SOFTWARE ENGINEERING

IMAGE ENCRYPTION AND DECRYPTION USING AES ALGORITHM

Group Member Names

18BCE0582 – Aditya Ruhatiya

18BCE0586 – Aditya Agrawal

Faculty Name

Dr. AKILA VICTOR

Associate Professor, SCOPE, VIT Vellore

Winter Semester

Dec 2019 – June 2020

VIT®

Vellore Institute of Technology

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

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.

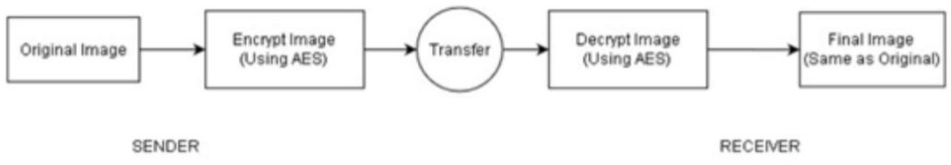

Original Image
Encrypt Image (Using AES)
Transfer
Decrypt Image (Using AES)
Final Image (Same as Original)
SENDER
RECEIVER

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

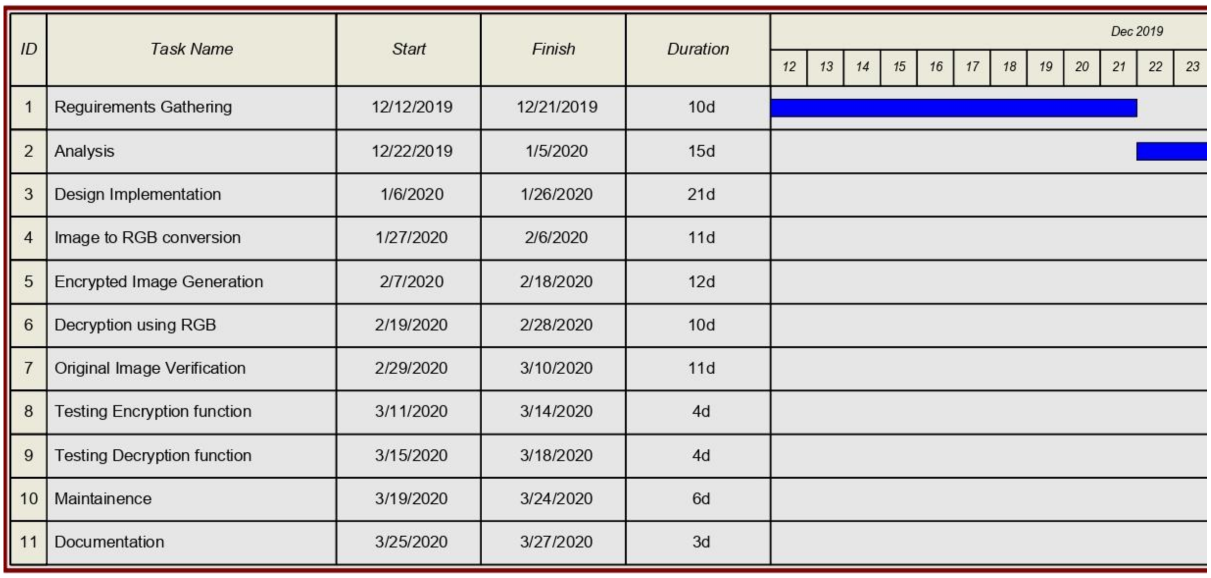
- Sender Computer: for seeing the original and encrypted image
- Receiver Computer: for seeing the decrypted image

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.

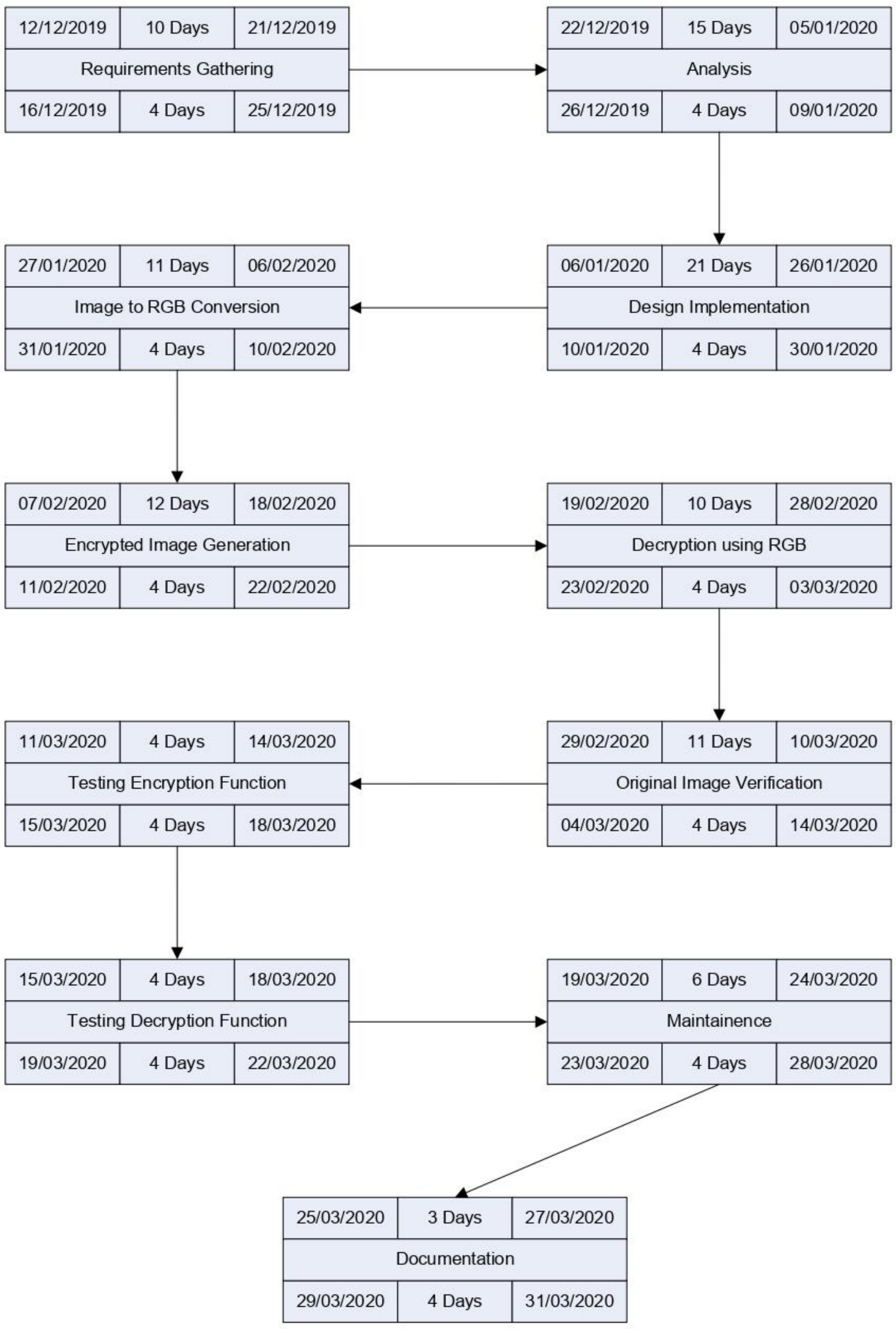
Scheduling Diagrams

Gantt Chart

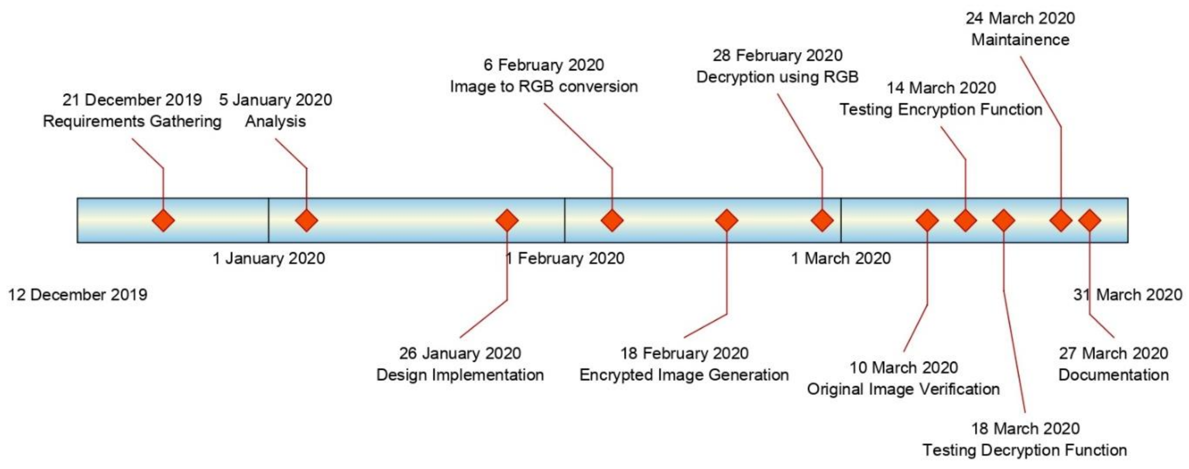


ID	Task Name	Start	Finish	Duration
1	Requirements Gathering	12/12/2019	12/21/2019	10d
2	Analysis	12/22/2019	1/5/2020	15d
3	Design Implementation	1/6/2020	1/26/2020	21d
4	Image to RGB conversion	1/27/2020	2/6/2020	11d
5	Encrypted Image Generation	2/7/2020	2/18/2020	12d
6	Decryption using RGB	2/19/2020	2/28/2020	10d
7	Original Image Verification	2/29/2020	3/10/2020	11d
8	Testing Encryption function	3/11/2020	3/14/2020	4d
9	Testing Decryption function	3/15/2020	3/18/2020	4d
10	Maintainence	3/19/2020	3/24/2020	6d
11	Documentation	3/25/2020	3/27/2020	3d

Pert Chart


12/12/2019
10 Days
21/12/2019
Requirements Gathering
16/12/2019
4 Days
25/12/2019
22/12/2019
15 Days
05/01/2020
Analysis
26/12/2019
4 Days
09/01/2020
27/01/2020
11 Days
06/02/2020
Image to RGB Conversion
31/01/2020
4 Days
10/02/2020
06/01/2020
21 Days
26/01/2020
Design Implementation
10/01/2020
4 Days
30/01/2020
07/02/2020
12 Days
18/02/2020
Encrypted Image Generation
11/02/2020
4 Days
22/02/2020
19/02/2020
10 Days
28/02/2020
Decryption using RGB
23/02/2020
4 Days
03/03/2020
11/03/2020
4 Days
14/03/2020
Testing Encryption Function
15/03/2020
4 Days
18/03/2020
29/02/2020
11 Days
10/03/2020
Original Image Verification
04/03/2020
4 Days
14/03/2020
15/03/2020
4 Days
18/03/2020
Testing Decryption Function
19/03/2020
4 Days
22/03/2020
19/03/2020
6 Days
24/03/2020
Maintainence
23/03/2020
4 Days
28/03/2020
25/03/2020
3 Days
27/03/2020
Documentation
29/03/2020
4 Days
31/03/2020

Timeline Chart



Work Breakdown Structure

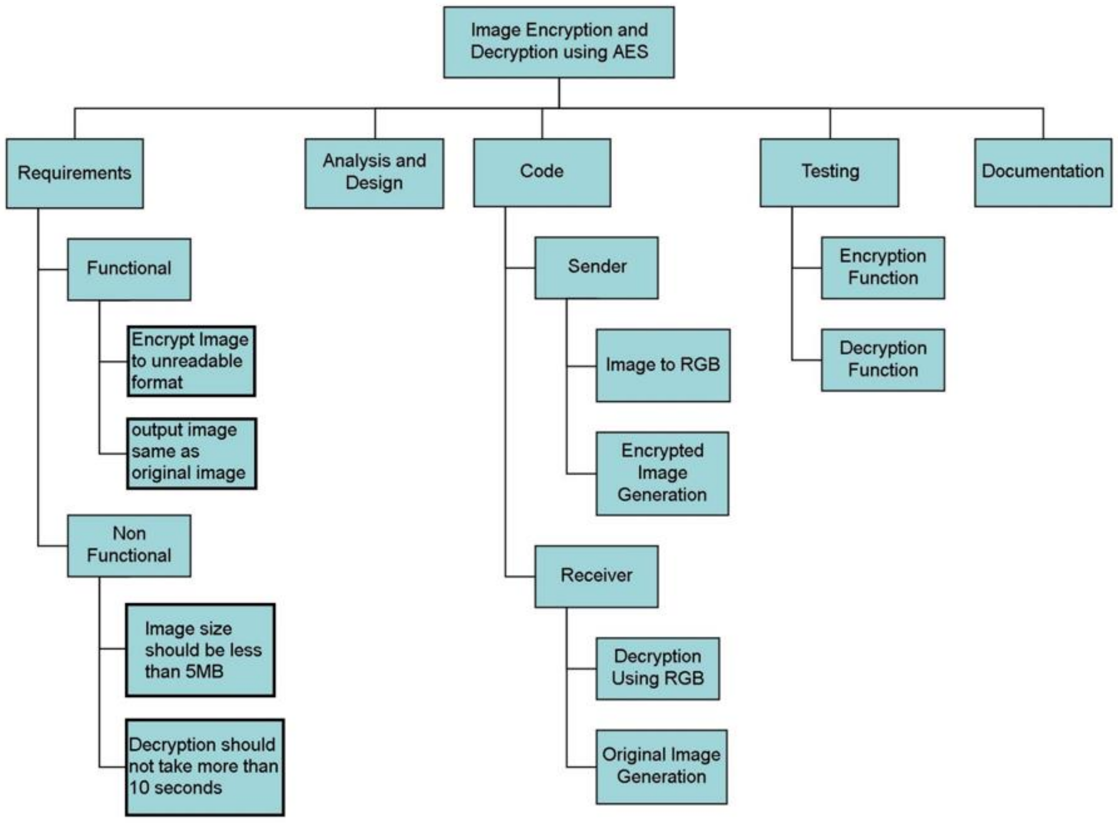
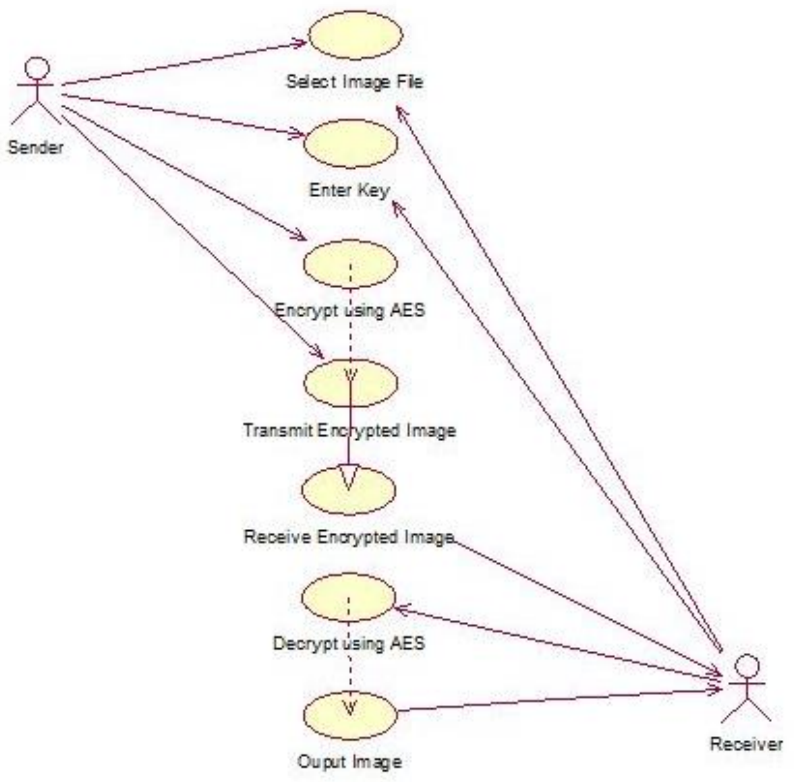


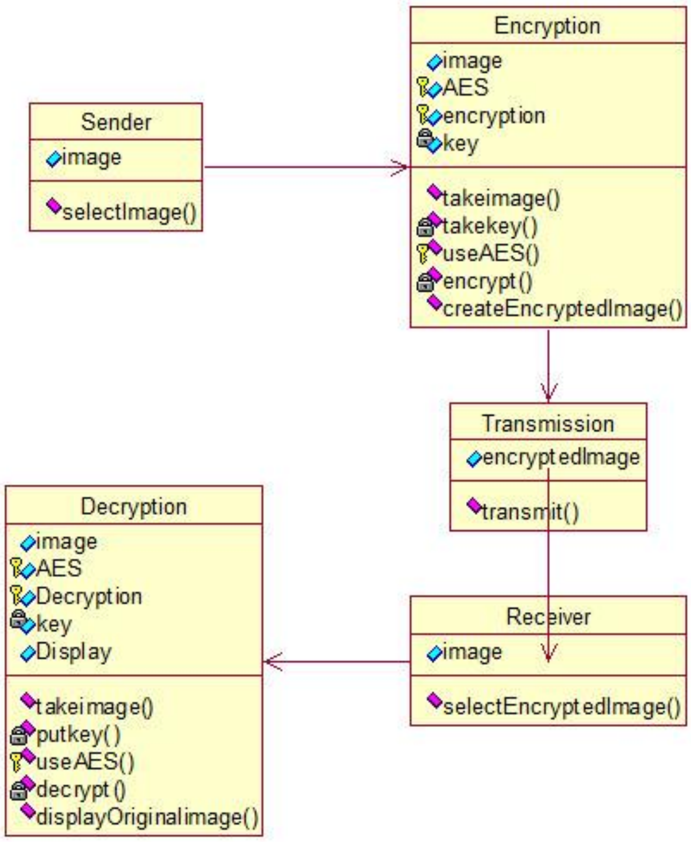
Image Encryption and Decryption using AES
Requirements
Analysis and Design
Code
Testing
Documentation
Functional
Encrypt Image to unreadable format
output image same as original image
Non Functional
Image size should be less than 5MB
Decryption should not take more than 10 seconds
Sender
Image to RGB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

Design Diagrams

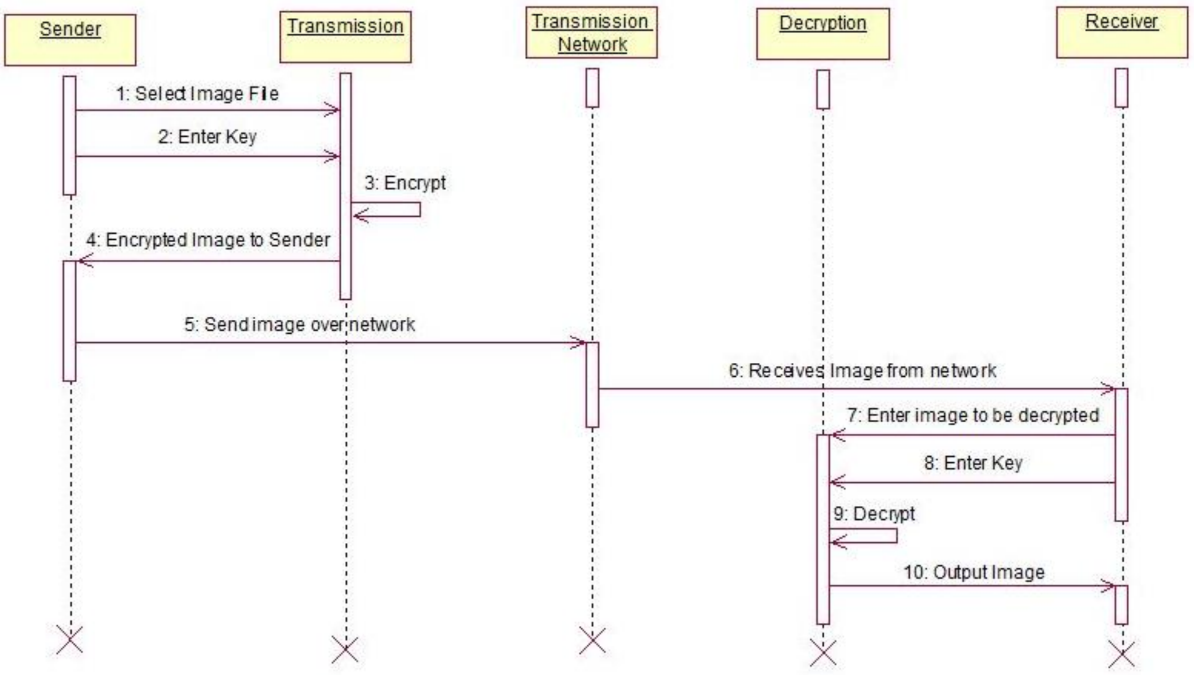
Usecase Diagram:



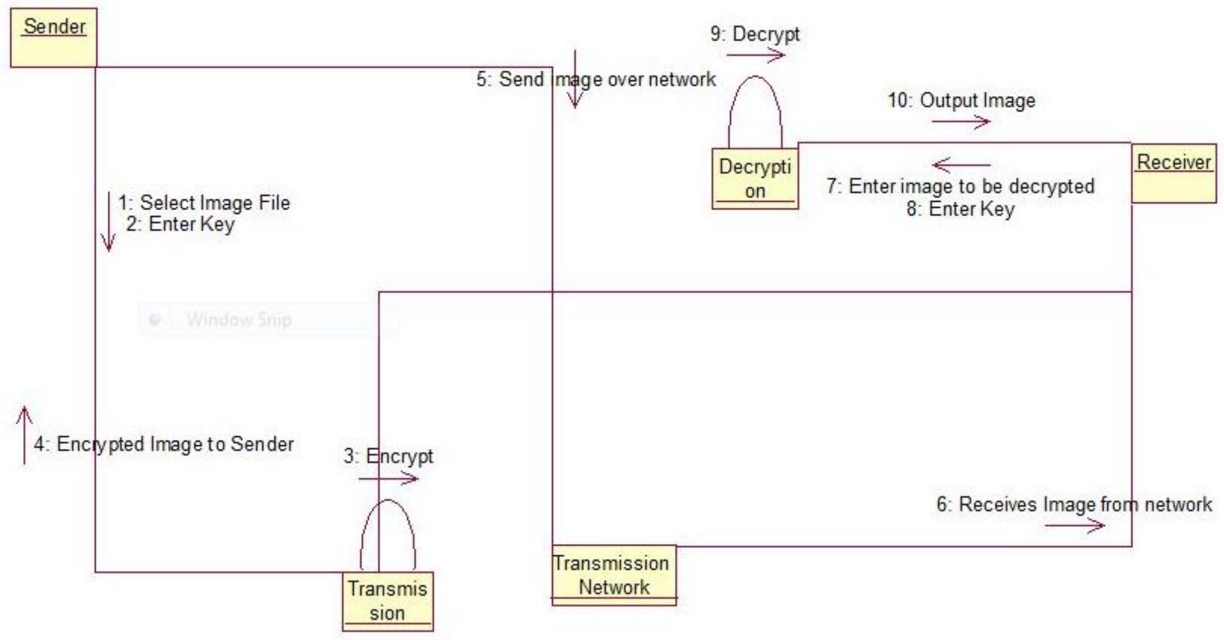
Class Diagram:



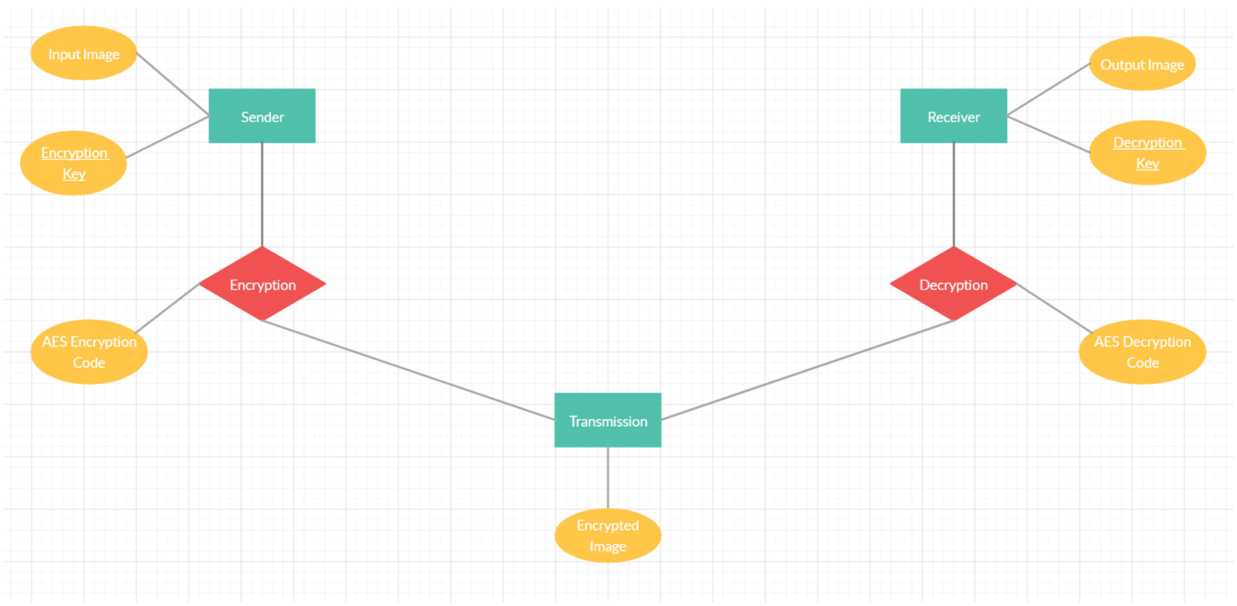
Sequence Diagram:



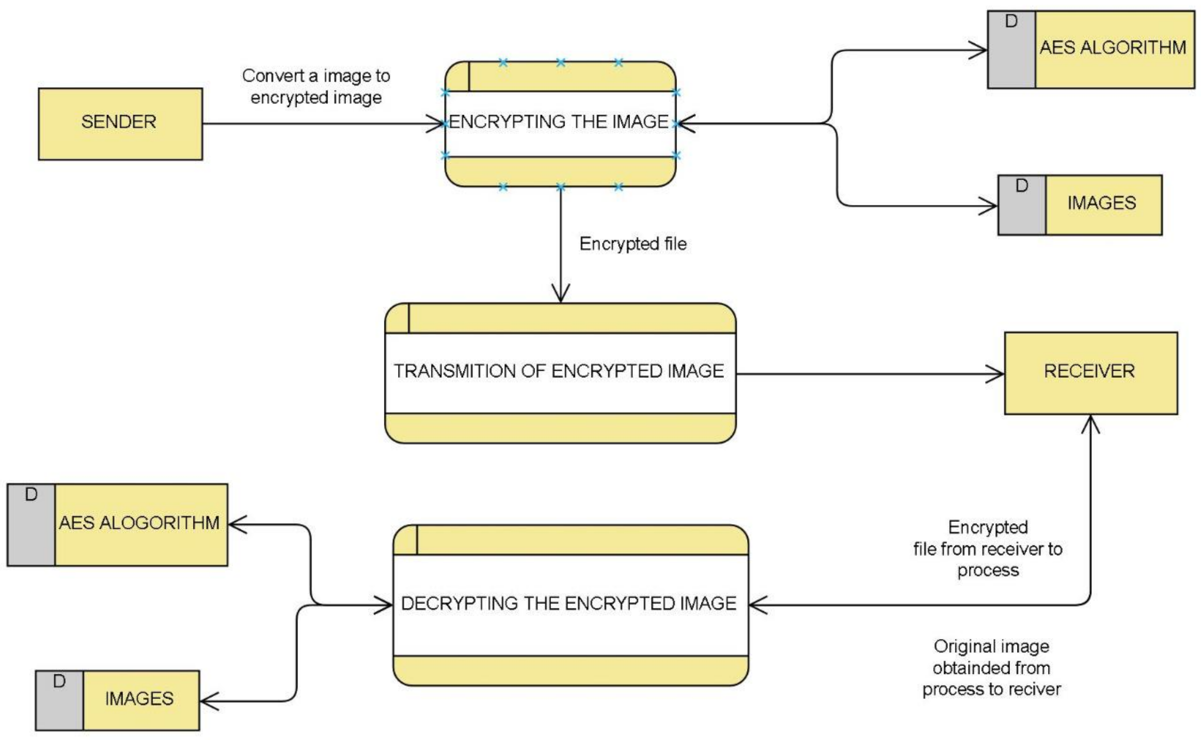
Collaboration Diagram:



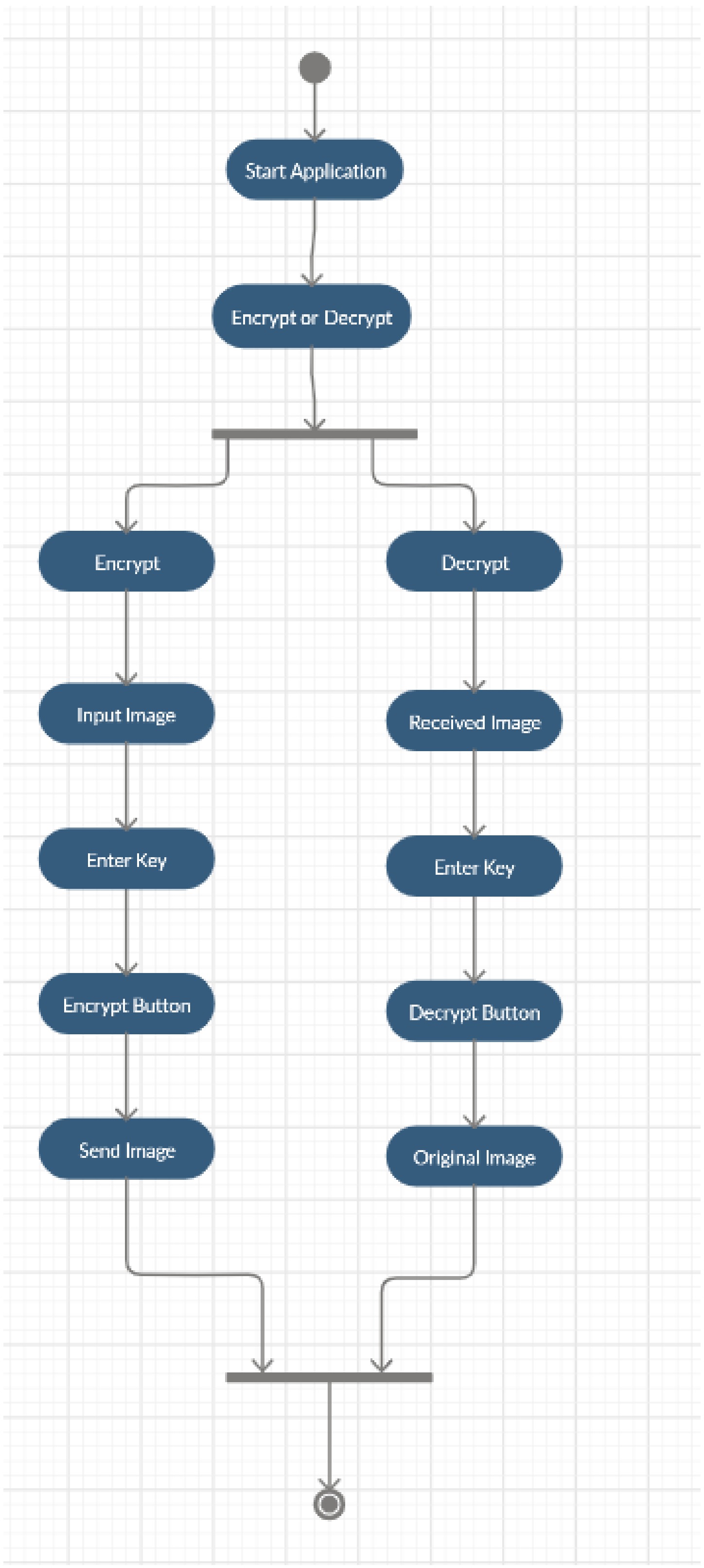
ER Diagram:



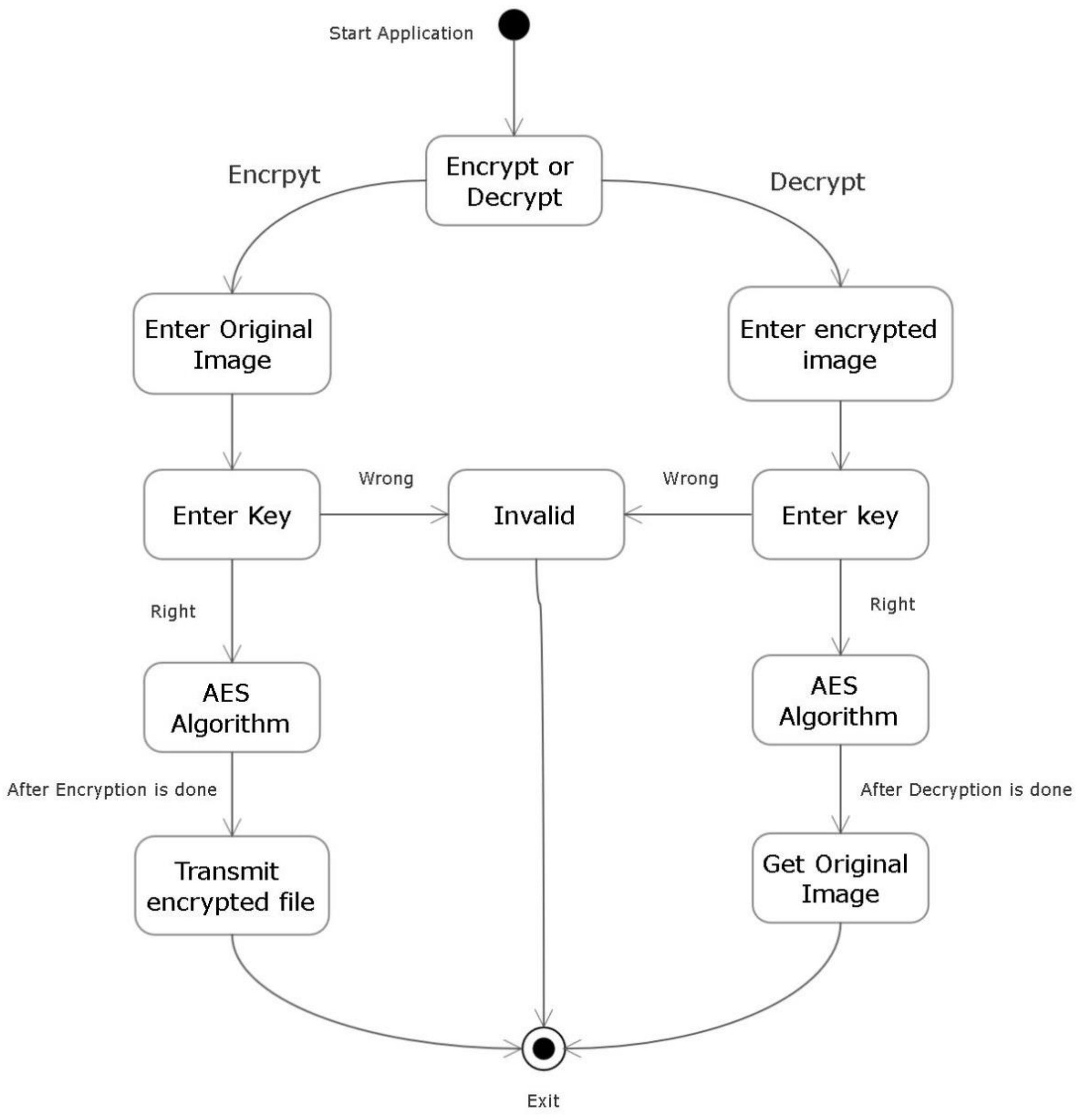
Dataflow Diagram:


SENDER
Convert a image to encrypted image
ENCRYPTING THE IMAGE
D
AES ALGORITHM
D
IMAGES
Encrypted file
TRANSMITION OF ENCRYPTED IMAGE
RECEIVER
D
AES ALOGORITHM
D
IMAGES
DECRYPTING THE ENCRYPTED IMAGE
Encrypted file from receiver to process
Original image obtainded from process to reciver

Activity Diagram:

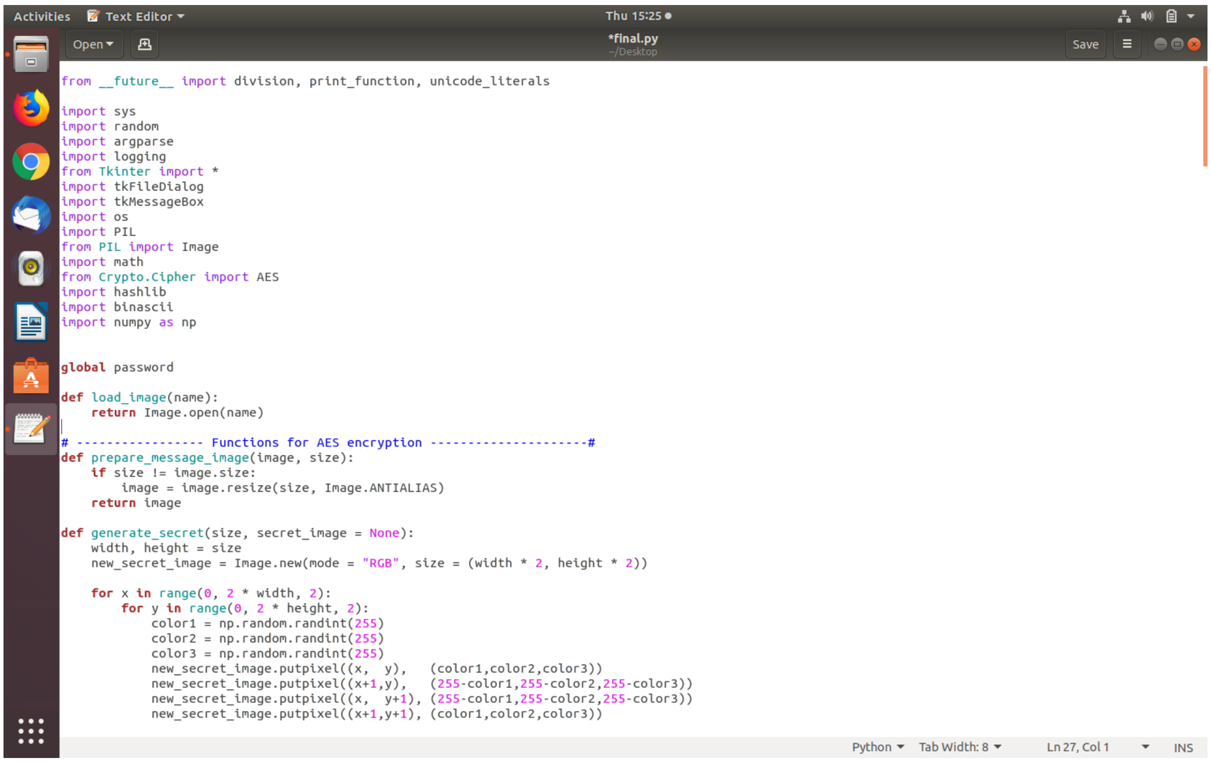


State Chart Diagram:

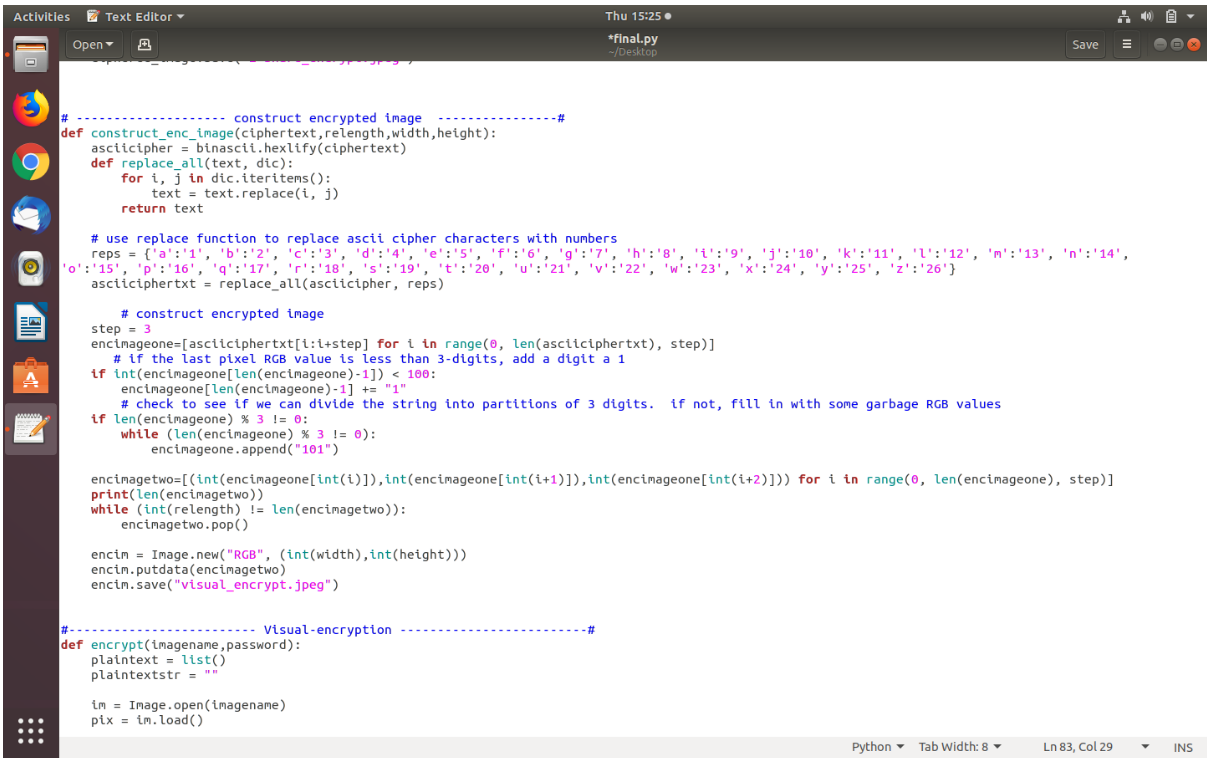


Screenshot of Implementation

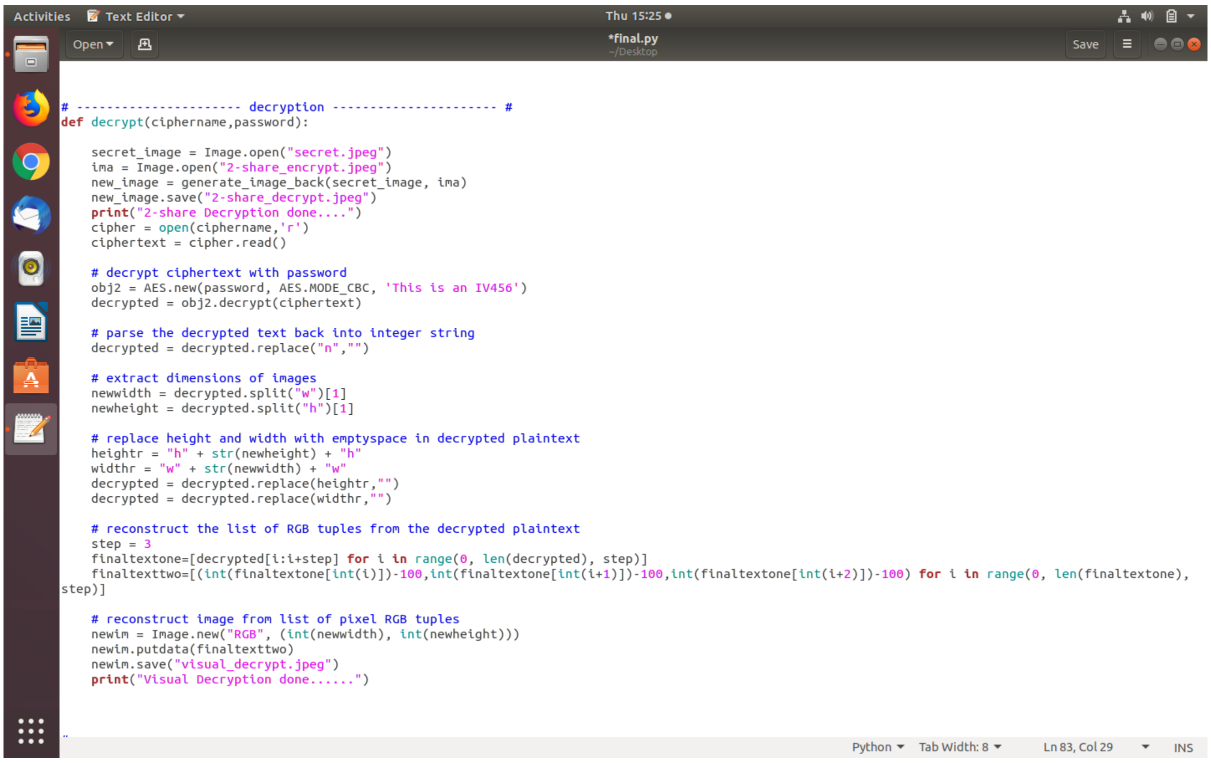
Code



```python
from __future__ import division, print_function, unicode_literals

import sys
import random
import argparse
import logging
from Tkinter import *
import tkFileDialog
import tkMessageBox
import os
import PIL
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 AES 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):
        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))
```

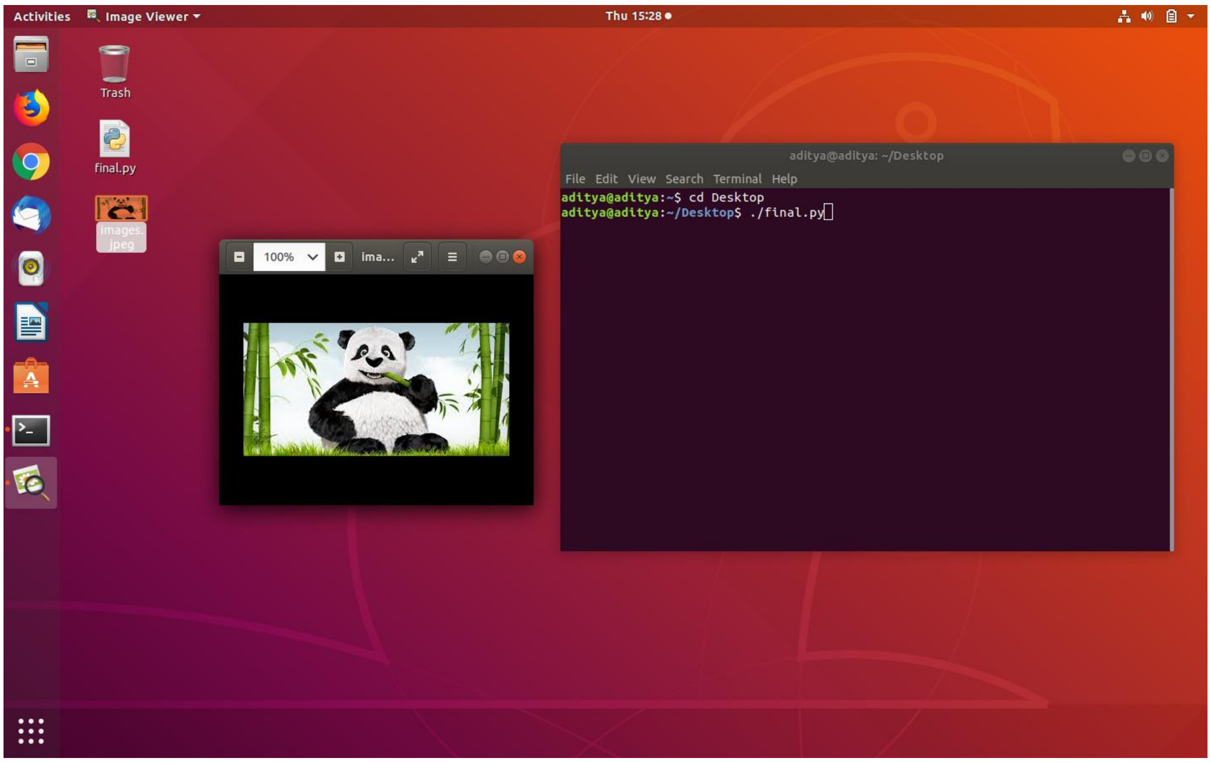


```python
# ------------------- construct encrypted image  ---------------#
def construct_enc_image(ciphertext,relength,width,height):
    asciicipher = binascii.hexlify(ciphertext)
    def replace_all(text, dic):
        for i, j in dic.iteritems():
            text = text.replace(i, j)
        return 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, 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()
```



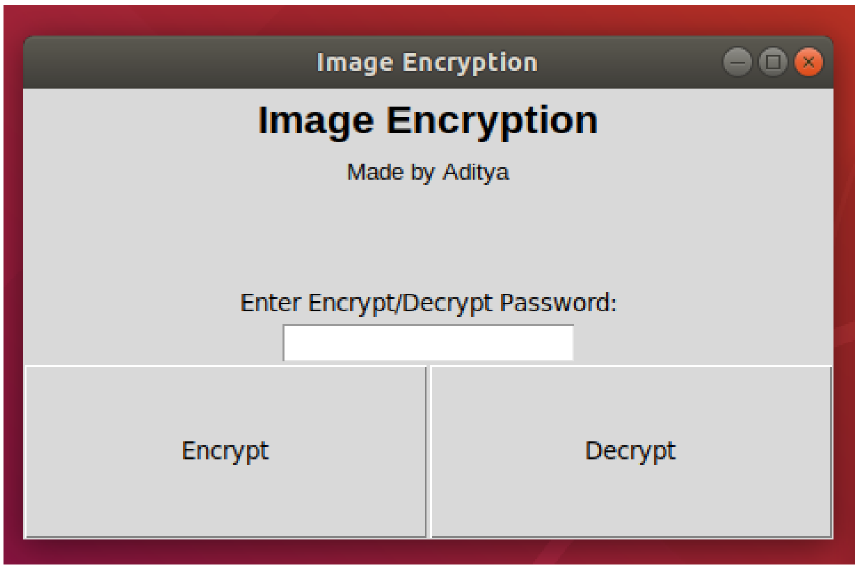
```
# --------------------- decryption --------------------- #
def decrypt(ciphername,password):

    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,'r')
    ciphertext = cipher.read()

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

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

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

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

    # 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......")
```

Original Image and terminal window:

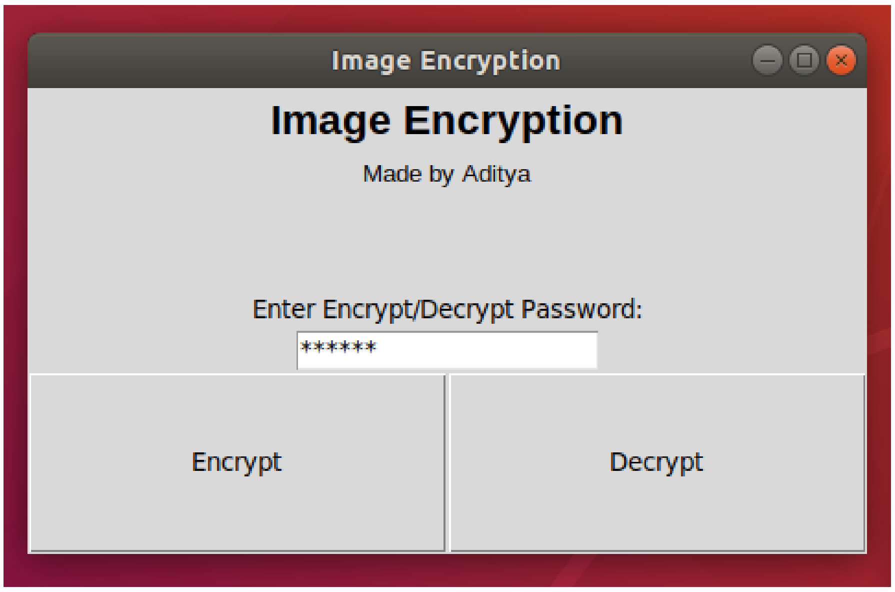


```
aditya@aditya:~$ cd Desktop
aditya@aditya:~/Desktop$ ./final.py
```

Application's first screen asking for key:



Key is entered:



After clicking Encrypt button:

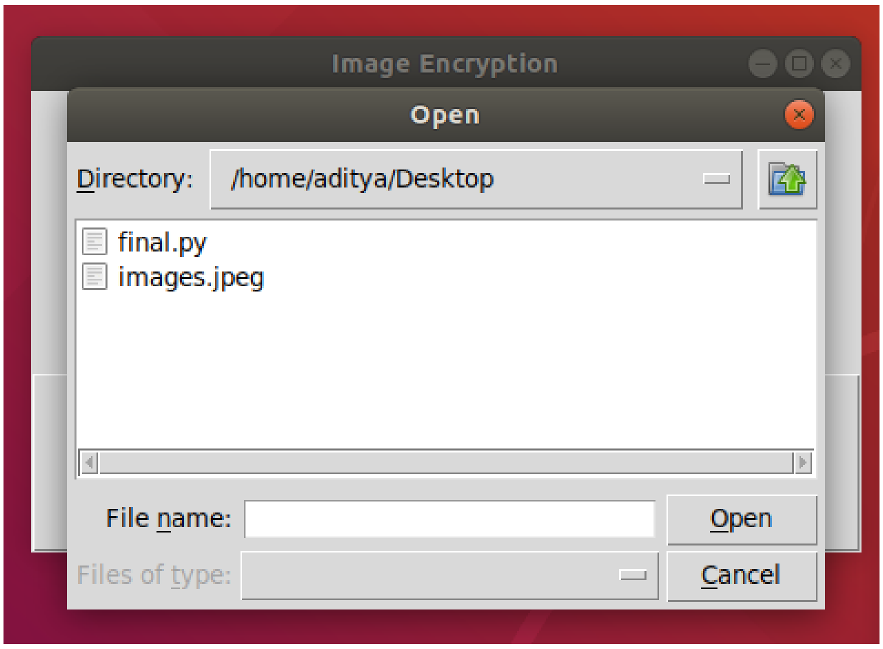


Image Encryption

Open

Directory: /home/aditya/Desktop

final.py
images.jpeg

File name:

Files of type:

Select the directory and then the original file to be encrypted:

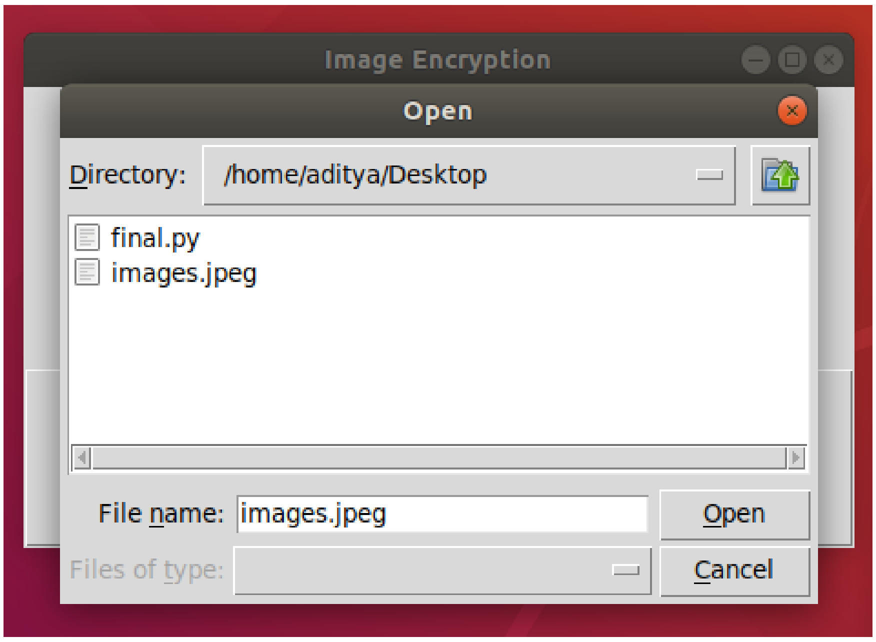


Image Encryption

Open

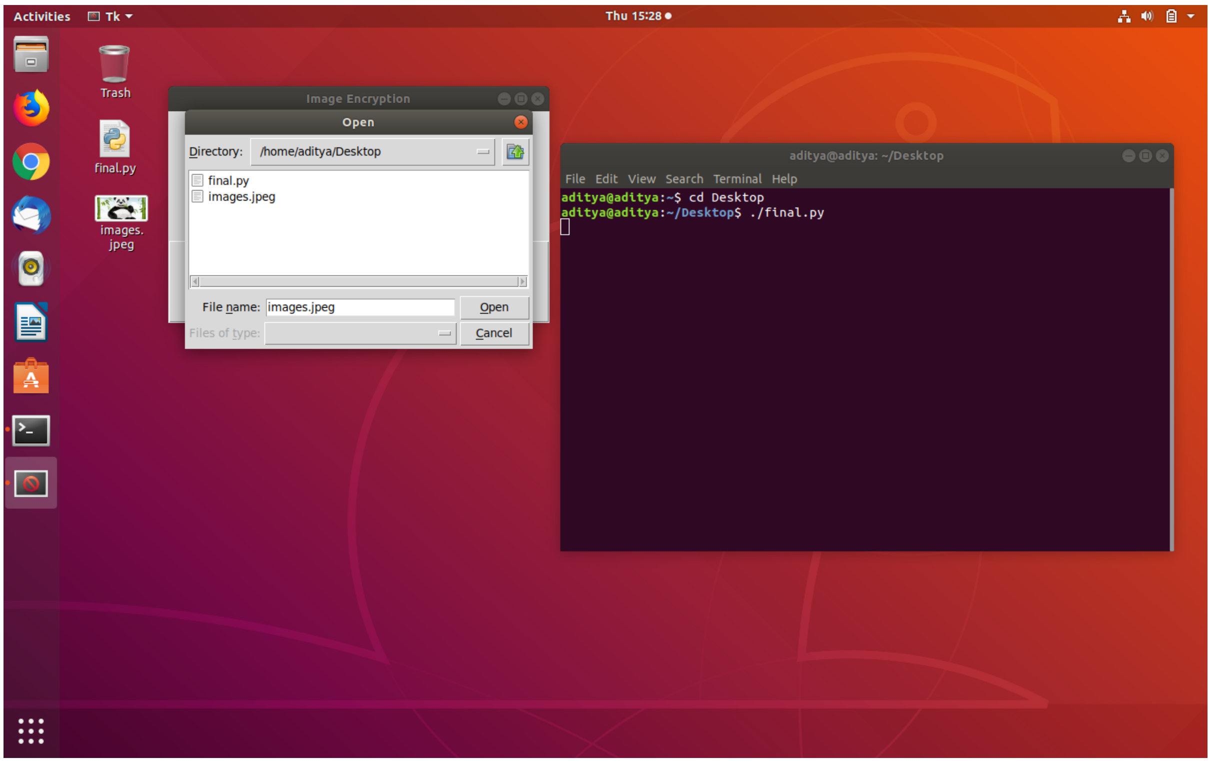
Directory: /home/aditya/Desktop

final.py
images.jpeg

File name: images.jpeg

Files of type:

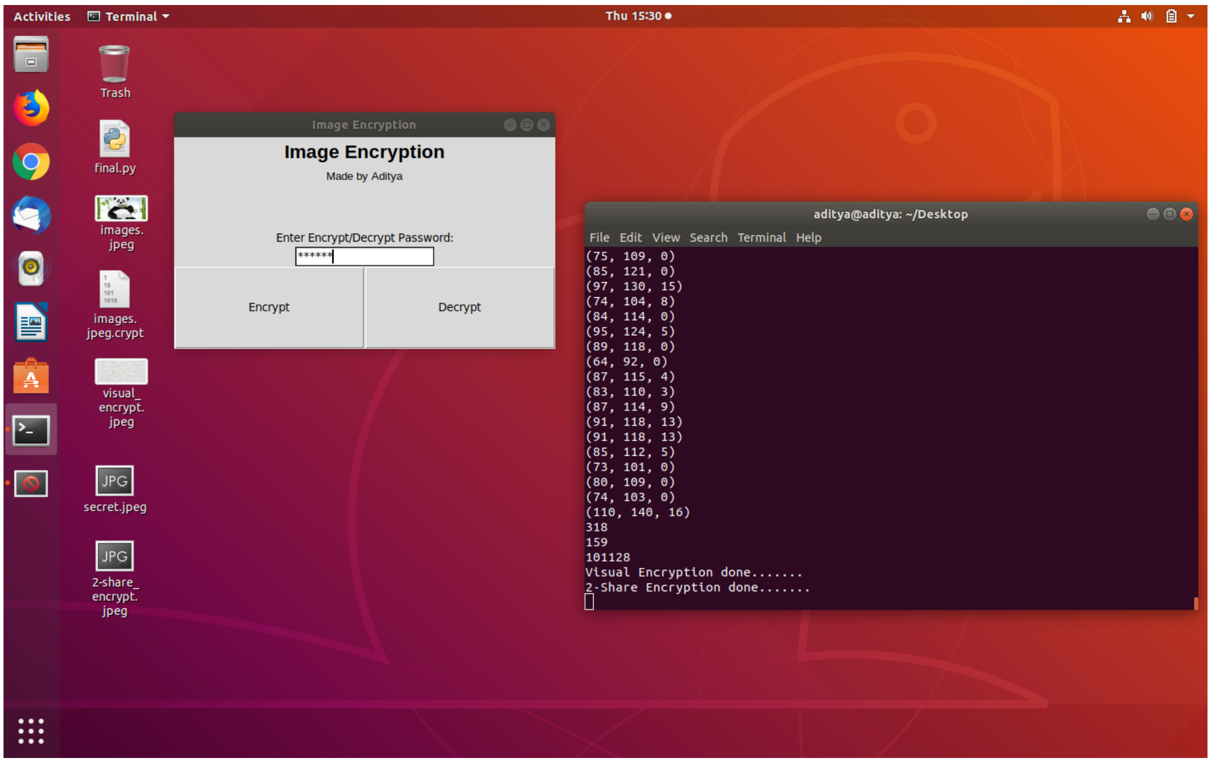
Terminal and desktop before clicking ‘open’:



```
aditya@aditya:~$ cd Desktop
aditya@aditya:~/Desktop$ ./final.py
```

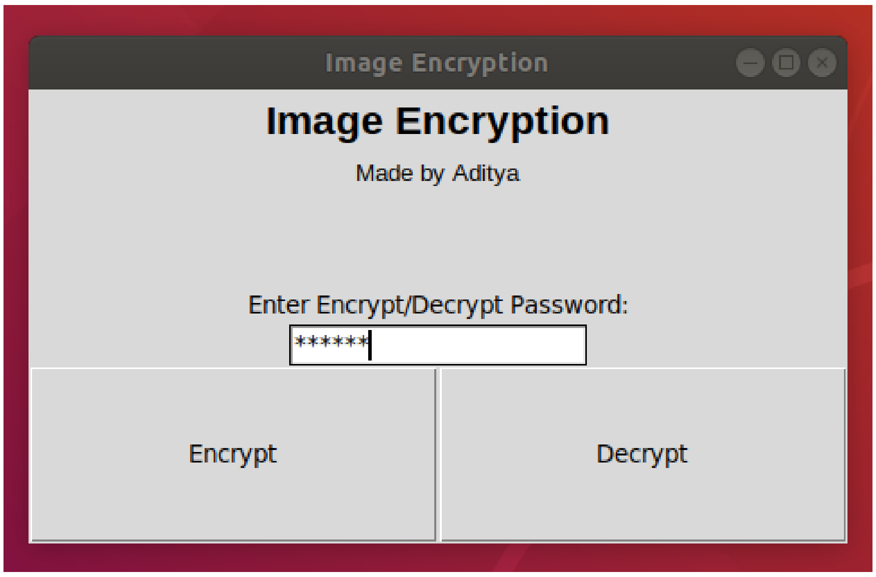
Terminal and desktop screen after clicking ‘open’:

This shows that Encryption is completed and a file named ‘images.jpeg.crypt’ is created. This is our Encrypted file.



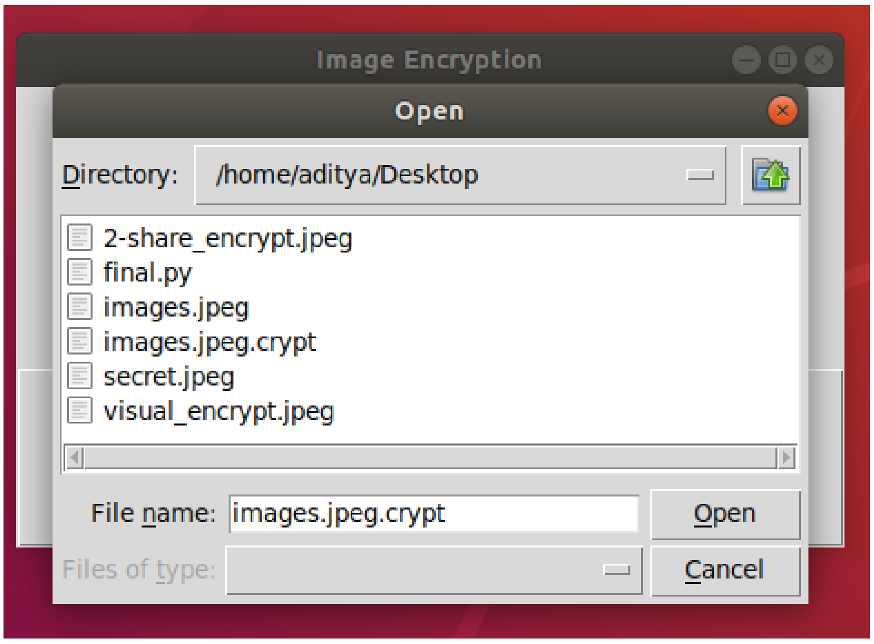
```
(75, 109, 0)
(85, 121, 0)
(97, 130, 15)
(74, 104, 8)
(84, 114, 0)
(95, 124, 5)
(89, 118, 0)
(64, 92, 0)
(87, 115, 4)
(83, 110, 3)
(87, 114, 9)
(91, 118, 13)
(91, 118, 13)
(85, 112, 5)
(73, 101, 0)
(80, 109, 0)
(74, 103, 0)
(110, 140, 16)
318
159
101128
Visual Encryption done.......
2-Share Encryption done.......
```

For decryption, enter same key:



After clicking decryption button:

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



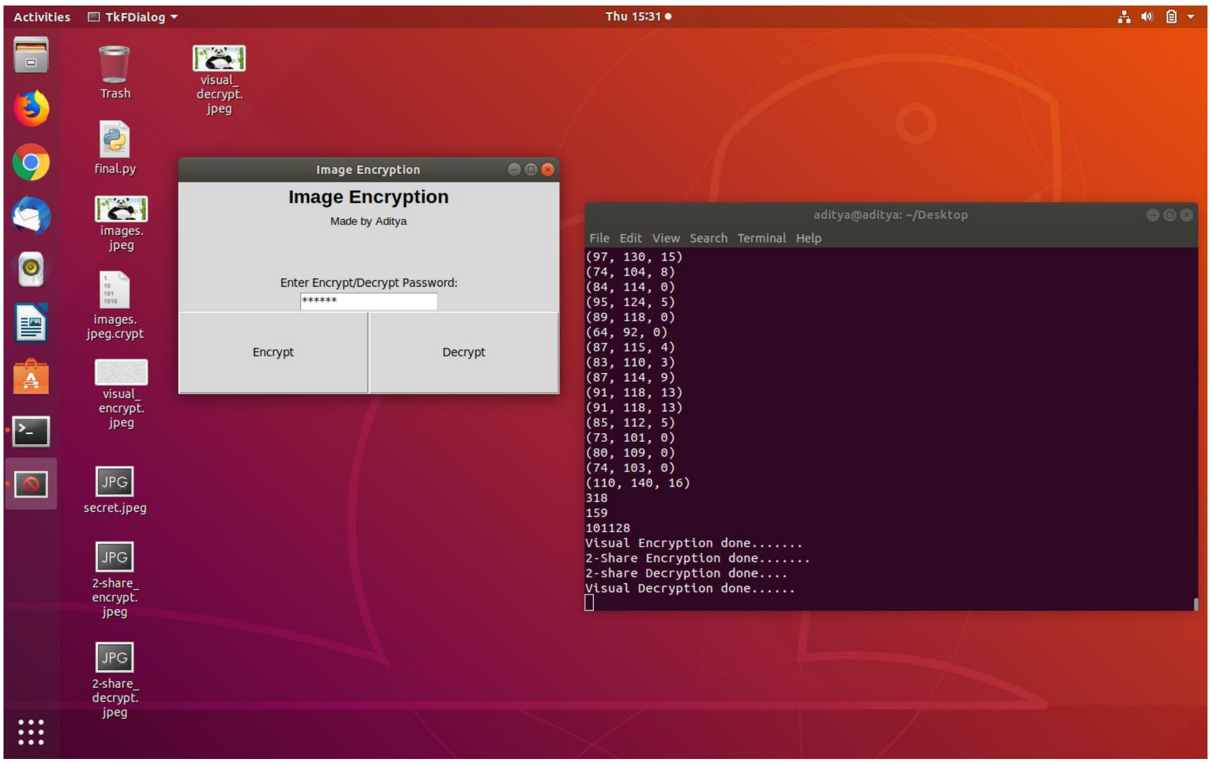
Directory: /home/aditya/Desktop

```
2-share_encrypt.jpeg
final.py
images.jpeg
images.jpeg.crypt
secret.jpeg
visual_encrypt.jpeg
```

File name: images.jpeg.crypt

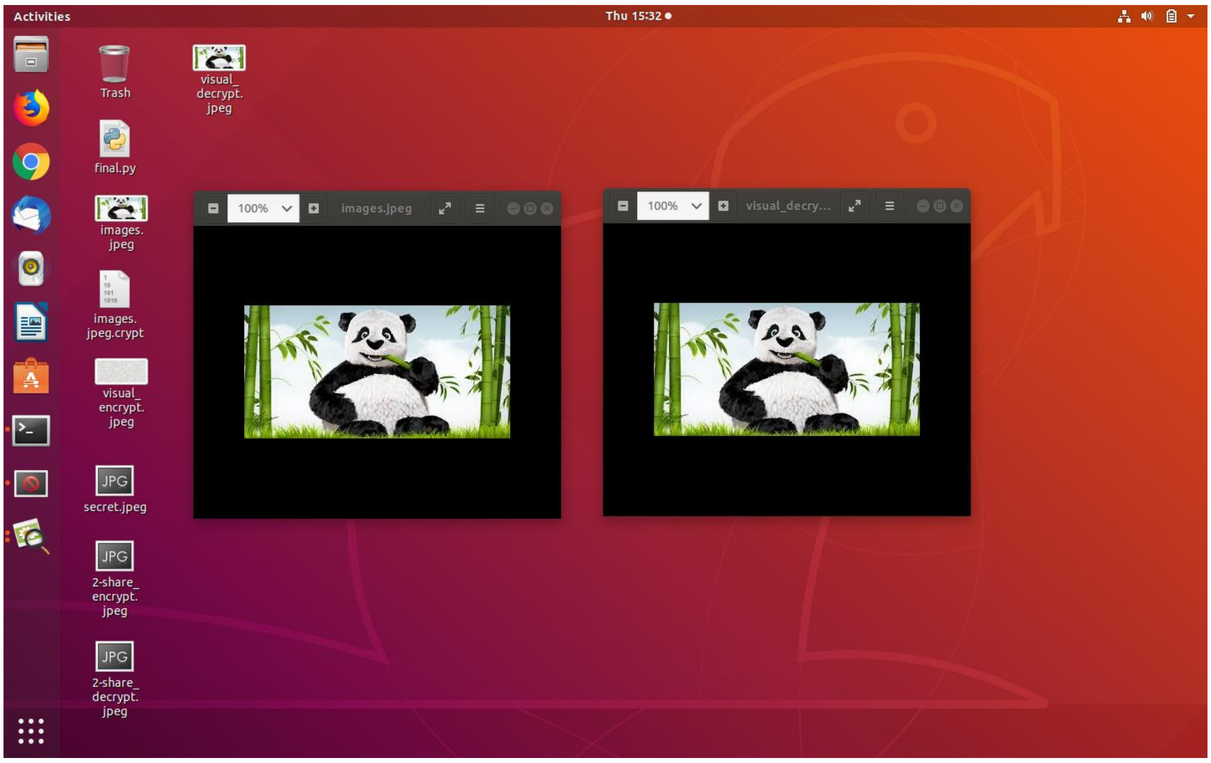
Terminal and desktop screen after clicking ‘open’:

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



```
(97, 130, 15)
(74, 104, 8)
(84, 114, 0)
(95, 124, 5)
(89, 118, 0)
(64, 92, 0)
(87, 115, 4)
(83, 110, 3)
(87, 114, 9)
(91, 118, 13)
(91, 118, 13)
(85, 112, 5)
(73, 101, 0)
(80, 109, 0)
(74, 103, 0)
(110, 140, 16)
318
159
101128
Visual Encryption done.......
2-Share Encryption done.......
2-share Decryption done....
Visual Decryption done......
```

Matching Decrypted Image with the Original Image:



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.

Link to Demo Video

https://youtu.be/4l-pS8uPaJ4