IMAGE ENCRYPTOR

A Project Report

Submitted as a J[th]Component for the course

MCA

By

RITIK JAIN	**17MCA0093**
Shaziya Nazz S.	**17MCA0022**

Under the Guidance of

Gundala Swathi

School of Information Technology & Engineering

May, 2018

Abstract

In the present world when whole web is now coming on from text data to multimedia data, one of the major security concerns is the protection of this multimedia data. Image, which covers the highest percentage of the multimedia data, its protection is very important. This can be achieved by Image encryption. There are so many different techniques should be used to protect confidential image data from unauthorized access.

In this project I have used Python for programming and Image Scrambling Algorithm for Encryption of Image.

In most software there is always a backend to Brute-Force attack to crack password, I kept it in mind in created such a software on which it is hard to Brute-Force Attack.

How Project Works:

1. Using Tkinter and wxpython GUI has been created.
2. Select .png file you want to encrypt.
3. Select destination folder to encrypted image.
4. Enter a strong password (8 char., Min. 1 lower, 1 upper alphabet, 1 numeric and 1 symbol).
5. Press Encrypt.
6. Using PIL image will be converted in pixel 3D matrix.
7. Convert 3D matrix into 1D using Numpy and operate according to AES Algorithm (using Hashlib, Random).
8. Convert Encrypted text into 3D and into Encrypted Image.
9. File will be saved in destination folder with extension .RTK.
10. To decrypt select file with .RTK extension and destination folder.
11. Enter your password and click on decrypt.
12. Your file will be decrypted in destination folder. Which folder you selected at start.

Why Use This Software?

The Biggest question ever asked why? There are many standard software the why this?

Because it protects encrypted image from Brute-Force attack even when attacker has exact algorithm it is impossible to guess password and to guess password attacker must perform Brute-Force.

This software it will provide an output image on every valid (strong) password, so to check for correct image or password attacker must perform it manually

Which will slows him down by Million Times. So with less secure algorithm, it is more secure than most software.

Introduction

Today web is going towards the multimedia data in which image covers the highest percentage of it. But with the ever-increasing growth of multimedia applications, security is an important aspect in communication and storage of images, and encryption is the way to ensure security. Image encryption techniques try to convert original image to another image that is hard to understand and to keeps the image confidential between users, in other word, it's important that without decryption key no one can access the content. Image encryption has applications in internet communication, multimedia systems, medical imaging, telemedicine, military communication; etc. Images are different from text. Although we may use the traditional cryptosystems to encrypt images directly, it is not a good idea for two reasons. One is that the image size is almost always much greater than that of text. Therefore, the old system takes more time for encrypting the image data directly. The other problem is that the decrypted text must be equal to the original text.

Image Scrambler

Objective evaluation of a scrambling algorithm can be done through unchanged pixel positions, entropy, correlation etc.

There are 2 main approaches of image scrambling:

I. Pixel position based scrambling

II. Pixel value based Scrambling.

In first approach, depending on the scrambling key, pixel positions of the original image are mapped to scrambled image. In this approach, as pixel values are not touched pixel distribution remains same as the original image. In second approach, pixel values are changed according to the scrambling key to get scrambled image. Second approach is more secure as compared to first but at the cost of chance of losing image data. To get highly secure image scrambling algorithm one can effectively combine both approaches. There are also the scrambling methods available which scramble the image by transforming original image to transform domain and then apply image scrambling algorithm to get scrambled image. Commonly an image scrambling algorithm is use for data (image) masking. This algorithm is providing a faster and efficient way to encrypt images.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python interpreters are available for many operating systems Python, is open source software and has a community-based development model.

Libraries used:

Tkinter: Tkinter is most used GUI Library in Python.

Wxpython: Wxpython is second most used GUI Library in Python.

PIL: Python Image Library is used to convert image into pixel matrix and visa-versa.

Random: Used to generate random number.

Hashlib: To generate Hash (SHA256) of password and keys.

Py2exe: To convert Python code into executable file.

Numpy: A large collection of high-level mathematical functions to operate on ND arrays.

OS: OS comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality. The “os” and “os.path” modules include many functions to interact with the file system.

RELATED WORK:

A NEW DIGITAL IMAGE SCRAMBLING METHOD BASED ON FIBONACCI NUMBER.

Image scrambling strategy in view of Fibonacci numbers, the institutionalization and periodicity of the scrambling change are talked about. The scrambling change has the accompanying preferences: Encoding and deciphering is extremely basic and they can be connected in real-time circumstances. The scrambling impact is exceptionally sensible; the information of the image is re-appropriated arbitrarily over the entire image. The strategy can persevere through basic image assaults, for example, pressure, commotion and loss of information parcel. They built up a strategy to think about video scrambling and test relating implanting calculations for advanced watermarks.

EXISTING ENCRYPTION ALGORITHM:

Image Encryption utilizing Digital mark calculation scrambles the image and installs the computerized signature into the image before transmission. This encryption procedure gives three layers of security. In the initial step, a mistake control code is utilized which is resolved continuously, in view of the span of the info image. Without the information of the particular mistake control code, it is exceptionally hard to acquire the first image. The measurement of the image likewise changes due to the additional repetition. This represents an extra trouble to unscramble the image. Likewise, the computerized mark is added to the encoded image in a particular way. At the beneficiary end, the computerized mark can be utilized to confirm the realness of the transmitted image. The calculation which utilizes SCAN dialect has lossless image pressure and encryption capacities. The unmistakable favorable position of synchronous lossless pressure and solid encryption makes the technique exceptionally helpful in applications, for example, medicinal imaging, mixed media

applications, and military applications. The downside of the procedure is that pressure encryption takes longer time. Image scrambling is a generally connected in computerized image watermark innovation. Along these lines, Arnold change is frequently utilized, however its security isn't sufficient. We can pick diverse change coefficient and times in image scrambling, it is hard to re-establish the first image after the change in light of the fact that the change coefficient isn't the no one but, which can enhance the proficiency of scrambling calculation and watermarking security

Implementation

In this project implementation we used “Pixel Position based Scrambling” where we need to change the position of pixel to make it unreadable.

Algorithm

Encryption

Inputs: Image file (gif/bmp/jpg), Destination Folder, Password

Output: Encrypted Image (RTK file)

Method:

Step 1: Taking image file, destination folder, and password from user.
Step 2: Validation of password format, destination folder and image format.
Step 3: Digesting password into SHA256 and getting hex-decimal values.
Step 4: Assigning password to sudo-random to fixing the values of random states to generate fixed random numbers.
Step 5: Converting image file into Sequence of 2D matrix.
Step 6: Shuffling the pixel row by row (using random library).
Step 7: Assigning shuffled pixels to a new sequence. Creating a new Image file.
Step 8: Saving the image into destination folder with extension “.RTK”

Decryption

Inputs: Encrypted Image file(.RTK), Destination Folder, Password

Output: Decrypted Image file (.png)

Method:

Step 1: Taking image file, destination folder, and password from user.
Step 2: Validation of password format, destination folder and image format.
Step 3: Digesting password into SHA256 and getting hex-decimal values.
Step 4: Assigning password to sudo-random to fixing the values of random states to generate fixed random numbers.
Step 5: Converting image file into Sequence of 2D matrix.
Step 6: Shuffling the pixel row by row (using random library).
Step 7: Creating new Sequence of indexes of image row size and shuffling the index.
Step 8: Enumerate through the sequence
and provide the index
for i,j -> enumerate(indexSequence)
NewImage[j] = EncryptedImage[i]
Step 9: Saving the image with extension “.png”

Code-wise Explanation

Code for Creating the User Interface

```
#Importing the necessary Libraries
import wx,sys
from tkinter import *
from tkinter.filedialog import *
from tkinter import Tk, ttk, messagebox
from hashlib import sha256
import Image_operation_Library as op

filename=""
file_destination=""
```

```
tc2 = ""
tc1 = ""
pwd = None

#Input source file
def File_Input(x):
    global filename
    root = Tk()
    filename = askopenfilename(title = "Select File",filetype =
(('PNG','.*png'),('RTK','*.RTK')))
    tc1.SetValue(filename)
    root.destroy()
    return

#Input destination file
def Folder_Input(x):
    global file_destination
    root = Tk()
    file_destination= askdirectory()
    tc2.SetValue(file_destination)
    root.destroy()

#Display Message Box when decryption is done.
def Successfully_Decrypted():
    root = Tk()
    root.withdraw()
    messagebox.showinfo("Success", "You Image has been Decrypted")
    return

#Display Message Box when encryption is done.
def Successfully_encrypted():
    root = Tk()
    root.withdraw()
    messagebox.showinfo("Success", "You Image has been Encrypted")
```

```
    return

# About US page
def about_us_panel(x):
    root = Tk()
    root.withdraw()
    messagebox.showinfo("About US","Ritik Jain \nritikjain51@gmail.com\nShaziya
Nazz S\nshaziyanazz.s2017@vitstudent.ac.in")
    return

#Initial Frame method
class RootFrame(wx.Frame):

    def __init__(self, parent, title):
        super(RootFrame, self).__init__(parent, title=title, size=(500, 300))

        self.InitUI()
        self.Centre()
        self.Show()

    def InitUI(self):
#Start creating the panel
          global password
          panel = wx.Panel(self)

         #Create a grid layout to dividing the panel into 5,5 maze
          sizer = wx.GridBagSizer(5, 5)
          #Inserting Image
          #icon = wx.StaticBitmap(panel, bitmap=wx.Bitmap('icon.jpg'))
          #sizer.Add(icon,                                pos=(0,                                4),
flag=wx.TOP|wx.RIGHT|wx.ALIGN_RIGHT,border=5)

          #Creating a static page
          line = wx.StaticLine(panel)
```

```
        sizer.Add(line, pos=(1, 0), span=(1, 5), flag=wx.EXPAND|wx.BOTTOM,
border=10)

        global tc2,tc1

        #Label printing
        text2 = wx.StaticText(panel, label="Browse Source File")
        sizer.Add(text2, pos=(2, 0), flag=wx.LEFT|wx.TOP, border=10)

        # Insert Textbox
        tc1 = wx.TextCtrl(panel)
        sizer.Add(tc1, pos=(2, 1), span=(1, 3), flag=wx.TOP|wx.EXPAND, border=5)

        #Button for source input
        button1 = wx.Button(panel, label="Browse...")
        sizer.Add(button1, pos=(2, 4), flag=wx.TOP|wx.RIGHT, border=5)
        button1.Bind(wx.EVT_BUTTON,File_Input)

        #Textbox for destination folder input
        text3 = wx.StaticText(panel, label="Browse Destination Folder")
        sizer.Add(text3, pos=(3, 0), flag=wx.LEFT|wx.TOP, border=10)

        #Textbox for Textbox
        tc2 = wx.TextCtrl(panel)
        sizer.Add(tc2, pos=(3, 1), span=(1, 3), flag=wx.TOP|wx.EXPAND, border=5)

        #Button for destination folder input
        button2 = wx.Button(panel, label="Browse...")
        sizer.Add(button2, pos=(3, 4), flag=wx.TOP|wx.RIGHT, border=5)
        button2.Bind(wx.EVT_BUTTON,Folder_Input)

       #Inserting the label
        text4 = wx.StaticText(panel, label="Enter Password")
        sizer.Add(text4, pos=(4, 0), flag=wx.LEFT, border=10)
```

```
        #Textbox for password
        global pwd
        pwd = wx.TextCtrl(panel, style= wx.TE_PASSWORD)
        sizer.Add(pwd, pos=(4, 1), span=(1, 3), flag=wx.EXPAND)

       # Inserting the Encryption Button on page layout.
        button4 = wx.Button(panel, label="Encrypt")
        sizer.Add(button4, pos=(5, 3))
        button4.Bind(wx.EVT_BUTTON,image_open)

        # Inserting the Decryption Button
        button5 = wx.Button(panel, label="Decrypt")
        sizer.Add(button5, pos=(5, 4), span=(1, 1), flag=wx.BOTTOM|wx.RIGHT,
border=5)
        button5.Bind(wx.EVT_BUTTON,cipher_open)

       #Inserting About US and calling the procedure s
        button6 = wx.Button(panel, label= 'About US')
        sizer.Add(button6, pos=(0,0), span=(0,1))
        button6.Bind(wx.EVT_BUTTON, about_us_panel)

        sizer.AddGrowableCol(2)
        panel.SetSizer(sizer)


def checkpassword(password):

    '''
    Password consist of Lower characters,
    upper characters, digits, symbols
    '''
    import string
```

```
    # Making boolean variables
    flags = flagc = flagd = flagsy = False


    if len(password)<8:
        return False

   #Checking for password string for lowercase Letter.
  # Checking password string for Uppercase Letter.
  # Checking password string for digits
 # Checking password string for punctuation( Symbols)
    for i in password:
        if i in string.ascii_lowercase:
            flags = True
        elif i in string.ascii_uppercase:
            flagc = True
        elif i in string.digits:
            flagd = True
        elif i in string.punctuation + " ":
            flagsy = True

# If everything is present then return true else return  false.
    if flags and flagc and flagd and flagsy:
        return True
    else:
        return False


# This function call when user clicked on Encrypt button. And complete Validation of
Encryption end.
def image_open(x):

    #Calling global Variables
    global filename
```

```
global file_destination
global pwd
password = pwd.GetValue()

root = Tk()

x = ('.PNG','.png')

#checking for source file
if filename == "":
    messagebox.showerror("Error", "Select File")

#checking for destination
elif file_destination == "":
    messagebox.showerror("Error", "Select File Destination")

#Checking for extension
elif  not (filename.endswith(x)):
    messagebox.showerror("Error", "Invalid File")

#Checking for null password
elif password == "":
    messagebox.showerror("Error", "Enter Password")

elif password != "" and filename != "":
    if not checkpassword(password):
      messagebox.showinfo("Error", "Enter a strong password")
    else:
        password = sha256(password.encode()).hexdigest()
        op.encrypt_image(password,filename,file_destination)
        Successfully_encrypted()
```

```
# This function call when user clicked on Decrypt button. And complete Validation of Decryption end before calling main algorithm
def cipher_open(x):

    global filename
    global file_destination
    global pwd
    password = pwd.GetValue()

    root = Tk()



    x = ('.RTK')
    #Variable Validation
        #checking for source file
    if filename == "":
        messagebox.showerror("Error", "Select File")

    #checking for destination
    elif file_destination == "":
        messagebox.showerror("Error", "Select File Destination")

    #Checking for extension
    elif  not (filename.endswith(x)):
        messagebox.showerror("Error", "Invalid File")

    #Checking for null password
    elif password == "":
        messagebox.showerror("Error", "Enter Password")

    elif password != "" and filename != "":
        if not checkpassword(password):
          messagebox.showinfo("Error", "Enter a strong password")
```

```
else :
            password = sha256(password.encode()).hexdigest()
            op.decrypt_image(password,filename,file_destination)
            Successfully_Decrypted()



if __name__ == '__main__':
  #Calling the class name
    app = wx.App()
    RootFrame(None, title="Image Encryptor")
    app.MainLoop()
```

Main Image Encryption Algorithm

```
#Importing libraries for converting image into matrix, random number and os

from PIL import Image
import random
import os

# Function for encryption of image
def encrypt_image(password, filename, file_destination):

    #Image Open with PIL Library
     im = Image.open(filename)

    #Extracting the pixels from image
     pixelList = list(im.getdata())

    #Fixing the random seeding values
     random.seed(password)

    # Shuffling the pixel row by row using random library
```

```
    random.shuffle(pixelList)
  #Creating a new Image
    img = Image.new(im.mode, im.size)
   # Putting pixels to the new image
    img.putdata(pixelList)
   taking the index from filename
    index = filename.rindex('/')
    count = 0

  #Checking for filename with extension RTK if present then create with number
    while os.path.exists(file_destination +"/"+ filename[index:-4]+ ".encrypted" +
str(count) +".RTK"):
        count += 1

    # Saving image into destination folder with filename and extension
    filename = file_destination+"/"+filename[index:-4]+ ".encrypted" + str(count)
+".RTK"

   #Creating a temporary image.
    temp = file_destination + "/" + password+"temp.png"

   #Changing the image file name and saving
    img.save(temp)
    img.show()
    os.rename(temp,filename)
    img.close()
    im.close()

# Function for decryption of image
def decrypt_image(password, filename,file_destination):

  #Creating a temporary image with the password names
    index = filename.rindex("/")
```

```
 #Storing the image with .png format for temp purpose
 temp = filename[:index] + "/" + password +"temp.png"

 # Renaming the filename with temporary filename
 os.rename(filename,temp)

# Opening the image for performing operation
 im = Image.open(temp)

#Extracting the pixels from image
 pixelList = list(im.getdata())
 #Creating a list with size of pixelList
 indexVal = list(range(len(pixelList)))

#Fixing the random seeding values
 random.seed(password)



 #Shuffling the random index using the seeded values
 random.shuffle(indexVal)

 # Creating new image with size of old image
 NewPixelList = [0] * (im.size[0] * im.size[1])

# Enumerating the pixels by pixel and storing into new Image array
 for i,j in enumerate(indexVal):
     NewPixelList[j] = pixelList[i]

#Creating new Image to insert the pixels
 out = Image.new(im.mode, im.size)

#Putting pixels to the Image
 out.putdata(NewPixelList)
```

```
    # Image renaming
     os.rename(temp,filename[:-4]+".RTK")
     count = 0

   #Checking for filename with extension png if present then create with number
     while os.path.exists(file_destination +"/"+ filename[index:-4].split(".")[0]+
".decrypted" + str(count) +".png"):
          count += 1

   # Saving image into destination folder with filename and extension
     filename = file_destination+"/"+filename[index:-4].split(".")[0]+ ".decrypted" +
str(count) +".png"

    # Replacing the filename with a temporary name
     temp = file_destination + "/" + password+"temp.png"

   #Saving the image with the temp filename
     out.save(temp)

#Displaying the image after decryption
     out.show()

     os.rename(temp, filename)
     out.close()
     im.close()
```

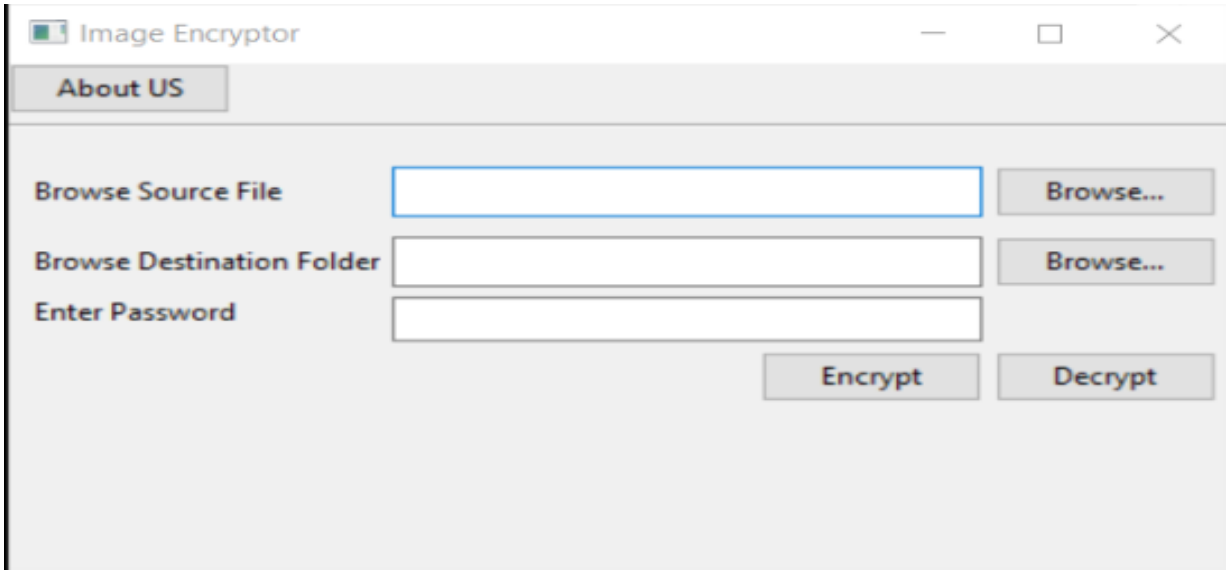
Project Screenshots

Requirement:

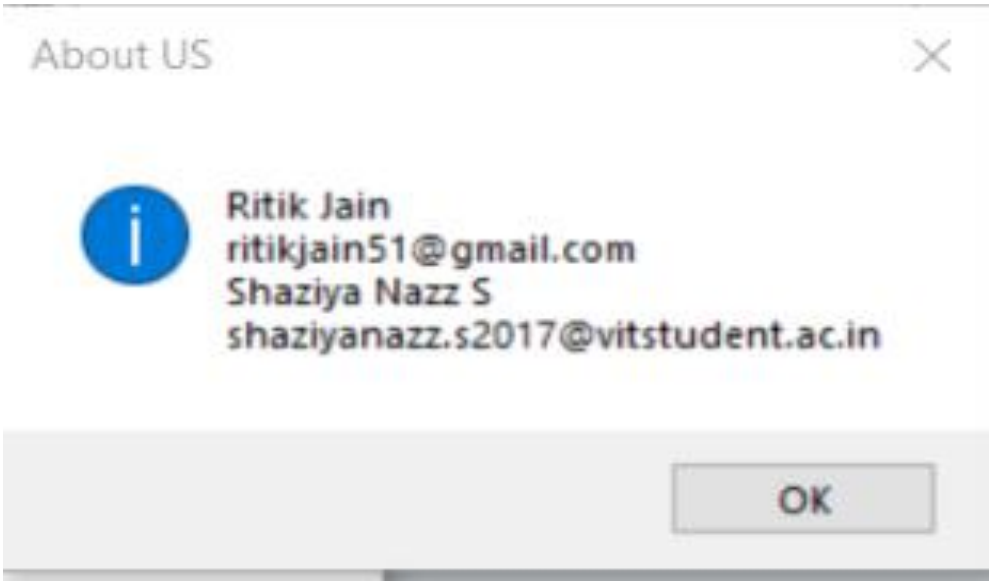
- Python 3.5.x or later
- Pillow or PIL installed
- Python Libraries
- wxPython
- Tkinter

Original Image

First Page of Project



About Us Page



For Encryption

Image Selection for Encryption

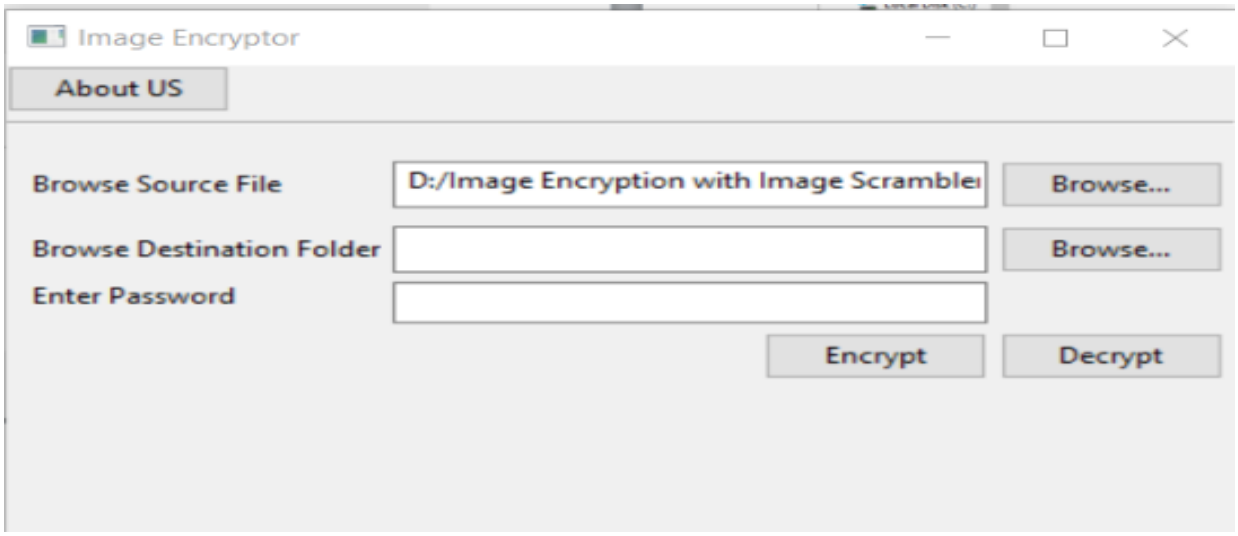
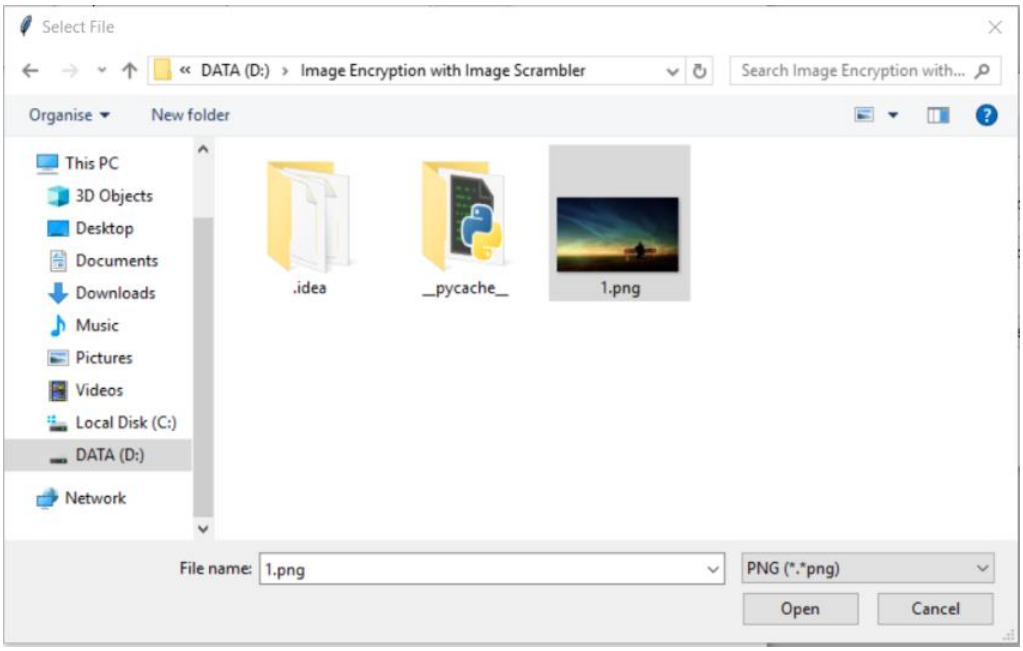
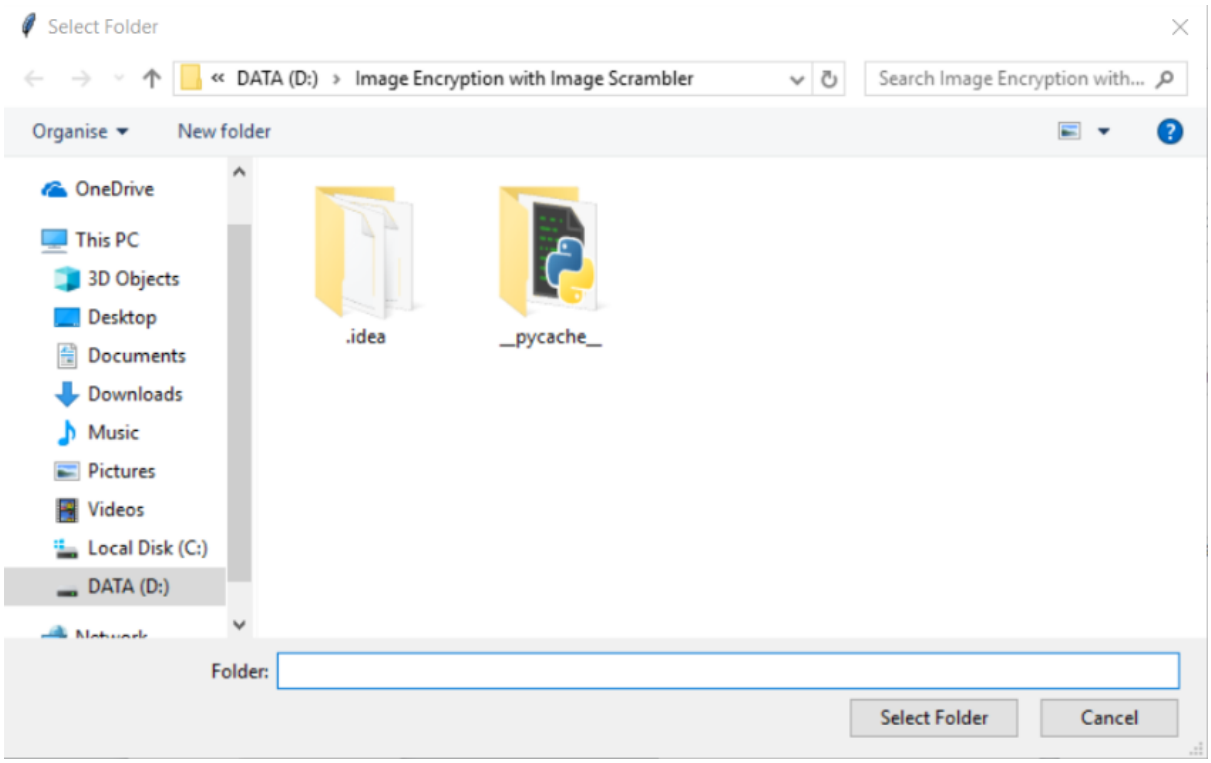


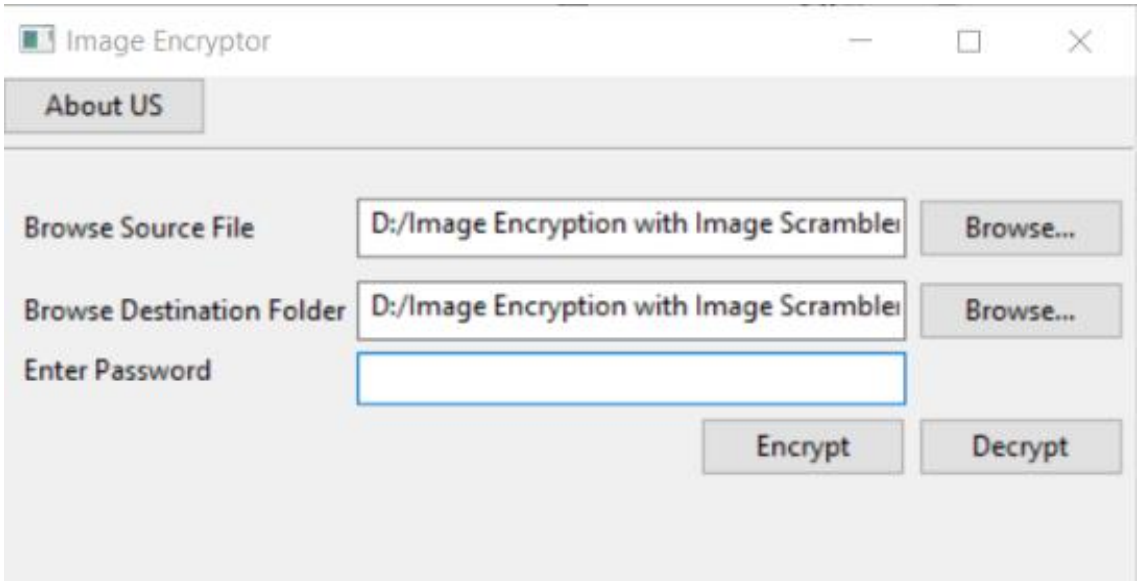
Image Selection from folder



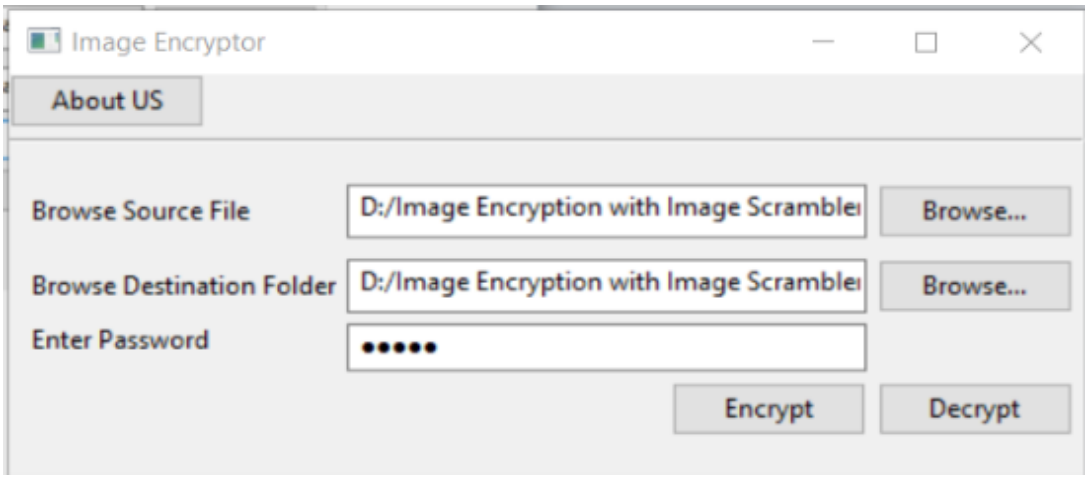
Destination folder selection



After folder Selection



Password Insertion



Password Validation

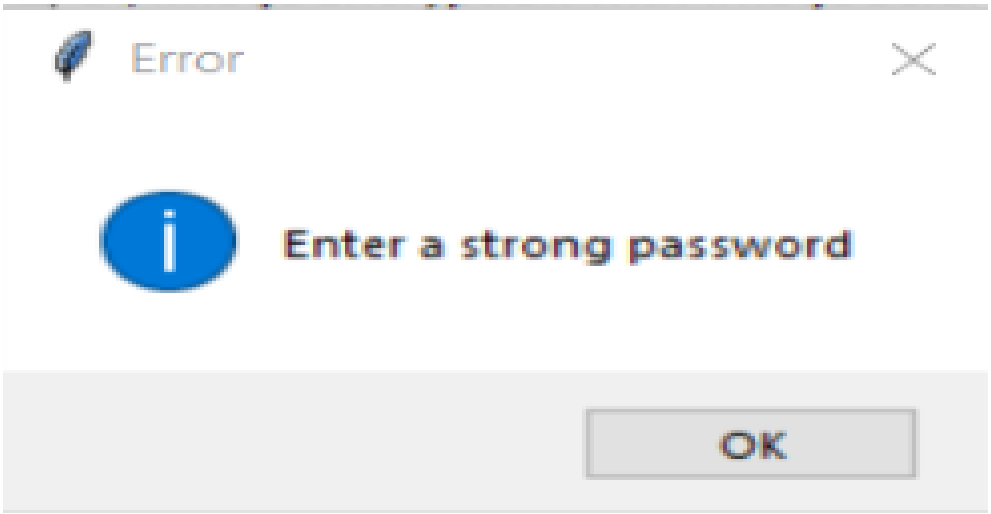
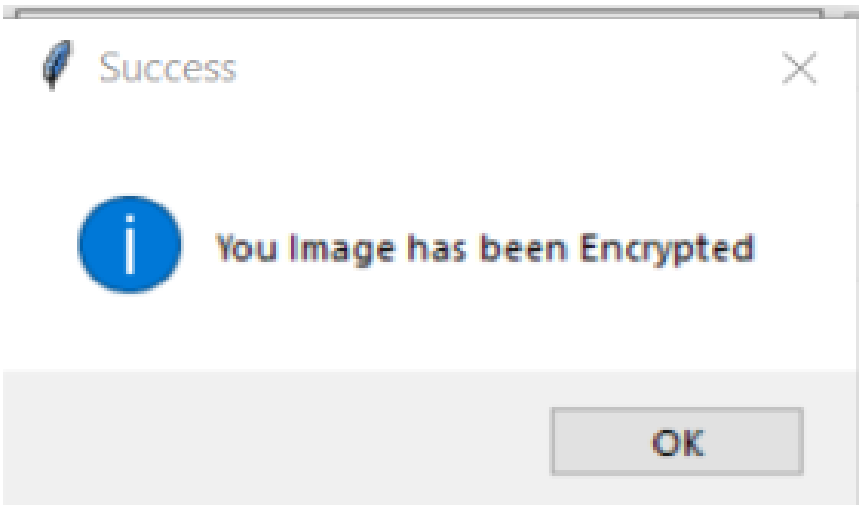


Image after Encryption

Image Encryption completed PopUp message



For Decryption

Selecting .RTK file

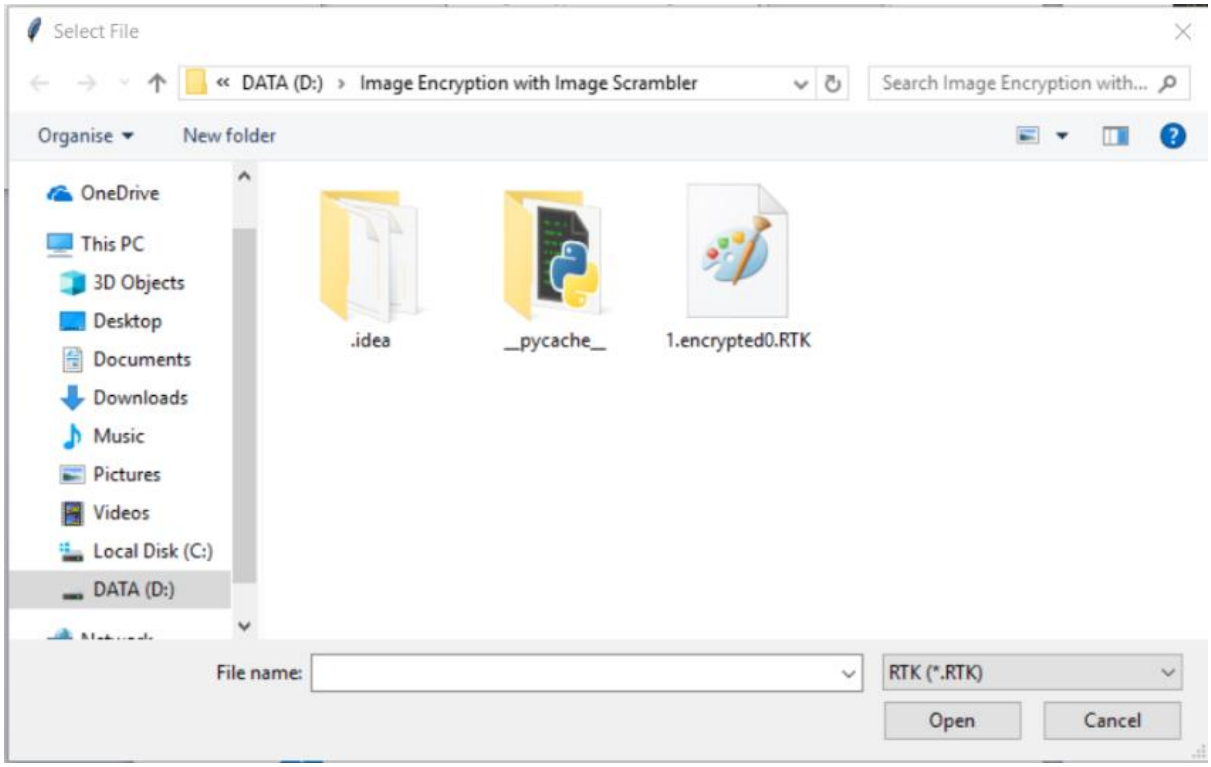


Image Decryption Completed PopUp message

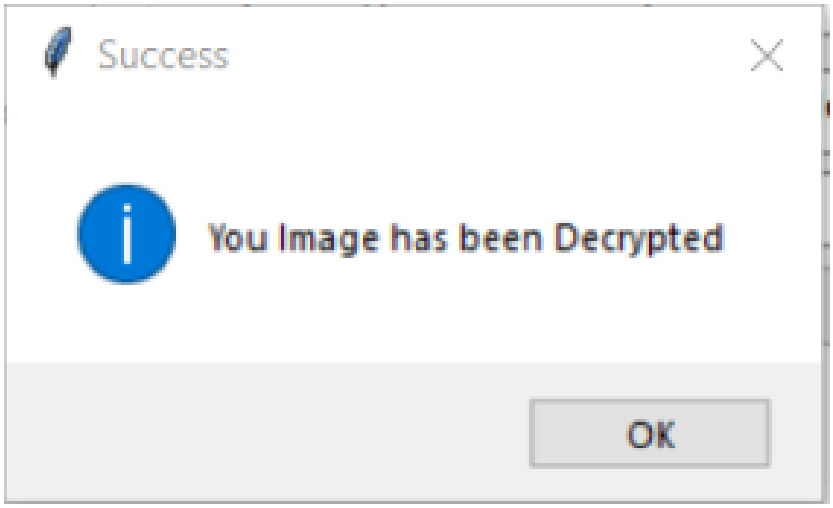


Image after decryption

Conclusion

This is bug free proper working software which can ensure to provide a highly secure algorithm to encrypt Image (png format). It is also hard to Brute-Force on encrypted file. By using Random functions it makes more complex to guess password. This software encrypt/decrypt with wrong password to transforming the password cracking speed from Computer time to human time.