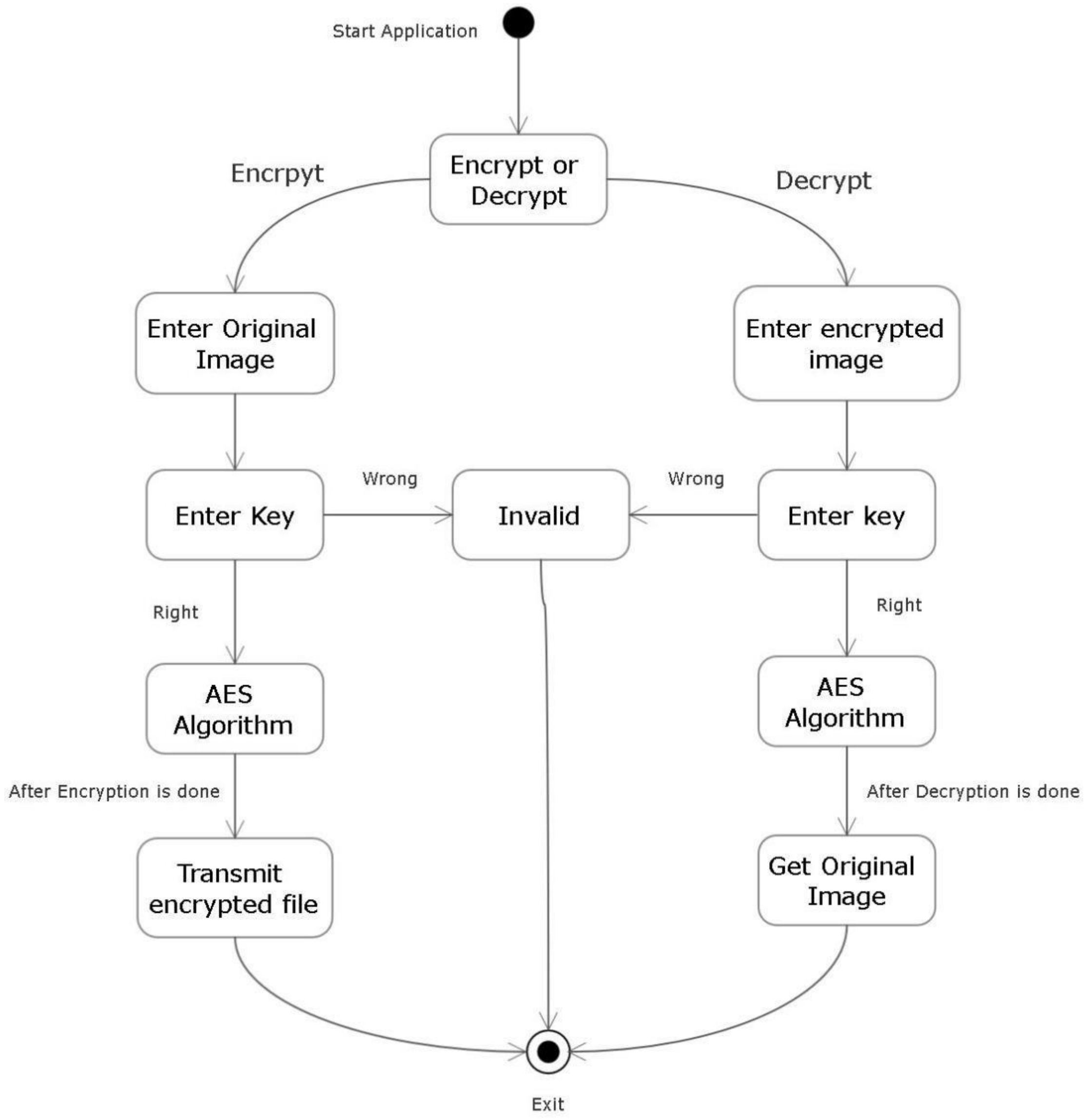
Dataflow Diagram:



Activity Diagram:

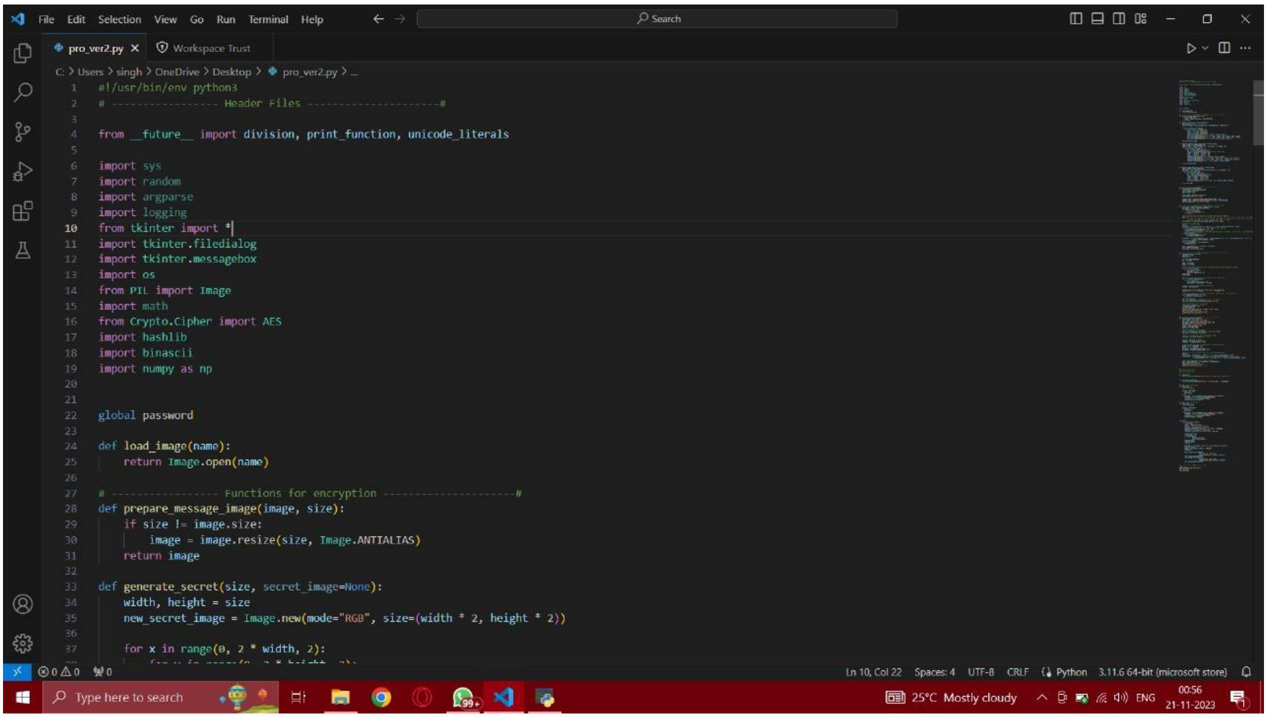


State Chart Diagram:

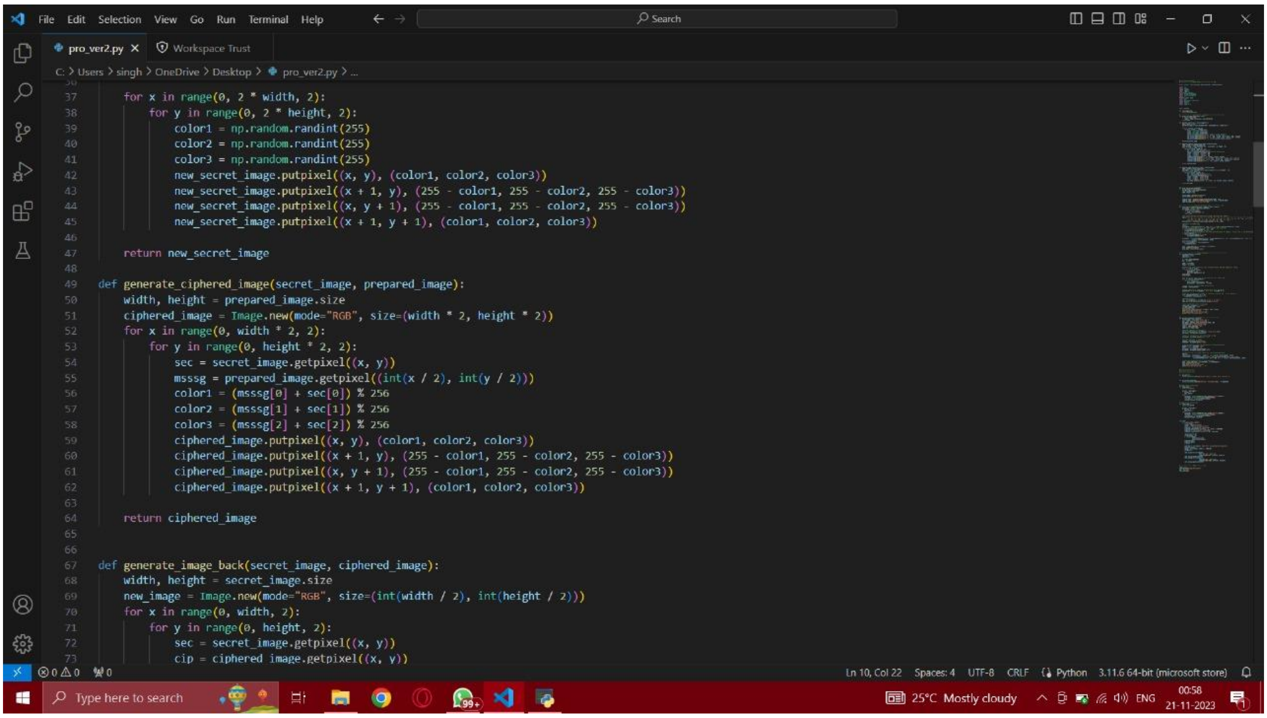


Screenshot of Implementation

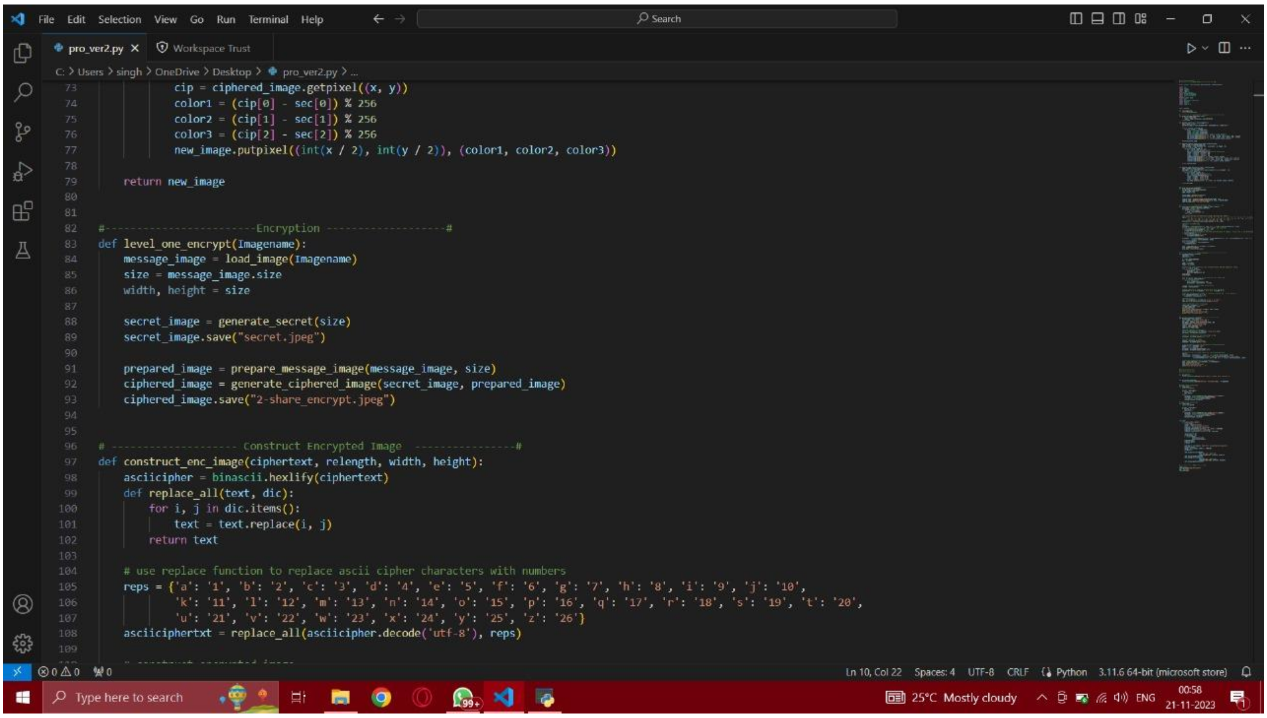
Code



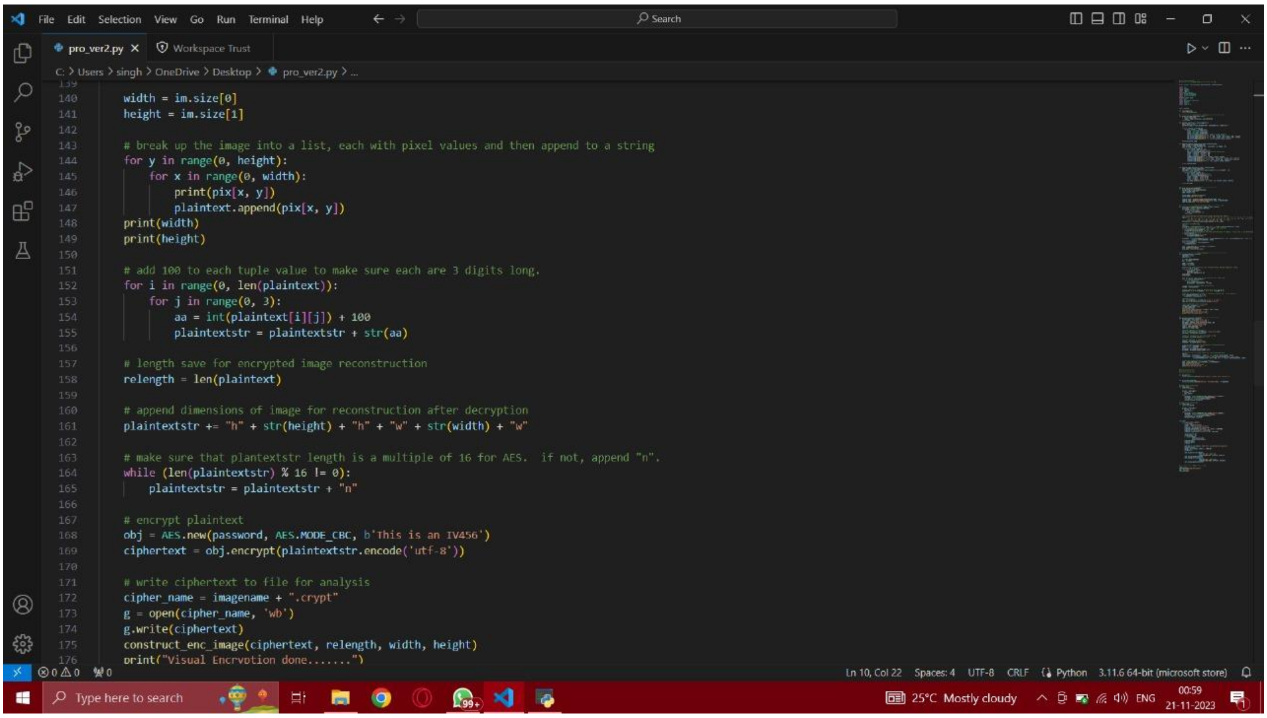
```python
#!/usr/bin/env python3
# ---------------- Header Files -------------------#

from __future__ import division, print_function, unicode_literals

import sys
import random
import argparse
import logging
from tkinter import *
import tkinter.filedialog
import tkinter.messagebox
import os
from PIL import Image
import math
from Crypto.Cipher import AES
import hashlib
import binascii
import numpy as np


global password

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

# ---------------- Functions for encryption -------------------#
def prepare_message_image(image, size):
    if size != image.size:
        image = image.resize(size, Image.ANTIALIAS)
    return image

def generate_secret(size, secret_image=None):
    width, height = size
    new_secret_image = Image.new(mode="RGB", size=(width * 2, height * 2))

    for x in range(0, 2 * width, 2):
```



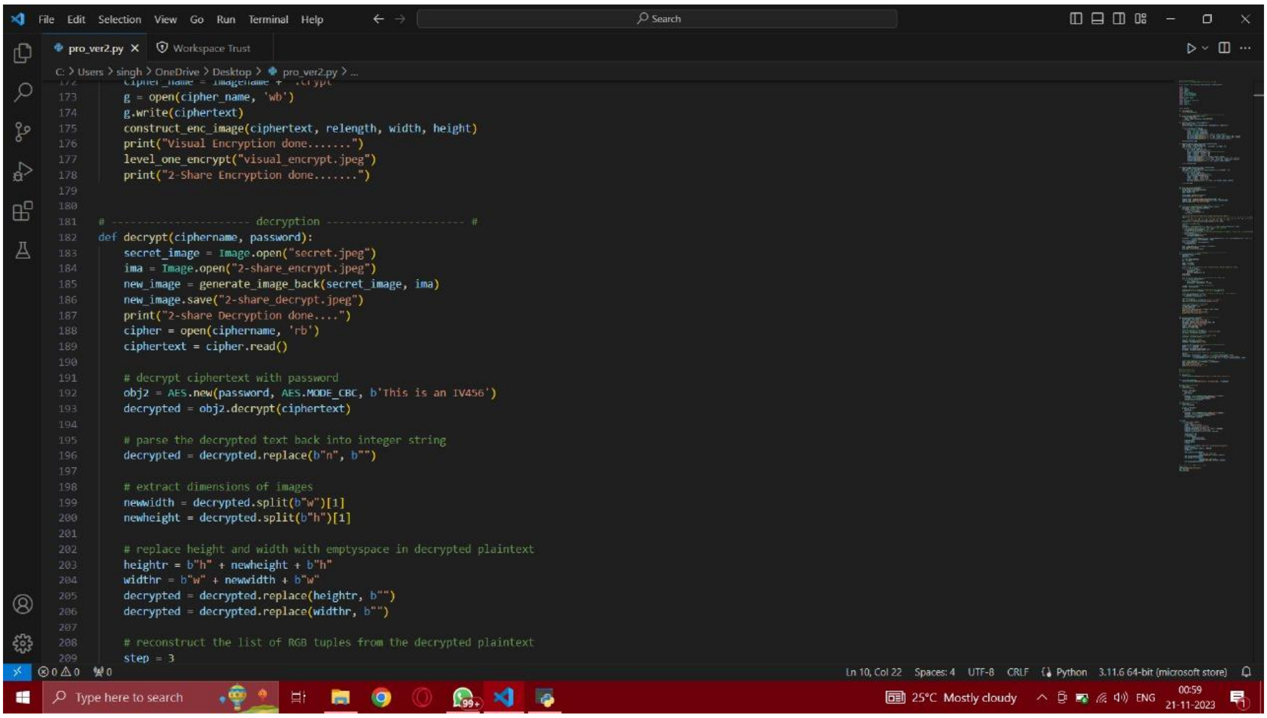
```python
    for x in range(0, 2 * width, 2):
        for y in range(0, 2 * height, 2):
            color1 = np.random.randint(255)
            color2 = np.random.randint(255)
            color3 = np.random.randint(255)
            new_secret_image.putpixel((x, y), (color1, color2, color3))
            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 generate_ciphered_image(secret_image, prepared_image):
    width, height = prepared_image.size
    ciphered_image = Image.new(mode="RGB", size=(width * 2, height * 2))
    for x in range(0, width * 2, 2):
        for y in range(0, height * 2, 2):
            sec = secret_image.getpixel((x, y))
            msssg = prepared_image.getpixel((int(x / 2), int(y / 2)))
            color1 = (msssg[0] + sec[0]) % 256
            color2 = (msssg[1] + sec[1]) % 256
            color3 = (msssg[2] + sec[2]) % 256
            ciphered_image.putpixel((x, y), (color1, color2, color3))
            ciphered_image.putpixel((x + 1, y), (255 - color1, 255 - color2, 255 - color3))
            ciphered_image.putpixel((x, y + 1), (255 - color1, 255 - color2, 255 - color3))
            ciphered_image.putpixel((x + 1, y + 1), (color1, color2, color3))

    return ciphered_image


def generate_image_back(secret_image, ciphered_image):
    width, height = secret_image.size
    new_image = Image.new(mode="RGB", size=(int(width / 2), int(height / 2)))
    for x in range(0, width, 2):
        for y in range(0, height, 2):
            sec = secret_image.getpixel((x, y))
            cip = ciphered_image.getpixel((x, y))
```



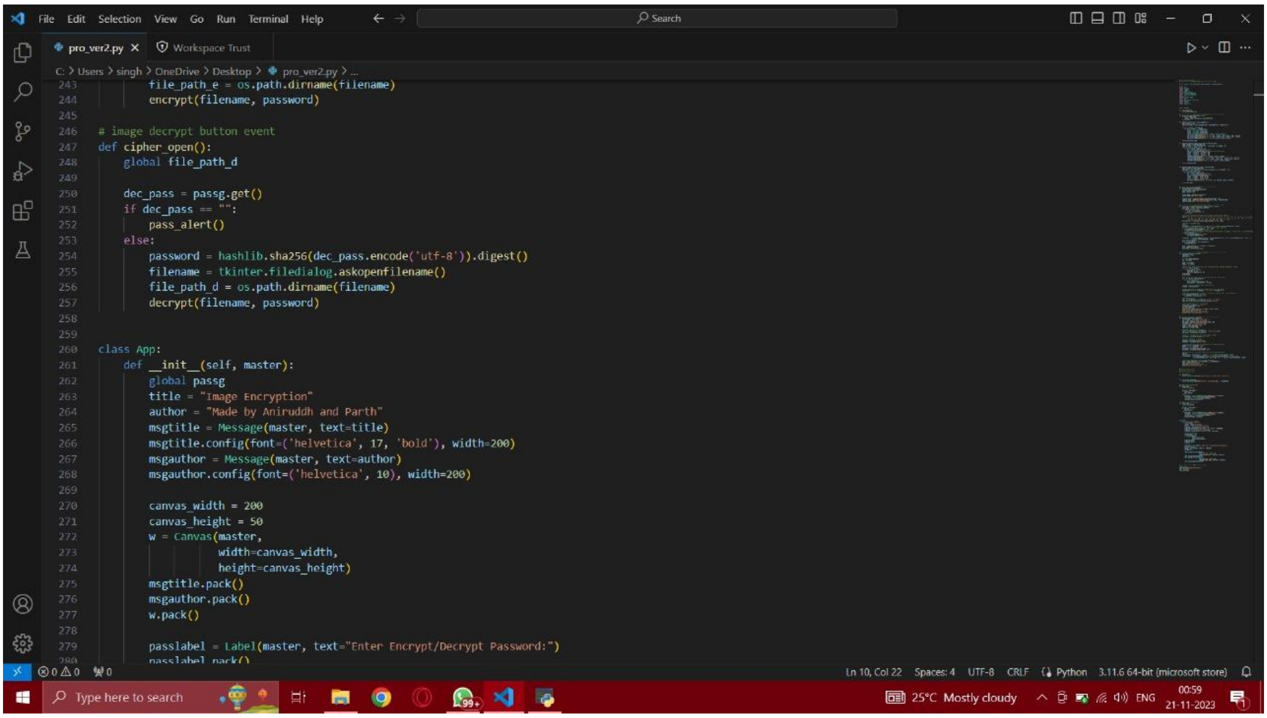
```python
            cip = ciphered_image.getpixel((x, y))
            color1 = (cip[0] - sec[0]) % 256
            color2 = (cip[1] - sec[1]) % 256
            color3 = (cip[2] - sec[2]) % 256
            new_image.putpixel((int(x / 2), int(y / 2)), (color1, color2, color3))

    return new_image


#-----------------------Encryption ------------------#
def level_one_encrypt(Imagename):
    message_image = load_image(Imagename)
    size = message_image.size
    width, height = size

    secret_image = generate_secret(size)
    secret_image.save("secret.jpeg")

    prepared_image = prepare_message_image(message_image, size)
    ciphered_image = generate_ciphered_image(secret_image, prepared_image)
    ciphered_image.save("2-share_encrypt.jpeg")


# -------------------- Construct Encrypted Image  ----------------#
def construct_enc_image(ciphertext, relength, width, height):
    asciicipher = binascii.hexlify(ciphertext)
    def replace_all(text, dic):
        for i, j in dic.items():
            text = text.replace(i, j)
        return text

    # use replace function to replace ascii cipher characters with numbers
    reps = {'a': '1', 'b': '2', 'c': '3', 'd': '4', 'e': '5', 'f': '6', 'g': '7', 'h': '8', 'i': '9', 'j': '10',
            'k': '11', 'l': '12', 'm': '13', 'n': '14', 'o': '15', 'p': '16', 'q': '17', 'r': '18', 's': '19', 't': '20',
            'u': '21', 'v': '22', 'w': '23', 'x': '24', 'y': '25', 'z': '26'}
    asciiciphertxt = replace_all(asciicipher.decode('utf-8'), reps)

```



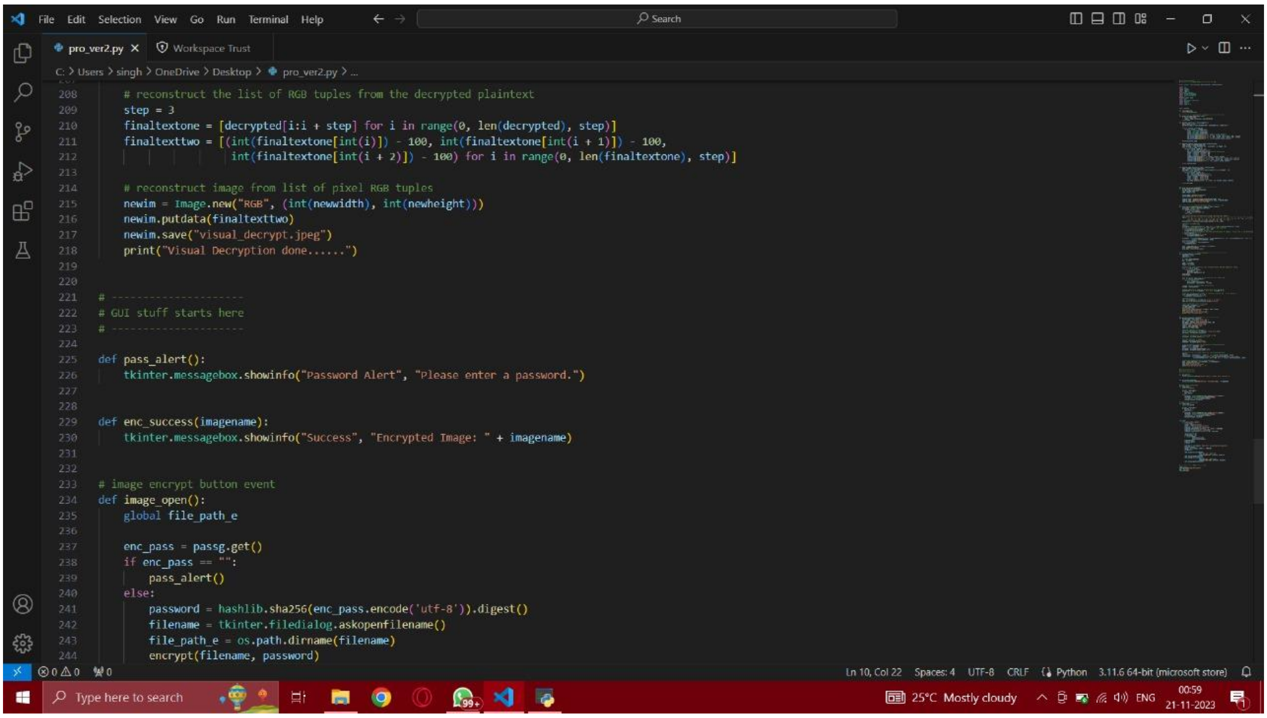
```python
    width = im.size[0]
    height = im.size[1]

    # break up the image into a list, each with pixel values and then append to a string
    for y in range(0, height):
        for x in range(0, width):
            print(pix[x, y])
            plaintext.append(pix[x, y])
    print(width)
    print(height)

    # add 100 to each tuple value to make sure each are 3 digits long.
    for i in range(0, len(plaintext)):
        for j in range(0, 3):
            aa = int(plaintext[i][j]) + 100
            plaintextstr = plaintextstr + str(aa)

    # length save for encrypted image reconstruction
    relength = len(plaintext)

    # append dimensions of image for reconstruction after decryption
    plaintextstr += "h" + str(height) + "h" + "w" + str(width) + "w"

    # make sure that plantextstr length is a multiple of 16 for AES.  if not, append "n".
    while (len(plaintextstr) % 16 != 0):
        plaintextstr = plaintextstr + "n"

    # encrypt plaintext
    obj = AES.new(password, AES.MODE_CBC, b'This is an IV456')
    ciphertext = obj.encrypt(plaintextstr.encode('utf-8'))

    # write ciphertext to file for analysis
    cipher_name = imagename + ".crypt"
    g = open(cipher_name, 'wb')
    g.write(ciphertext)
    construct_enc_image(ciphertext, relength, width, height)
    print("Visual Encryption done.......")
```


File Edit Selection View Go Run Terminal Help
Search
pro_ver2.py
Workspace Trust
C: > Users > singh > OneDrive > Desktop > pro_ver2.py > ...
Ln 10, Col 22 Spaces: 4 UTF-8 CRLF Python 3.11.6 64-bit (microsoft store)
Type here to search
25°C Mostly cloudy
ENG
00:59
21-11-2023

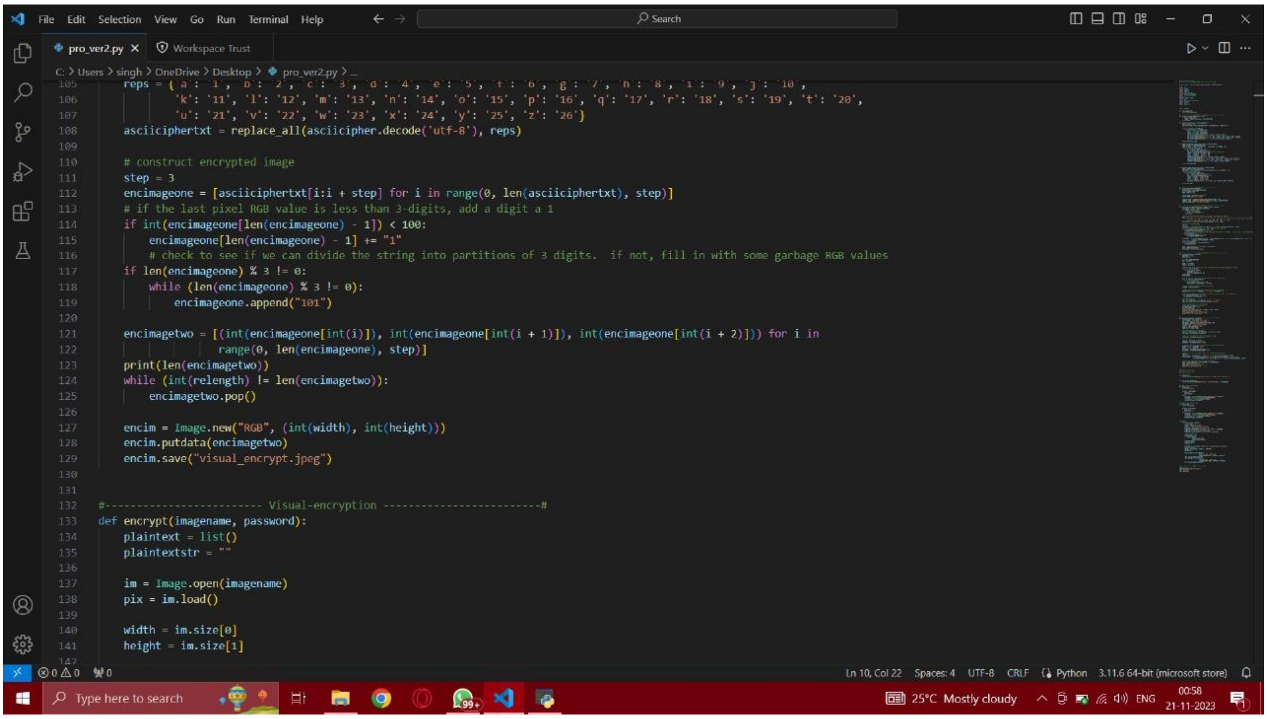
```
    cipher_name = imagename + ".crypt"
    g = open(cipher_name, 'wb')
    g.write(ciphertext)
    construct_enc_image(ciphertext, relength, width, height)
    print("Visual Encryption done.......")
    level_one_encrypt("visual_encrypt.jpeg")
    print("2-Share Encryption done.......")


# --------------------- decryption --------------------- #
def decrypt(ciphername, password):
    secret_image = Image.open("secret.jpeg")
    ima = Image.open("2-share_encrypt.jpeg")
    new_image = generate_image_back(secret_image, ima)
    new_image.save("2-share_decrypt.jpeg")
    print("2-share Decryption done....")
    cipher = open(ciphername, 'rb')
    ciphertext = cipher.read()

    # decrypt ciphertext with password
    obj2 = AES.new(password, AES.MODE_CBC, b'This is an IV456')
    decrypted = obj2.decrypt(ciphertext)

    # parse the decrypted text back into integer string
    decrypted = decrypted.replace(b"n", b"")

    # extract dimensions of images
    newwidth = decrypted.split(b"w")[1]
    newheight = decrypted.split(b"h")[1]

    # replace height and width with emptyspace in decrypted plaintext
    heightr = b"h" + newheight + b"h"
    widthr = b"w" + newwidth + b"w"
    decrypted = decrypted.replace(heightr, b"")
    decrypted = decrypted.replace(widthr, b"")

    # reconstruct the list of RGB tuples from the decrypted plaintext
    step = 3
```


File Edit Selection View Go Run Terminal Help
Search
pro_ver2.py
Workspace Trust
C: > Users > singh > OneDrive > Desktop > pro_ver2.py > ...
Ln 10, Col 22 Spaces: 4 UTF-8 CRLF Python 3.11.6 64-bit (microsoft store)
Type here to search
25°C Mostly cloudy
ENG
00:59
21-11-2023

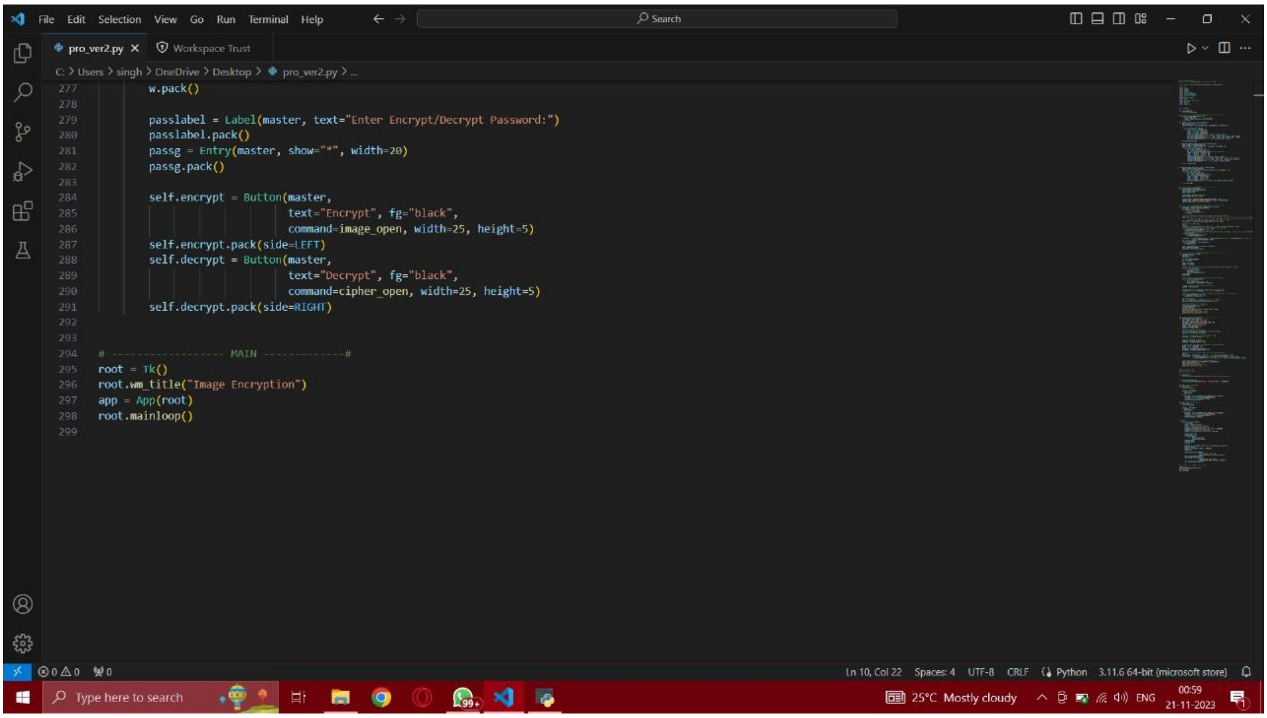
```
        file_path_e = os.path.dirname(filename)
        encrypt(filename, password)

# image decrypt button event
def cipher_open():
    global file_path_d

    dec_pass = passg.get()
    if dec_pass == "":
        pass_alert()
    else:
        password = hashlib.sha256(dec_pass.encode('utf-8')).digest()
        filename = tkinter.filedialog.askopenfilename()
        file_path_d = os.path.dirname(filename)
        decrypt(filename, password)


class App:
    def __init__(self, master):
        global passg
        title = "Image Encryption"
        author = "Made by Aniruddh and Parth"
        msgtitle = Message(master, text=title)
        msgtitle.config(font=('helvetica', 17, 'bold'), width=200)
        msgauthor = Message(master, text=author)
        msgauthor.config(font=('helvetica', 10), width=200)

        canvas_width = 200
        canvas_height = 50
        w = Canvas(master,
                   width=canvas_width,
                   height=canvas_height)
        msgtitle.pack()
        msgauthor.pack()
        w.pack()

        passlabel = Label(master, text="Enter Encrypt/Decrypt Password:")
        passlabel.pack()
```


File Edit Selection View Go Run Terminal Help
Search
pro_ver2.py
Workspace Trust
C: > Users > singh > OneDrive > Desktop > pro_ver2.py > ...
Ln 10, Col 22 Spaces: 4 UTF-8 CRLF Python 3.11.6 64-bit (microsoft store)
Type here to search
25°C Mostly cloudy ENG 00:59 21-11-2023

```python
    # reconstruct the list of RGB tuples from the decrypted plaintext
    step = 3
    finaltextone = [decrypted[i:i + step] for i in range(0, len(decrypted), step)]
    finaltexttwo = [(int(finaltextone[int(i)]) - 100, int(finaltextone[int(i + 1)]) - 100,
                     int(finaltextone[int(i + 2)]) - 100) for i in range(0, len(finaltextone), step)]

    # reconstruct image from list of pixel RGB tuples
    newim = Image.new("RGB", (int(newwidth), int(newheight)))
    newim.putdata(finaltexttwo)
    newim.save("visual_decrypt.jpeg")
    print("Visual Decryption done......")


# --------------------
# GUI stuff starts here
# --------------------

def pass_alert():
    tkinter.messagebox.showinfo("Password Alert", "Please enter a password.")


def enc_success(imagename):
    tkinter.messagebox.showinfo("Success", "Encrypted Image: " + imagename)


# image encrypt button event
def image_open():
    global file_path_e

    enc_pass = passg.get()
    if enc_pass == "":
        pass_alert()
    else:
        password = hashlib.sha256(enc_pass.encode('utf-8')).digest()
        filename = tkinter.filedialog.askopenfilename()
        file_path_e = os.path.dirname(filename)
        encrypt(filename, password)
```


File Edit Selection View Go Run Terminal Help
Search
pro_ver2.py
Workspace Trust
C: > Users > singh > OneDrive > Desktop > pro_ver2.py > ...
Ln 10, Col 22 Spaces: 4 UTF-8 CRLF Python 3.11.6 64-bit (microsoft store)
Type here to search
25°C Mostly cloudy ENG 00:58 21-11-2023

```python
    reps = {'a': '1', 'b': '2', 'c': '3', 'd': '4', 'e': '5', 'f': '6', 'g': '7', 'h': '8', 'i': '9', 'j': '10',
            'k': '11', 'l': '12', 'm': '13', 'n': '14', 'o': '15', 'p': '16', 'q': '17', 'r': '18', 's': '19', 't': '20',
            'u': '21', 'v': '22', 'w': '23', 'x': '24', 'y': '25', 'z': '26'}
    asciiciphertxt = replace_all(asciicipher.decode('utf-8'), reps)

    # construct encrypted image
    step = 3
    encimageone = [asciiciphertxt[i:i + step] for i in range(0, len(asciiciphertxt), step)]
    # if the last pixel RGB value is less than 3-digits, add a digit a 1
    if int(encimageone[len(encimageone) - 1]) < 100:
        encimageone[len(encimageone) - 1] += "1"
        # check to see if we can divide the string into partitions of 3 digits.  if not, fill in with some garbage RGB values
    if len(encimageone) % 3 != 0:
        while (len(encimageone) % 3 != 0):
            encimageone.append("101")

    encimagetwo = [(int(encimageone[int(i)]), int(encimageone[int(i + 1)]), int(encimageone[int(i + 2)])) for i in
                   range(0, len(encimageone), step)]
    print(len(encimagetwo))
    while (int(relength) != len(encimagetwo)):
        encimagetwo.pop()

    encim = Image.new("RGB", (int(width), int(height)))
    encim.putdata(encimagetwo)
    encim.save("visual_encrypt.jpeg")


#----------------------- Visual-encryption -----------------------#
def encrypt(imagename, password):
    plaintext = list()
    plaintextstr = ""

    im = Image.open(imagename)
    pix = im.load()

    width = im.size[0]
    height = im.size[1]
```



File Edit Selection View Go Run Terminal Help
pro_ver2.py
Workspace Trust
C: > Users > singh > OneDrive > Desktop > pro_ver2.py > ...
Ln 10, Col 22 Spaces: 4 UTF-8 CRLF Python 3.11.6 64-bit (microsoft store)
Type here to search
25°C Mostly cloudy
ENG
00:59
21-11-2023

```python
        w.pack()

        passlabel = Label(master, text="Enter Encrypt/Decrypt Password:")
        passlabel.pack()
        passg = Entry(master, show="*", width=20)
        passg.pack()

        self.encrypt = Button(master,
                              text="Encrypt", fg="black",
                              command=image_open, width=25, height=5)
        self.encrypt.pack(side=LEFT)
        self.decrypt = Button(master,
                              text="Decrypt", fg="black",
                              command=cipher_open, width=25, height=5)
        self.decrypt.pack(side=RIGHT)


# ------------------ MAIN -------------#
root = Tk()
root.wm_title("Image Encryption")
app = App(root)
root.mainloop()
```

Application’s first screen asking for key:

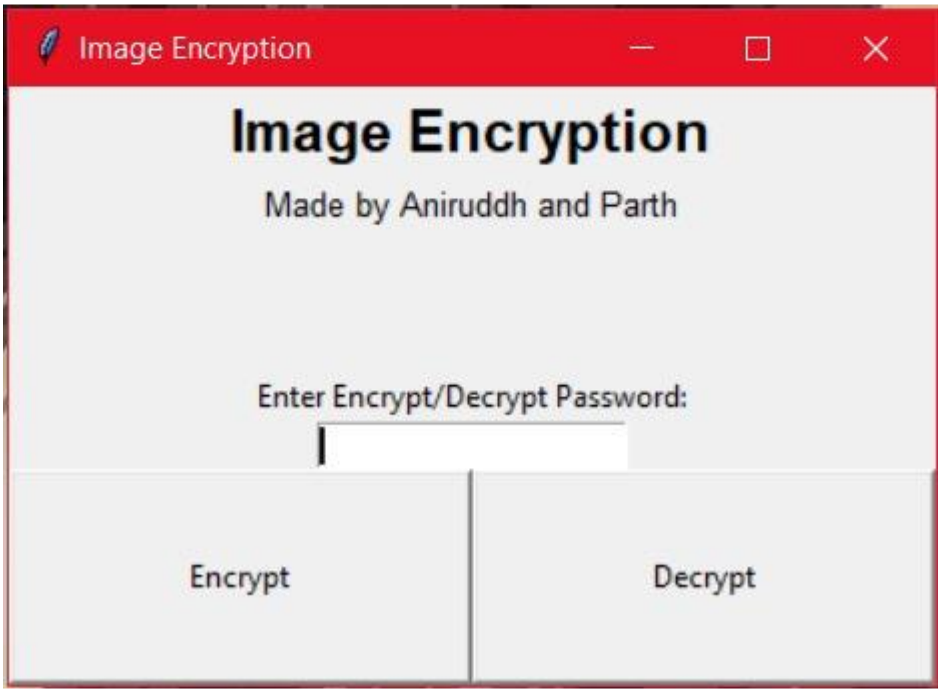
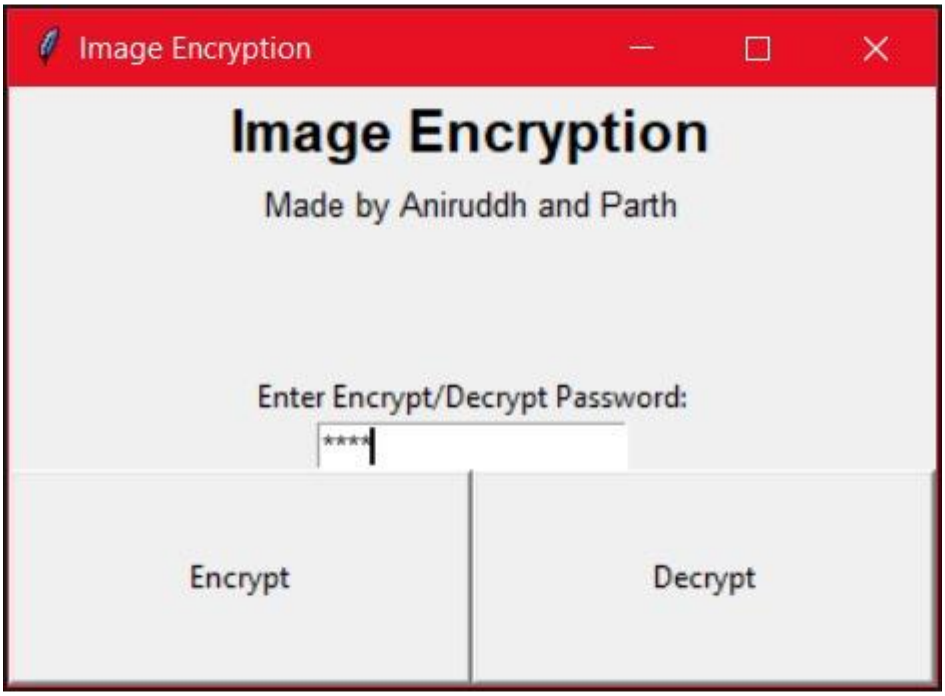


Image Encryption

Made by Aniruddh and Parth

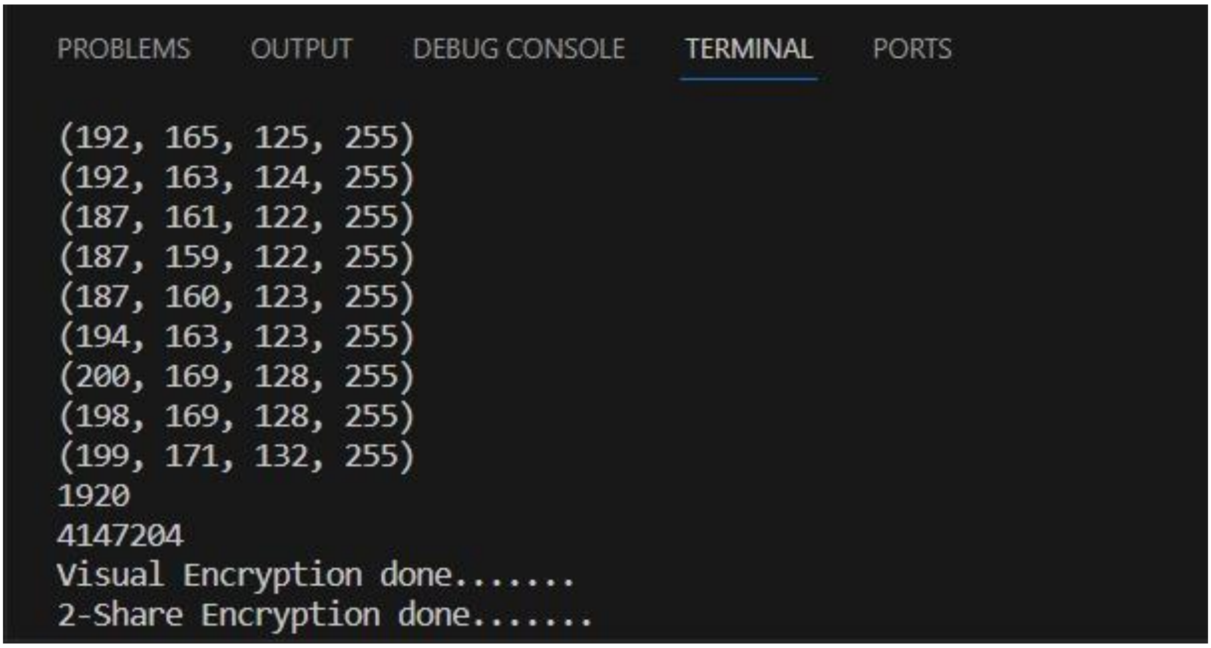
Enter Encrypt/Decrypt Password:

Key is entered:



After clicking Encrypt button:

Select the directory and then the original file to be encrypted

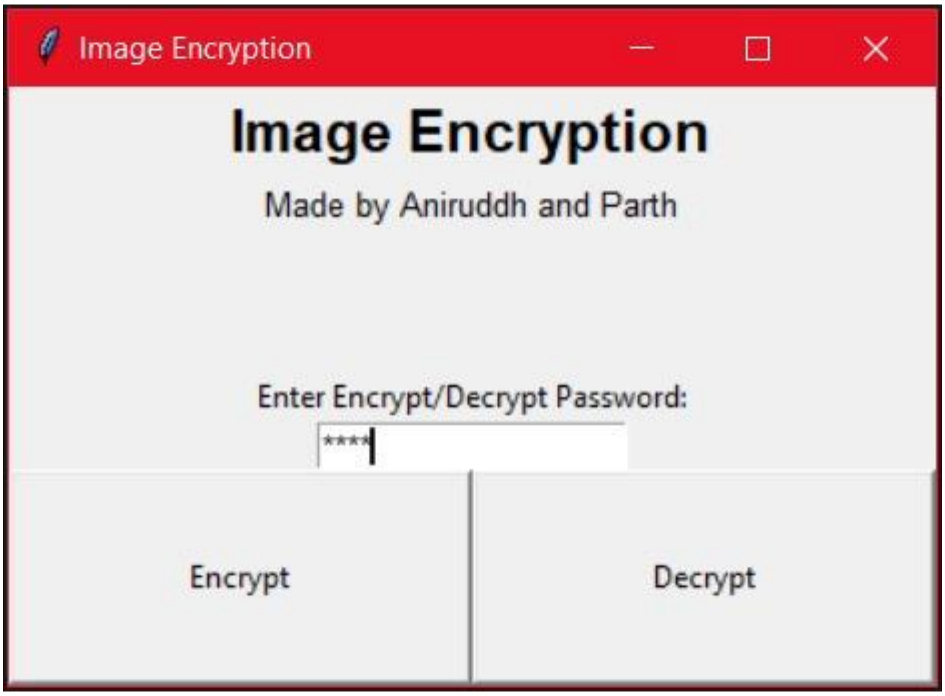


```
(192, 165, 125, 255)
(192, 163, 124, 255)
(187, 161, 122, 255)
(187, 159, 122, 255)
(187, 160, 123, 255)
(194, 163, 123, 255)
(200, 169, 128, 255)
(198, 169, 128, 255)
(199, 171, 132, 255)
1920
4147204
Visual Encryption done.......
2-Share Encryption done.......
```

The Encryption is completed and a file named ‘images.jpeg.crypt’ is created.

This is our Encrypted file.

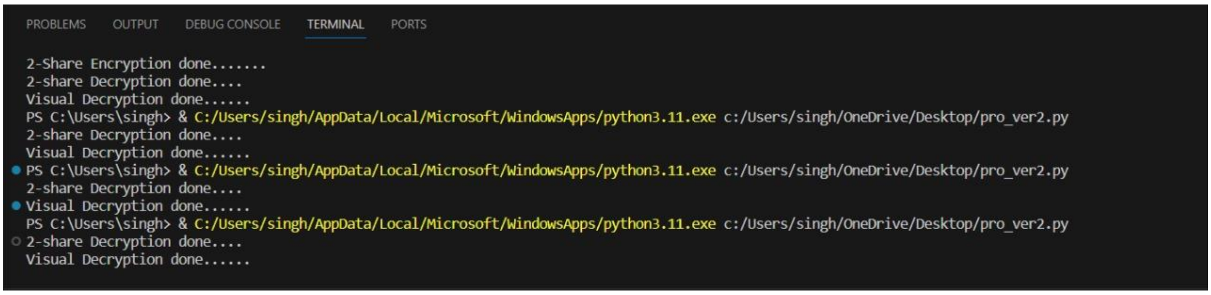
For decryption, enter same key:



After clicking decryption button:

Select the same directory and then select the file named ‘images.jpeg.crypt

Terminal and desktop screen after clicking ‘open’:



```
2-Share Encryption done.......
2-share Decryption done....
Visual Decryption done......
PS C:\Users\singh> & C:/Users/singh/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/singh/OneDrive/Desktop/pro_ver2.py
2-share Decryption done....
Visual Decryption done......
PS C:\Users\singh> & C:/Users/singh/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/singh/OneDrive/Desktop/pro_ver2.py
2-share Decryption done....
Visual Decryption done......
PS C:\Users\singh> & C:/Users/singh/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/singh/OneDrive/Desktop/pro_ver2.py
2-share Decryption done....
Visual Decryption done......
```

Decryption is complete and file named ‘visual_decrypt.jpeg’ is the decrypted file

Conclusion

We have successfully developed a program that encrypts and decrypts the image files accurately. This will help in minimizing 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.