

# Table of Contents

<b>Table of Contents</b> .....	6
<b>1. Introduction</b> .....	7
1.1.Purpose .....	7
1.2.Document Conventions.....	7
1.3.Product Scope .....	7
1.4.References.....	<b>Error! Bookmark not defined.</b>
<b>2. Overall Description</b> .....	8
2.1.Product Perspective.....	8
2.2.Product Functions .....	8
2.3.User Classes and Characteristics .....	8
2.4.Operating Environment.....	8
2.5.Design and Implementation Constraints .....	8
2.6.Assumptions and Dependencies .....	9
<b>3. External Interface Requirements</b> .....	9
3.1.User Interfaces .....	9
3.2.Hardware Interfaces .....	9
3.3.Software Interfaces .....	9
3.4.Communications Interfaces .....	9
<b>4. System Features</b> .....	10
4.1.Encryption.....	10
4.2.Decryption .....	10
<b>5.Other Nonfunctional Requirements</b> .....	11
5.1.Performance Requirements.....	11
5.2.Security Requirements .....	11
5.3.Software Quality Attributes .....	11
<b>6.Snapshots Of Code</b> .....	8
<b>7.Test Cases</b> .....	17
Appendix: Analysis Model .....	13

# Introduction

In computing, encryption is the method by which plaintext or any other type of data is converted from a readable form to an encoded version that can only be decoded by another entity if they have access to a decryption key. Encryption is one of the most important methods for providing data security, especially for end-to-end protection of data transmitted across networks.

## 1.1. Purpose

The purpose of this software is provide the security of Data for which no one except the customer can see the private information.

## 1.2. Document Conventions

In this SRS we will be following certain conventions as given below:

**Key(Paraphrase):** It is a password given by the user to help encryption and decryption.

**Plain text:** Initial message by the sender.

**Cipher text:** The encrypted text.

## 1.3. Product Scope

**Aim:** To create an encryption system in order to establish and maintain privacy.

**Objective:** To design and develop a software that enables encryption process and decryption process to maintain security.

**Applicability:** This can be used in all places where client server transactions take place.

## **2.Overall Description**

### **2.1. Product Perspective**

This project will be a framework that provides reusability of agent's communication. It is implemented in python. Its GUI based software, technical and non-technical person can also use this software very easily. It requires 16 MB RAM to execute this software and this software is supportable to any of version OS windows.

### **2.2. Product Functions**

The product will provide following functionality: -

It will provide a GUI interface in which the user will select the image and on the click of button the input image will translate into an unintelligent form and transmitted which can be decrypted later.

### **2.3. User Classes and Characteristics**

This is a cryptographic application that is usable by all classes of users. This is usable by both for novice or technical users as well as expert technical users.

### **2.4. Operating Environment**

The software is built using the Idle 2.7.15 with a full knowledge of python. The system must have 16 MB RAM and around 50 MB disk space.

### **2.5. Design and Implementation Constraints**

There are some areas that can create an issue for running the software.

In this case, corresponding libraries have been configured in this laptop. If another pc isn't configured according to the software it may cause a problem.

For hardware there are no specific issues but processor may be the reason to affect the computation of the software.

## **2.6. Assumptions and Dependencies**

1. We assume that there is an agent whom each agent can request for the different encryption and accordingly generated decryption key. This agent should maintain a list of agents who are allowed to get the keys.
2. We assume that each agent has enough knowledge to decide the best way to communicate with the other agents.

## **3.External Interface Requirements**

### **3.1. User Interfaces**

1. In this software, have two main buttons (Encrypt, decrypt) and a textbox for input paraphrase.
2. First tab i.e. encrypt tab will allow a user to select the image they wish to encrypt and provides the encrypted image.
3. Second tab i.e. decrypts option, will allow the user to decrypt the encrypted image.

### **3.2. Hardware Interfaces**

Image security system is made in python so it is Hardware Independent software. There is no need of any hardware. It runs on any operating system and it's not a heavy software so no extra RAM is needed. It's required 16 MB RAM to executed software and run on any OS.

### **3.3. Software Interfaces**

Implementation will be in python programming language that will execute in any system in which has a python compiler that is depend on operating system to system.

### **3.4. Communications Interfaces**

None

## **4.System Features**

### **4.1. Encryption**

**4.1.1 Description:** This system feature involves encrypting the image using Advanced Encryption System Algorithm which is symmetric.

#### **4.1.2 Function Requirements:**

1. Firstly, the user types the paraphrase in the given input field.
2. Select the file using Encrypt option that he wants to encrypt.
3. Before pressing the Encrypt button, you must enter the key that helps to encrypt your file.
4. After selecting your file successfully, it gives the encrypted file.
5. After encryption, file is saved in destination folder.

### **4.2. Decryption**

**4.2.1 Description:** This system feature involves decrypting the image using the same paraphrase and AES algorithm.

#### **4.2.2Functional Requirement:**

1. Select file using Decrypt option that you want to decrypt.
2. Enter the paraphrase value in key field; without entering the key value you cannot decrypt the image.
3. Press the decrypt button to get the decrypted file.

## **5.Other Non-functional Requirements**

### **5.1. Performance Requirements**

Most cryptographic ciphers rely on high computational cost operations. Therefore, keeping performance considerations in mind, for data encryption/decryption computational effort to encryption/decryption using Asymmetric key is very powerful compared to symmetric key algorithm. It provides more security compared to symmetric key and also performance of encrypting file is very good. It is general purpose software. But, for simplicity reasons we have chosen symmetric algorithm.

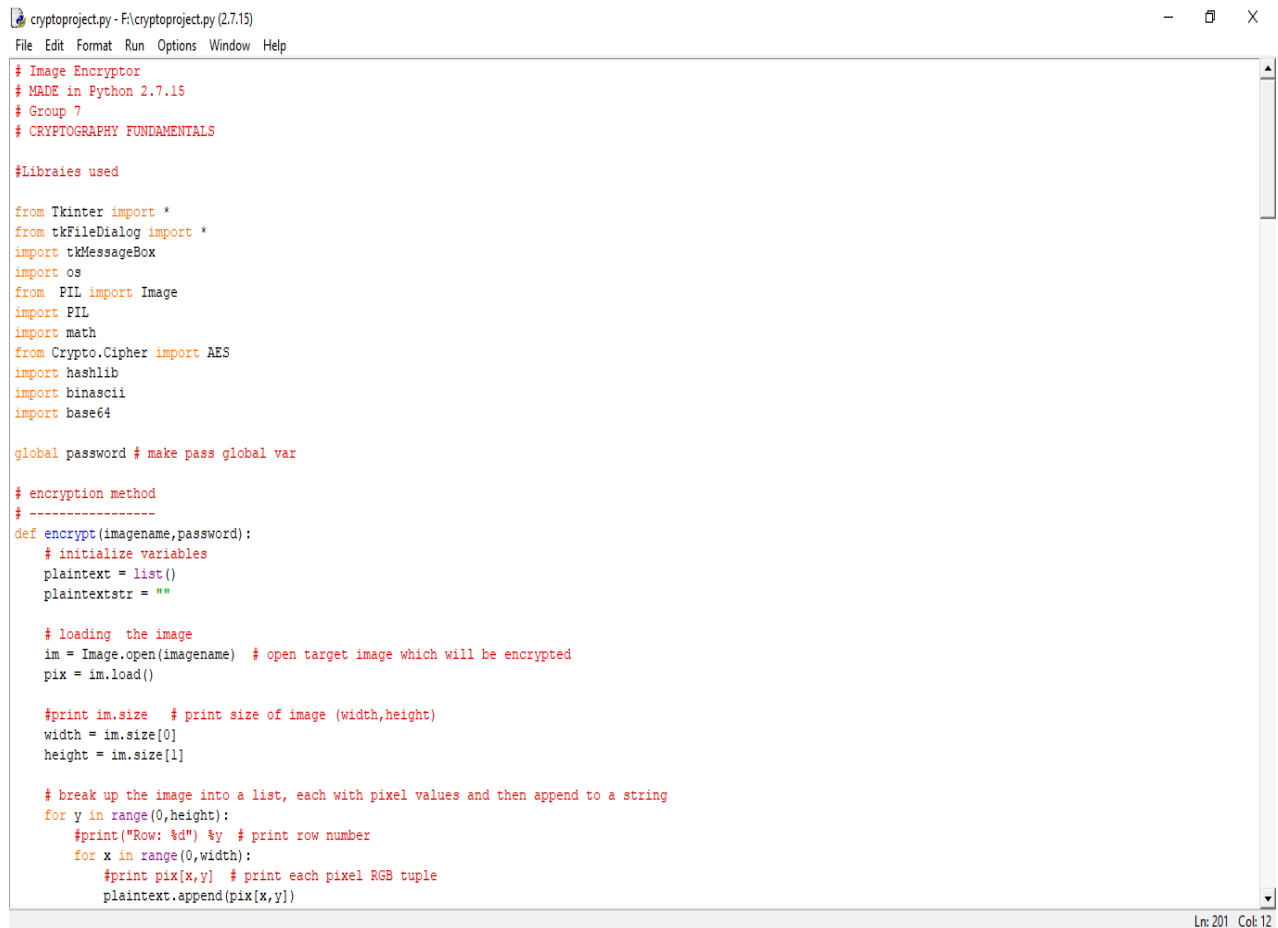
### **5.2. Security Requirements**

User is required to remember his password that he/she used to encrypt data (or lock password safe) because most of secure cryptographic algorithms implemented in this suite are secure enough so that no algorithms better than brute-force can be used to recover lost password.

### **5.3. Software Quality Attributes**

The source code should be properly documented, so that new developers will be able to understand the code as easily as possible. It should be easily available for every end user for better security.

## 6.Code Screenshots (Procedural):



The screenshot shows a Python IDE window titled 'cryptoproject.py - F:\cryptoproject.py (2.7.15)'. The menu bar includes 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code is written in a syntax-highlighted format with red for comments and blue for function definitions. The code defines an 'encrypt' function that takes an image name and a password as input. It initializes variables for plaintext and then proceeds to load the image, print its dimensions, and iterate over each pixel to build a plaintext string. The status bar at the bottom right indicates 'Ln: 201 Col: 12'.

```
# Image Encryptor
# MADE in Python 2.7.15
# Group 7
# CRYPTOGRAPHY FUNDAMENTALS

#Libraies used

from Tkinter import *
from tkFileDialog import *
import tkMessageBox
import os
from PIL import Image
import PIL
import math
from Crypto.Cipher import AES
import hashlib
import binascii
import base64

global password # make pass global var

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

    # loading the image
    im = Image.open(imagename) # open target image which will be encrypted
    pix = im.load()

    #print im.size # print size of image (width,height)
    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):
        #print("Row: %d" %y # print row number
        for x in range(0,width):
            #print pix[x,y] # print each pixel RGB tuple
            plaintext.append(pix[x,y])
```

Ln: 201 Col: 12

```

# add 100 to each tuple value to make sure each are 3 digits long. being able to do this is really just a PoC
# that you'll be able to use a raw application of RSA to encrypt, rather than PyCrypto if you wanted.
for i in range(0, len(plaintext)):
    for j in range(0, 3):
        plaintextstr = plaintextstr + "%d" % (int(plaintext[i][j]) + 100)

# 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 plaintextstr length is a multiple of 16 for AES. if not, append "n". not safe in theory
# and i should probably replace this with an initialization vector IV = 16 * '\x00' at some point. In practice
# this IV buffer should be random.
while (len(plaintextstr) % 16 != 0):
    plaintextstr = plaintextstr + "n"

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

# write ciphertext to file for analysis
cipher_name = imagename + ".crypt"
g = open(cipher_name, 'w')
base64_ciphertext = base64.b64encode(ciphertext)
g.write(base64_ciphertext)

# -----
# construct encrypted image (not currently using since Tkinter isn't very nice)
# -----
def construct_enc_image():
    # hexlify the ciphertext
    asciicipher = binascii.hexlify(ciphertext)

    # replace function
    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

```

Ln: 201 Col: 12

```

# 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',
asciiciphertext = replace_all(asciicipher, reps)

# construct encrypted image
step = 3
encimageone=[asciiciphertext[i:i+step] for i in range(0, len(asciiciphertext), 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)]

# make sizes of images equal
while (int(relength) != len(encimagetwo)):
    encimagetwo.pop()

# encrypted image
encim = Image.new("RGB", (int(width),int(height)))
encim.putdata(encimagetwo)

encim.show()
# alert success and path to image
enc_success(cipher_name)

construct_enc_image()

# decryption method
# -----
def decrypt(ciphername, password):
    # reach ciphertext into memory
    cipher = open(ciphername, 'r')
    ciphertext = cipher.read()
    deno=base64.b64decode(ciphertext)

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

```

Ln: 201 Col: 12



```
cryptoproject.py - F:\cryptoproject.py (2.7.15)
File Edit Format Run Options Window Help

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

# 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 empty space in decrypted plaintext
height = "h" + str(newheight) + "h"
width = "w" + str(newwidth) + "w"
decrypted = decrypted.replace(height,"")
decrypted = decrypted.replace(width,"")

# 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.show()

# -----
# GUI CODE
# -----

# empty password box alert
def pass_alert():
    tkMessageBox.showinfo("Password Alert","Please enter a Valid password.")

def enc_success(imagenam):
    tkMessageBox.showinfo("Successful","Encrypted Image: " + imagenam)

# image encrypt button event
def image_open():
    # useless for now, may need later
    global file_path_a
```

Ln: 201 Col: 12

```
cryptoproject.py - F:\cryptoproject.py (2.7.15)
File Edit Format Run Options Window Help

# check to see if password entry is null. if yes, alert
dec_pass = passg.get()
if dec_pass == "":
    pass_alert()
else:
    password = hashlib.sha256(dec_pass).digest()
    filename = askopenfilename()
    file_path_d = os.path.dirname(filename)
    # decrypt the ciphertext
    decrypt(filename,password)

# main gui app starts here
class App:
    def __init__(self, master):
        # make passg global to use in functions
        global passg
        # setup frontend titles etc blah blah
        title = "Image Security System"
        author = "Made By: Group 7"
        msgtitle = Message(master, text =title)
        msgtitle.config(font=("Times", "24", "bold italic"), width=200)
        msgauthor = Message(master, text=author)
        msgauthor.config(font=("helvetica", "15", "bold"), width=200)

        # draw canvas
        canvas_width = 350
        canvas_height = 100
        w = Canvas(master,
            width=canvas_width,
            height=canvas_height)

        # pack the GUI, this is basic, we should use a grid system
        msgtitle.pack()
        msgauthor.pack()
        w.pack()

        # password field here above buttons
        passlabel = Label(master, text="Enter Encryption/Decryption Password:",font=25)
        passlabel.pack()
        passg = Entry(master, show="*", width=45)
        passg.pack()
```

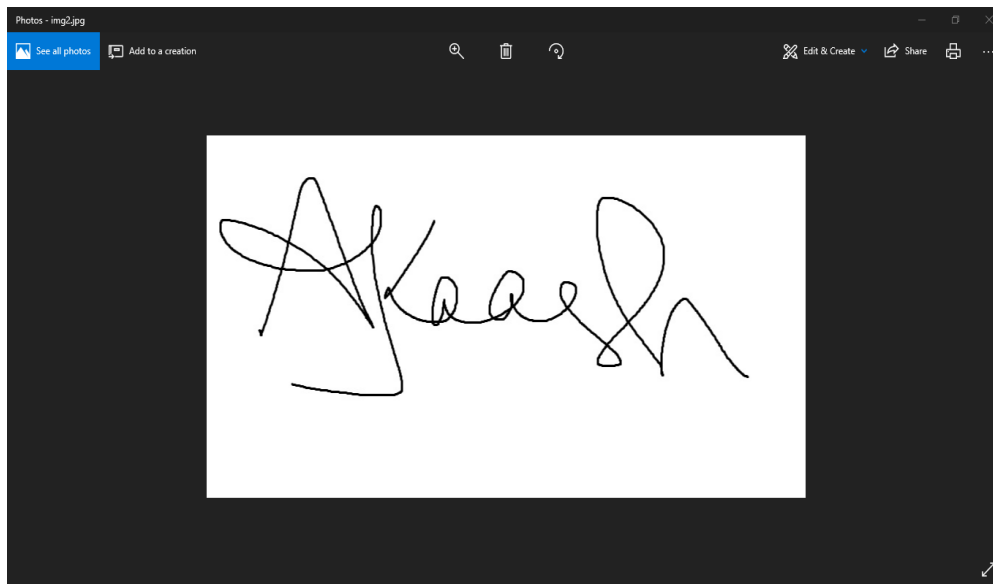
Ln: 201 Col: 12

```
# add both encrypt/decrypt buttons here which trigger file browsers
self.encrypt = Button(master,
                       text="Encrypt", fg="black", font=30,
                       command=image_open, width=20, height=5)
self.encrypt.pack(side=LEFT)
self.decrypt = Button(master,
                       text="Decrypt", fg="black", font=30,
                       command=cipher_open, width=20, height=5)
self.decrypt.pack(side=RIGHT)

root = Tk()
root.wm_title("Image Security System")
app = App(root)
root.mainloop()
```

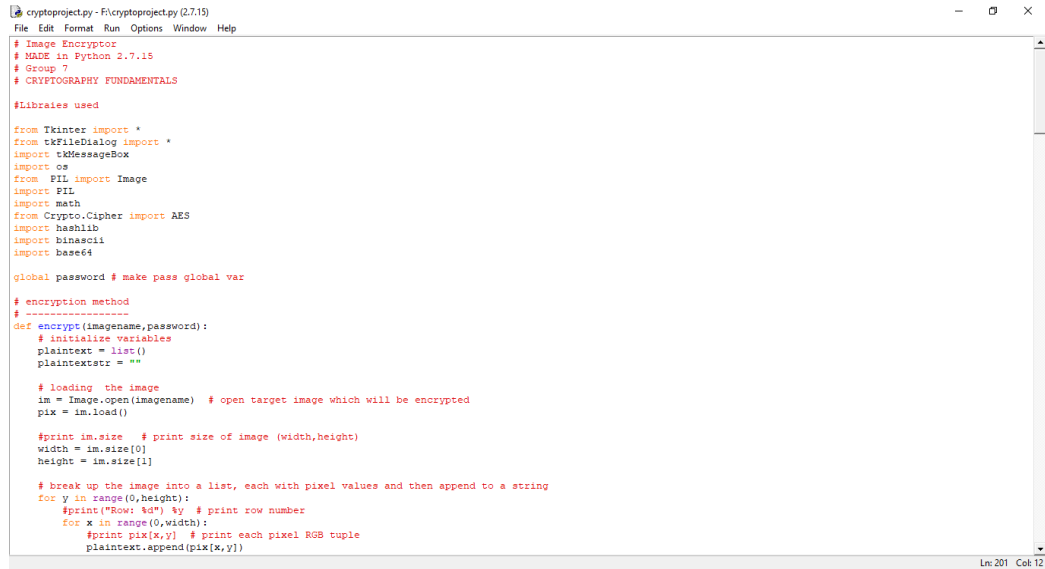
## PROCESS

Step1) Select the file to be encrypted (the file should be in JPEG format)



## ENCRYPTION STARTS HERE

Step2) Open IDLE and open cryptoproject.py



```
cryptoproject.py - F:\cryptoproject.py (2.7.15)
File Edit Format Run Options Window Help

# Image Encryptor
# MADE in Python 2.7.15
# Group 7
# CRYPTOGRAPHY FUNDAMENTALS

#Libraies used

from Tkinter import *
from tkFileDialog import *
import tkMessageBox
import os
from PIL import Image
import PIL
import math
from Crypto.Cipher import AES
import hashlib
import binascii
import base64

global password # make pass global var

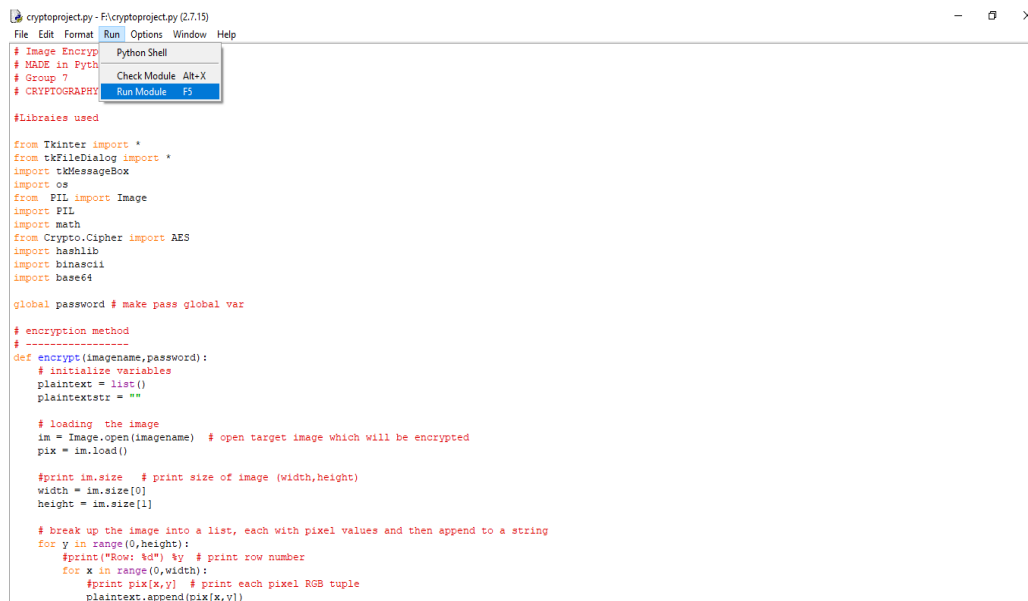
# encryption method
# -----
def encrypt(imagename,password):
    # initialize variables
    plaintext = list()
    plaintextstr = ""

    # loading the image
    im = Image.open(imagename) # open target image which will be encrypted
    pix = im.load()

    #print im.size # print size of image (width,height)
    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):
        #print("Row: %d" % y) # print row number
        for x in range(0,width):
            #print pix[x,y] # print each pixel RGB tuple
            plaintext.append(pix[x,y])
```

Step3) Click on Run and then select “Run Module” option



```
cryptoproject.py - F:\cryptoproject.py (2.7.15)
File Edit Format Run Options Window Help

# Image Encryptor
# MADE in Python 2.7.15
# Group 7
# CRYPTOGRAPHY FUNDAMENTALS

#Libraies used

from Tkinter import *
from tkFileDialog import *
import tkMessageBox
import os
from PIL import Image
import PIL
import math
from Crypto.Cipher import AES
import hashlib
import binascii
import base64

global password # make pass global var

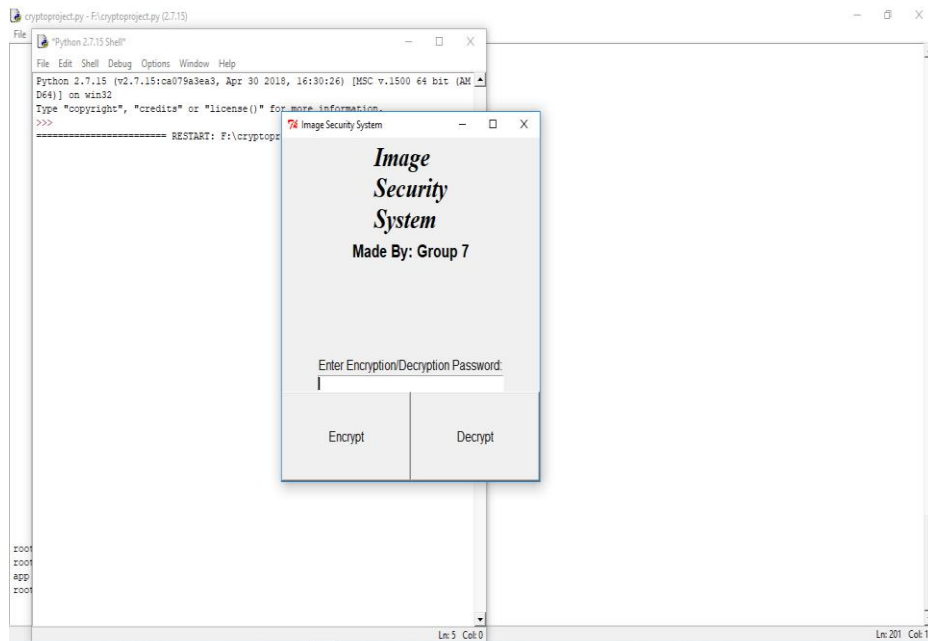
# encryption method
# -----
def encrypt(imagename,password):
    # initialize variables
    plaintext = list()
    plaintextstr = ""

    # loading the image
    im = Image.open(imagename) # open target image which will be encrypted
    pix = im.load()

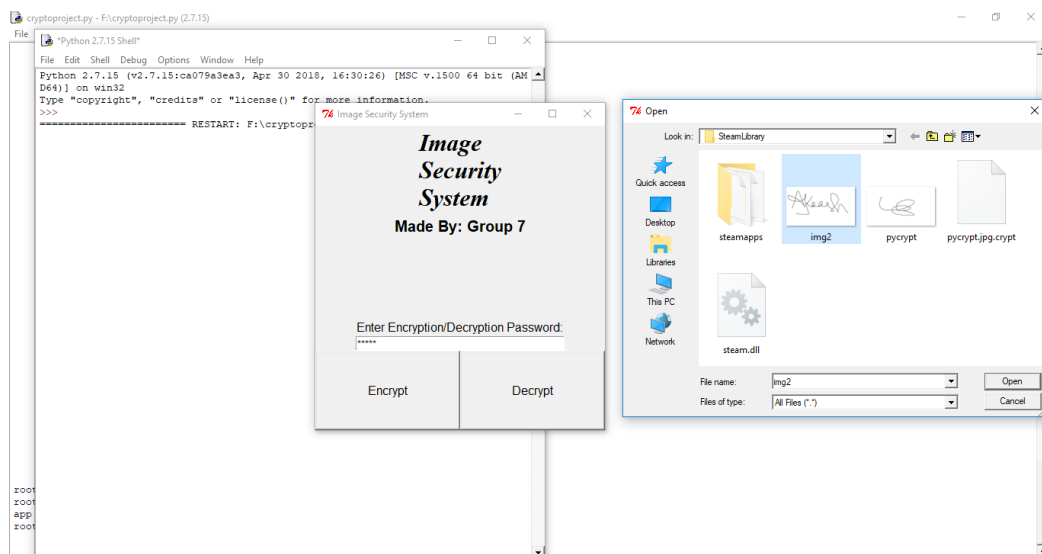
    #print im.size # print size of image (width,height)
    width = im.size[0]
    height = im.size[1]

    # break up the image into a list, each with pixel values and then append to a string
    for y in range(0,height):
        #print("Row: %d" % y) # print row number
        for x in range(0,width):
            #print pix[x,y] # print each pixel RGB tuple
            plaintext.append(pix[x,y])
```

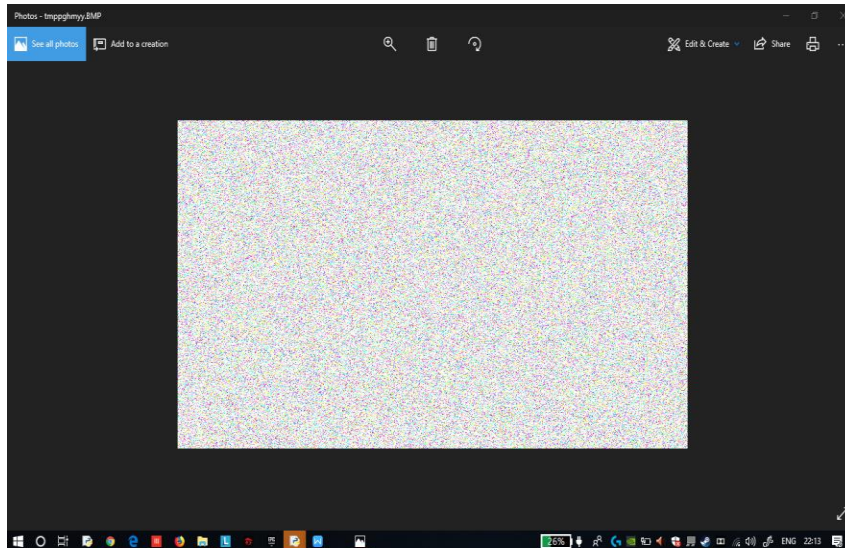
Step4) A dialog box will open which contain the UI of our code



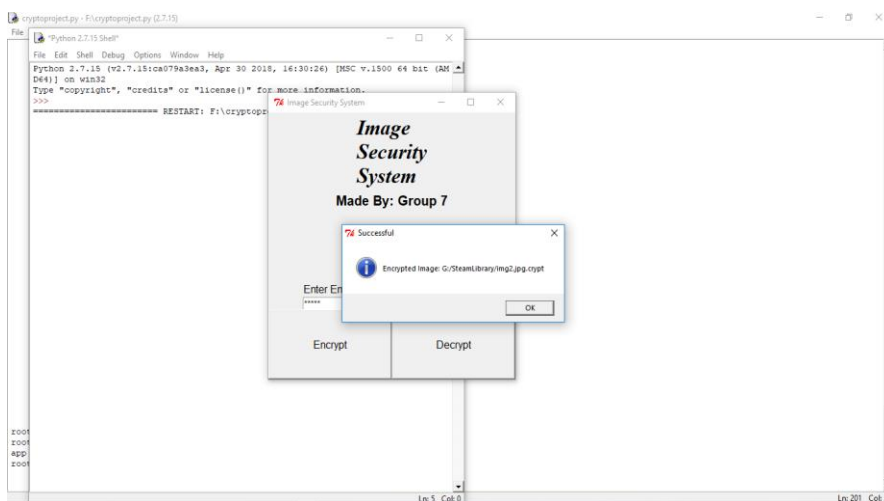
Step5) Enter password that you want (it can be a string or numbers, here it is 12345). Click on “Encrypt” button and another dialog box will open, select the image and click open



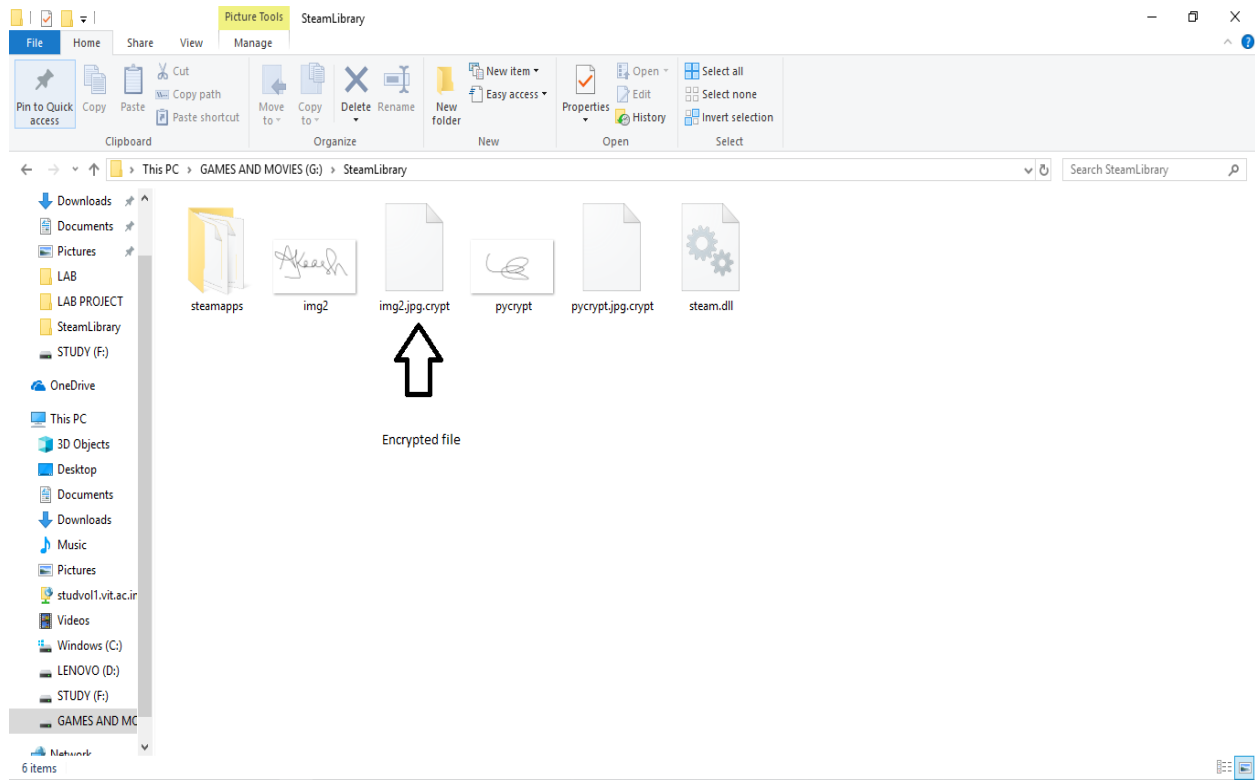
Step6) when you press “Open”. The program will freeze and in sometime a new window will open containing a temporary file which will be the Image constructed by the program using the cypher text.



Step7) Close the photo or save it whichever you like. When you will close the photo you will be greeted with a dialog box

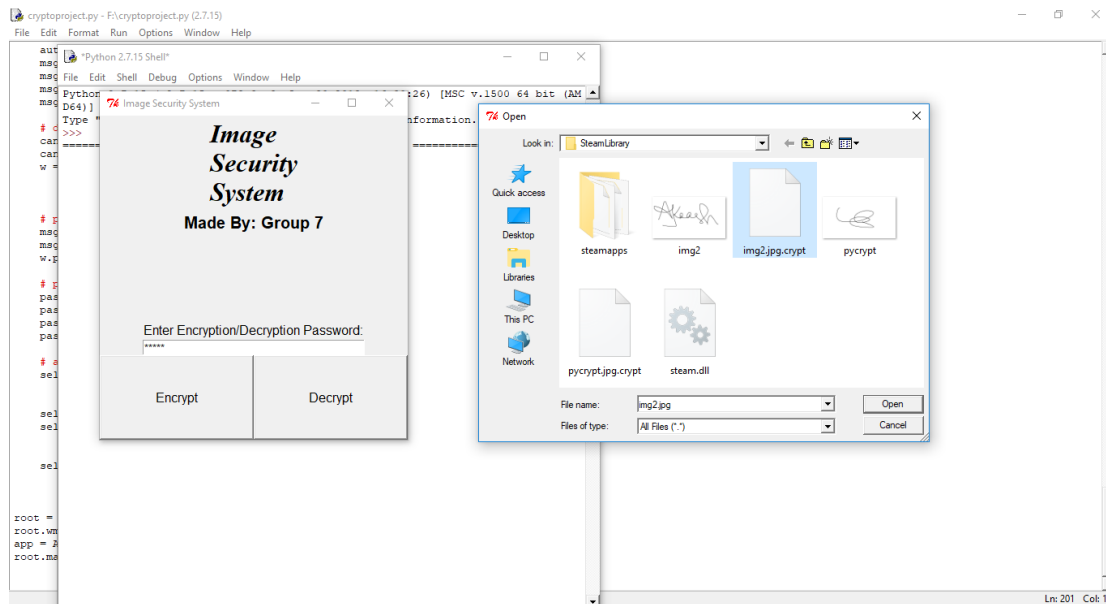


Step8) Go to the original file location and you will see a new file created there with same file name but not an executable. It will contain an extension of “.crypt”.



## DECRYPTION STARTS HERE

Step9) If you run the program again and enter the password (again here password is 12345).Click on Decrypt and a dialog box will open



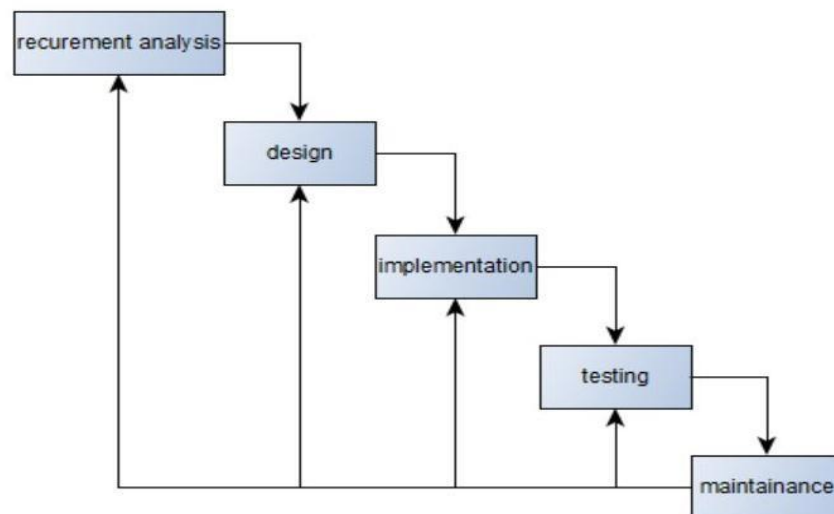
Step10) Select the encrypted file and click on “Open” and wait for some time as program will free and a photo will open contain the original file with a random name.

## 7. Test Cases:

Test ID	Test Case	Expected Outcome	Actual Outcome	Remarks (Pass/Fail)
01	When no paraphrase	Error	Error	Pass
02	Wrong image format	Error	Error	Pass
03	Image does not exist	Error	Error	Pass
04	Wrong paraphrase	Cannot decrypt	Outputs wrong image	Pass
05	Image file present in C:	File encrypted	Error/ File cannot be created	Fail

## Appendix: Analysis Models

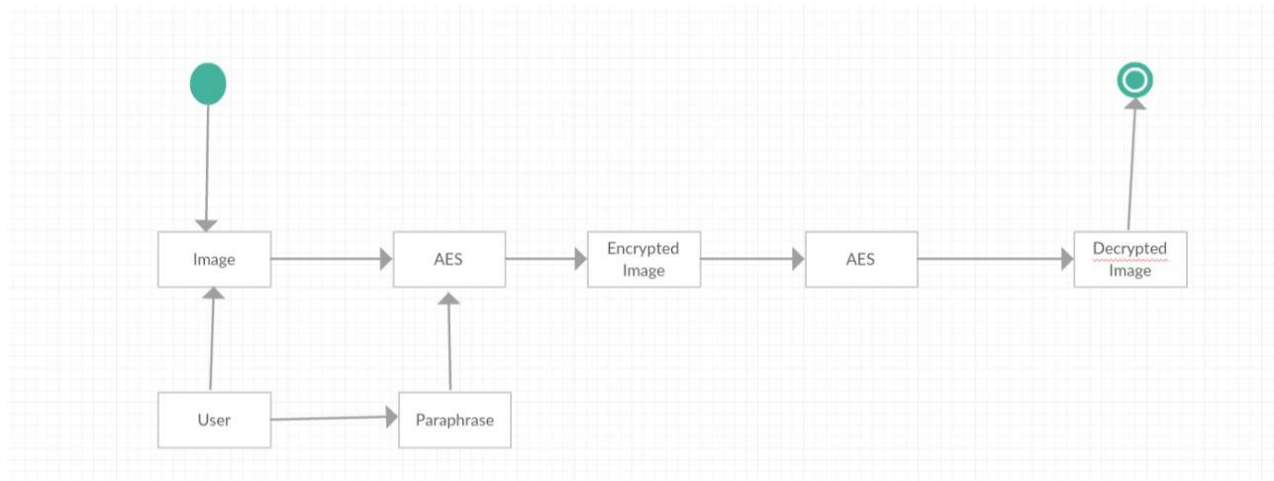
### 1. Software Model:



We are implementing this project using waterfall model.



## 2. State transition diagram:



# **REFERENCES**

## **Python 2.7.15 Release:**

<https://www.python.org/downloads/release/python-2715/>

## **Pillow and Pycrypto:**

<https://web.archive.org/web/20161201092248/http://www.voidspace.org.uk/downloads/pycrypto26/pycrypto-2.6.win-amd64-py2.7.exe>

<https://web.archive.org/web/20161201092248/http://www.voidspace.org.uk/downloads/pycrypto26/pycrypto-2.6.win32-py2.7.exe>

## **General Steganography:**

<https://pdfs.semanticscholar.org/a85d/59cefb53375ad0e5e23df7b5a0431b2eefc5.pdf>

<https://learncryptography.com/steganography/what-is-steganography>

- **Cryptography and network security by Bose and Vijaya Kumar.**
- **Cryptography and network security by William Stallings.**