IMAGE ENCRYPTION AND DECRYPTION USING AES ALGORITHM

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

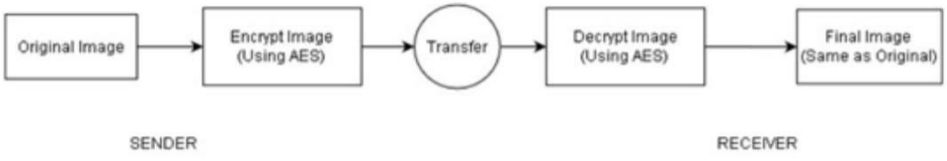
• Secure transfer of an image over the network such as internet

• 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.



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

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 unauthorized access is not supported.

Non-Functional Requirements

Performance Requirements

- For smooth & efficient encryption, image size must be large.
- Decryption should not take more time.

Safety and Security Requirements

- If the decryption takes more time, 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 buttons, 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

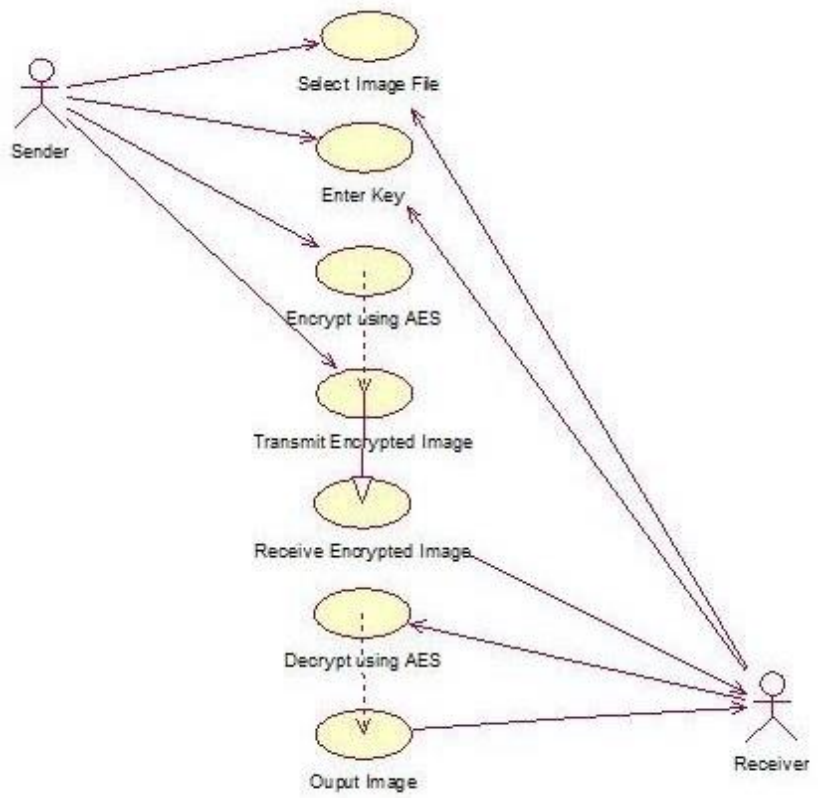
- Sender Computer: for seeing the original and encrypted image
- Receiver Computer: for seeing the decrypted image

Software Interfaces

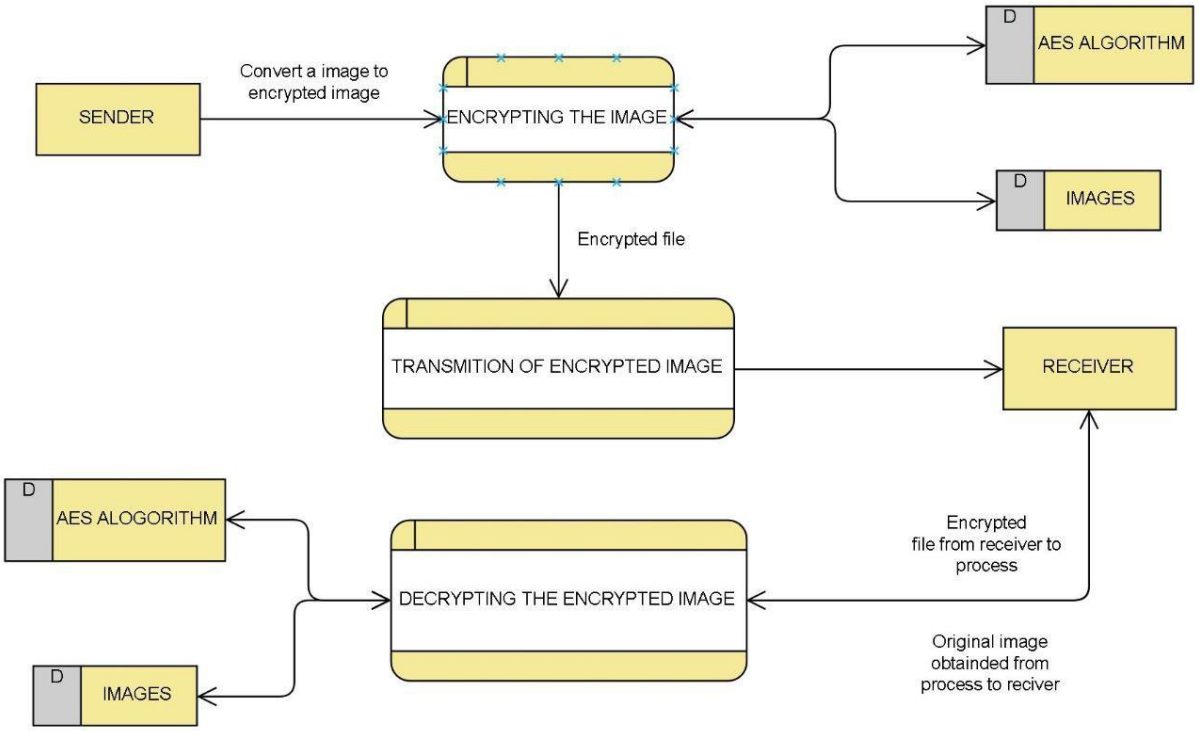
We have frontend made using python module named Tkinter(). This provides an interactive window for the user. The user needs to provide the location of the image to be encoded/decoded. After this, user can either go for encoding or decoding the image. It asks for a password, which is basically your encryption 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.

Design Diagrams: -

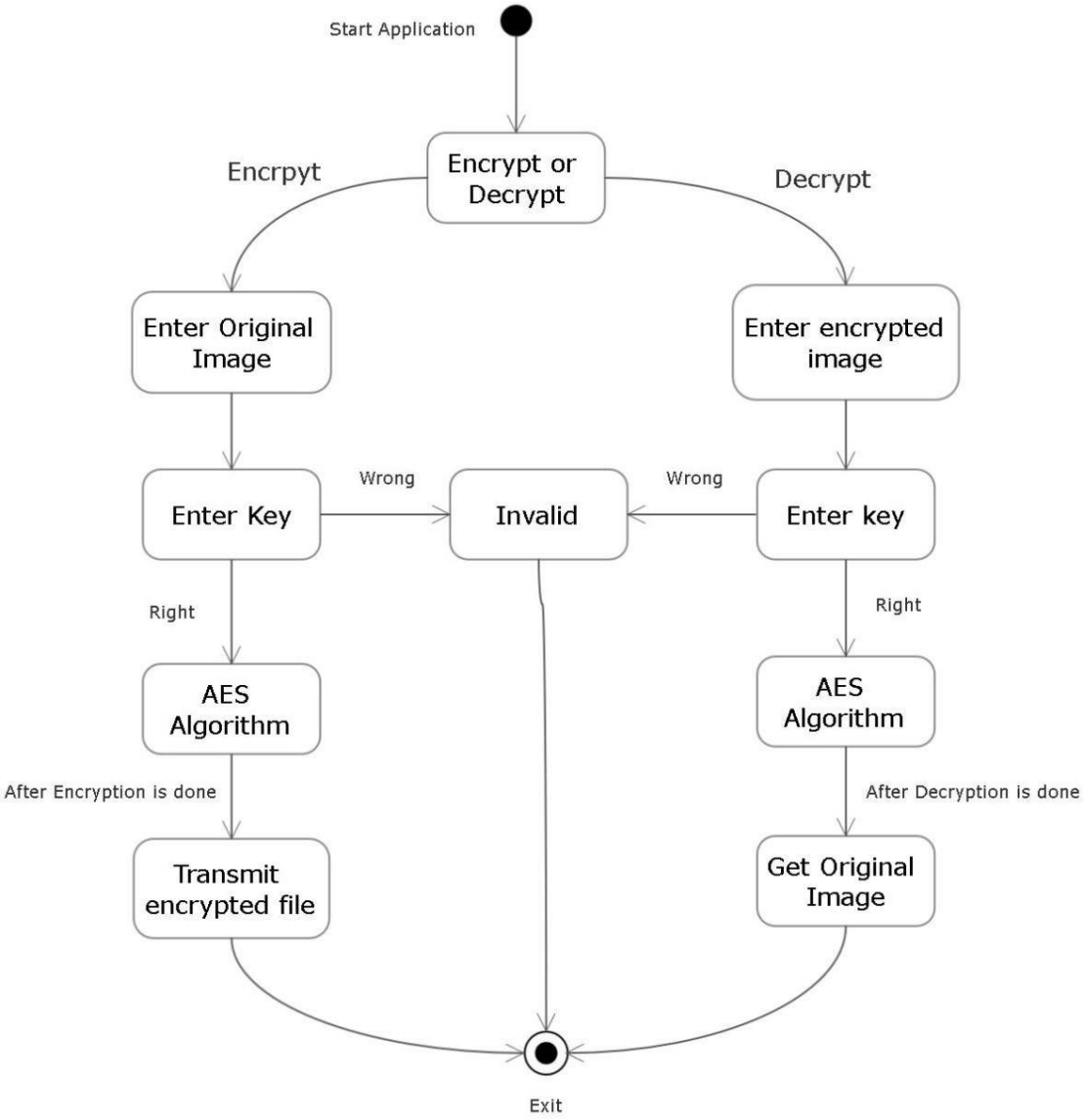
Use case Diagram:



Dataflow Diagram:



State Chart Diagram:



Screenshot of Implementation

Code

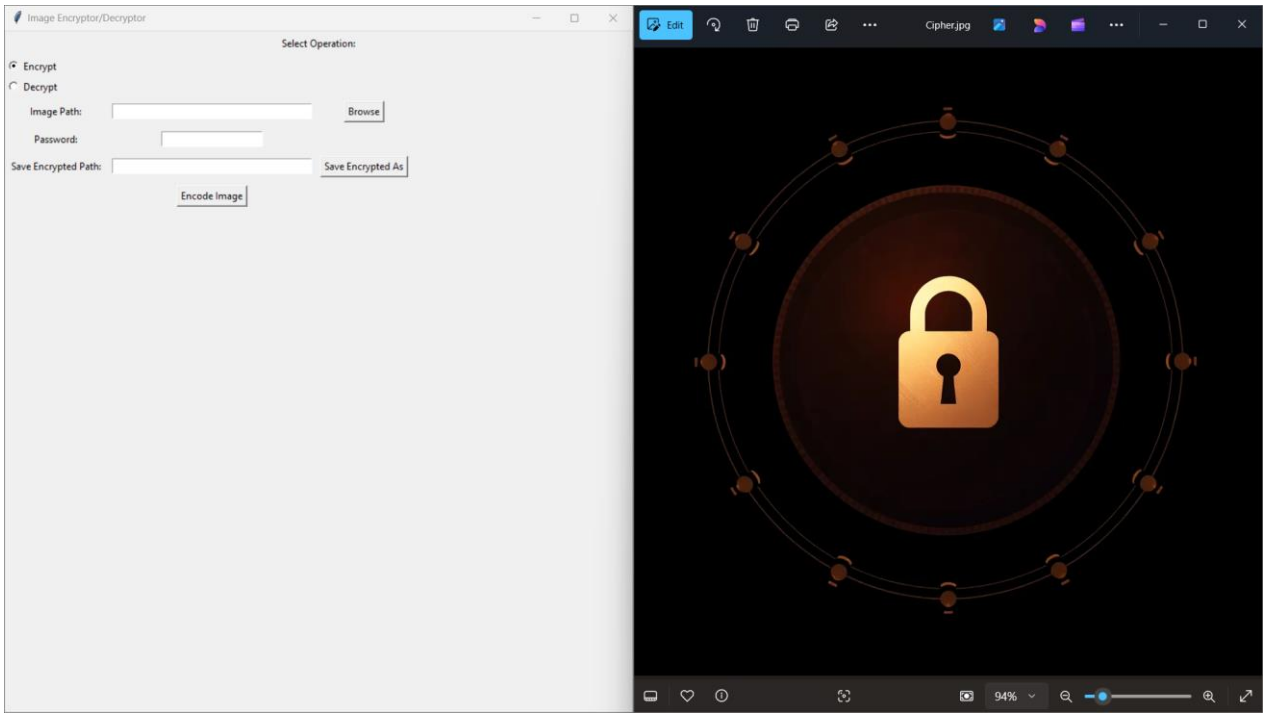
```python
import os
import tkinter as tk
from tkinter import filedialog, messagebox
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
import base64

# Function to derive a key from a password
def derive_key(password):
    salt = os.urandom(16)  # Use a random salt for security
    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        length=32,  # AES key length of 32 bytes
        salt=salt,
        iterations=100000,
        backend=default_backend()
    )
    key = kdf.derive(password.encode())
    return key, salt

# Function to encrypt the image
def encrypt_image(image_path, password, save_path):
    with open(image_path, 'rb') as file:
        original_image = file.read()

    key, salt = derive_key(password)

    # Initialization vector for AES
    iv = os.urandom(16)  # AES block size is 16 bytes
    cipher = Cipher(algorithms.AES(key), modes.CFB(iv), backend=default_backend())
    encryptor = cipher.encryptor()
    encrypted_image = iv + salt + encryptor.update(original_image) + encryptor.finalize()

    with open(save_path, 'wb') as file:
        file.write(encrypted_image)

    return save_path

# Function to decrypt the image
def decrypt_image(encrypted_image_path, password, save_path):
    with open(encrypted_image_path, 'rb') as file:
        encrypted_image = file.read()

    iv = encrypted_image[:16]  # Extract the IV
    salt = encrypted_image[16:32]  # Extract the salt
    ciphertext = encrypted_image[32:]  # The rest is the actual encrypted data
```

```python
 # Derive the key using the extracted salt
    key = derive_key_from_salt(password, salt)

    cipher = Cipher(algorithms.AES(key), modes.CFB(iv), backend=default_backend())
    decryptor = cipher.decryptor()
    decrypted_image = decryptor.update(ciphertext) + decryptor.finalize()

    with open(save_path, 'wb') as file:
        file.write(decrypted_image)

    return save_path

def derive_key_from_salt(password, salt):
    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        length=32,  # AES key length of 32 bytes
        salt=salt,
        iterations=100000,
        backend=default_backend()
    )
    return kdf.derive(password.encode())

# Function to show/hide fields based on selection
def show_fields(option):
    if option == "Encrypt":
        encrypt_frame.pack(fill=tk.BOTH, expand=True)
        decrypt_frame.pack_forget()
    elif option == "Decrypt":
        decrypt_frame.pack(fill=tk.BOTH, expand=True)
        encrypt_frame.pack_forget()

# UI Functions
def select_image():
    file_path = filedialog.askopenfilename(filetypes=[("Image Files", "*.jpg;*.jpeg;*.png")])
    if file_path:
        image_path_var.set(file_path)

def save_encrypted_image():
    format_choice = messagebox.askquestion("Choose Format", "Do you want to save as PNG?", icon='question')
    extension = ".png" if format_choice == 'yes' else ".jpg"

    save_path = filedialog.asksaveasfilename(defaultextension=extension, filetypes=[("Image Files", "*.png;*.jpg")])
    if save_path:
        save_encrypted_path_var.set(save_path)

def save_decrypted_image():
    save_path = filedialog.asksaveasfilename(defaultextension=".jpg", filetypes=[("Image Files", "*.jpg;*.jpeg;*.png")])
    if save_path:
        save_decrypted_path_var.set(save_path)

```

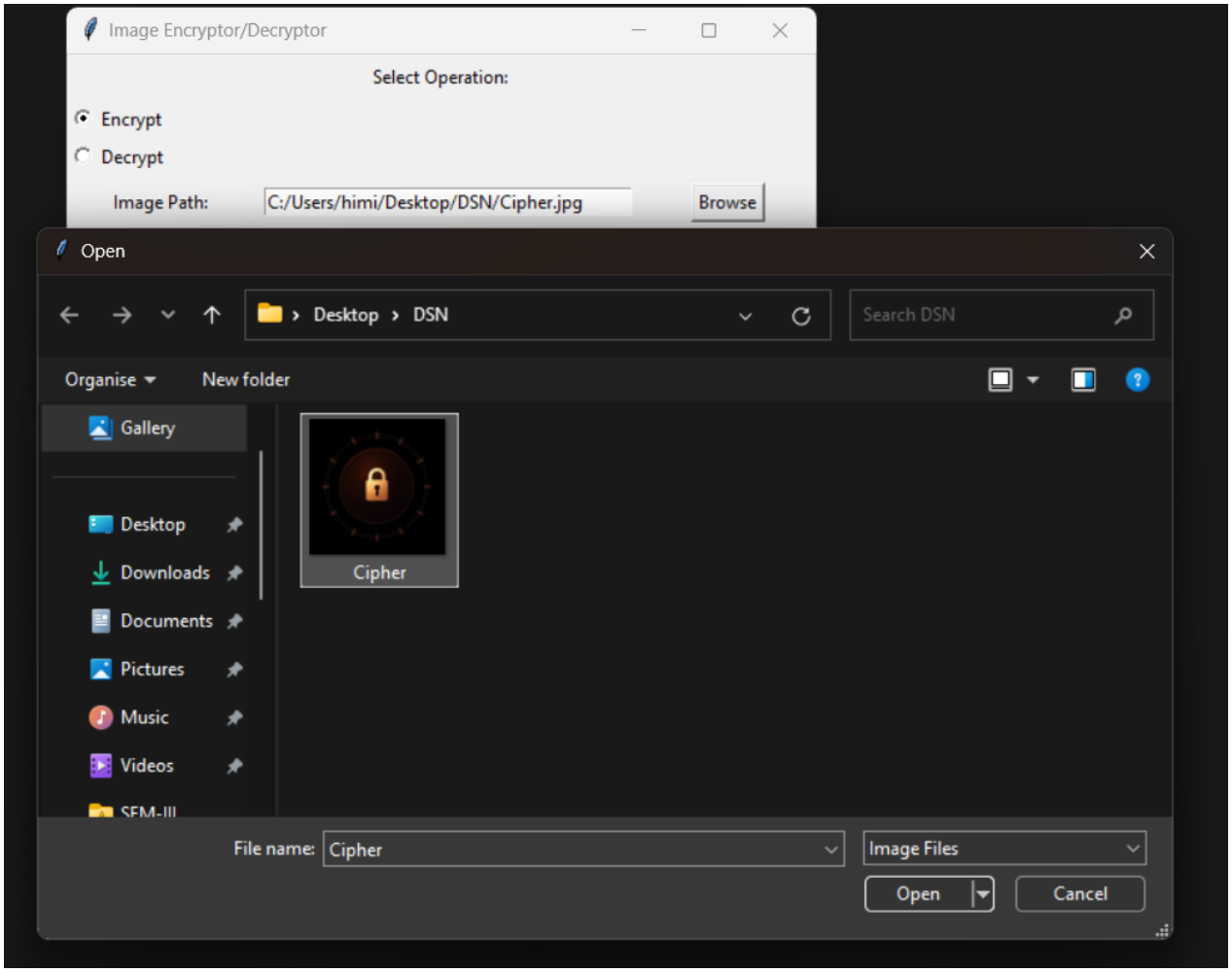
```python
def encode_image():
    image_path = image_path_var.get()
    password = password_var.get()
    save_path = save_encrypted_path_var.get()

    if not image_path or not password or not save_path:
        messagebox.showerror("Error", "Please select an image, enter a password, and specify a save location.")
        return

    encrypted_image_path = encrypt_image(image_path, password, save_path)
    messagebox.showinfo("Success", f"Image encrypted and saved as {encrypted_image_path}")

def decode_image():
    encrypted_image_path = image_path_var.get()
    password = password_var.get()
    save_path = save_decrypted_path_var.get()

    if not encrypted_image_path or not password or not save_path:
        messagebox.showerror("Error", "Please select an encrypted image, enter a password, and specify a save location.")
        return

    decrypted_image_path = decrypt_image(encrypted_image_path, password, save_path)
    messagebox.showinfo("Success", f"Image decrypted and saved as {decrypted_image_path}")

# GUI Setup
root = tk.Tk()
root.title("Image Encryptor/Decryptor")

# Option selection
option_var = tk.StringVar(value="Encrypt")

tk.Label(root, text="Select Operation:").pack(pady=5)
tk.Radiobutton(root, text="Encrypt", variable=option_var, value="Encrypt", command=lambda: show_fields("Encrypt")).pack(anchor=tk.W)
tk.Radiobutton(root, text="Decrypt", variable=option_var, value="Decrypt", command=lambda: show_fields("Decrypt")).pack(anchor=tk.W)

# Frames for Encrypt and Decrypt
encrypt_frame = tk.Frame(root)
decrypt_frame = tk.Frame(root)

```

```python
# Encrypt Frame
image_path_var = tk.StringVar()
password_var = tk.StringVar()
save_encrypted_path_var = tk.StringVar()

tk.Label(encrypt_frame, text="Image Path:").grid(row=0, column=0, padx=5, pady=5)
tk.Entry(encrypt_frame, textvariable=image_path_var, width=40).grid(row=0, column=1, padx=5, pady=5)
tk.Button(encrypt_frame, text="Browse", command=select_image).grid(row=0, column=2, padx=5, pady=5)

tk.Label(encrypt_frame, text="Password:").grid(row=1, column=0, padx=5, pady=5)
tk.Entry(encrypt_frame, textvariable=password_var, show='*').grid(row=1, column=1, padx=5, pady=5)

tk.Label(encrypt_frame, text="Save Encrypted Path:").grid(row=2, column=0, padx=5, pady=5)
tk.Entry(encrypt_frame, textvariable=save_encrypted_path_var, width=40).grid(row=2, column=1, padx=5, pady=5)
tk.Button(encrypt_frame, text="Save Encrypted As", command=save_encrypted_image).grid(row=2, column=2, padx=5, pady=5)

tk.Button(encrypt_frame, text="Encode Image", command=encode_image).grid(row=3, column=1, padx=5, pady=5)

# Decrypt Frame
decrypt_frame = tk.Frame(root)

save_decrypted_path_var = tk.StringVar()

tk.Label(decrypt_frame, text="Encrypted Image Path:").grid(row=0, column=0, padx=5, pady=5)
tk.Entry(decrypt_frame, textvariable=image_path_var, width=40).grid(row=0, column=1, padx=5, pady=5)
tk.Button(decrypt_frame, text="Browse", command=select_image).grid(row=0, column=2, padx=5, pady=5)

tk.Label(decrypt_frame, text="Password:").grid(row=1, column=0, padx=5, pady=5)
tk.Entry(decrypt_frame, textvariable=password_var, show='*').grid(row=1, column=1, padx=5, pady=5)

tk.Label(decrypt_frame, text="Save Decrypted Path:").grid(row=2, column=0, padx=5, pady=5)
tk.Entry(decrypt_frame, textvariable=save_decrypted_path_var, width=40).grid(row=2, column=1, padx=5, pady=5)
tk.Button(decrypt_frame, text="Save Decrypted As", command=save_decrypted_image).grid(row=2, column=2, padx=5, pady=5)

tk.Button(decrypt_frame, text="Decode Image", command=decode_image).grid(row=3, column=1, padx=5, pady=5)

# Show encrypt fields by default
show_fields("Encrypt")

root.mainloop()

```

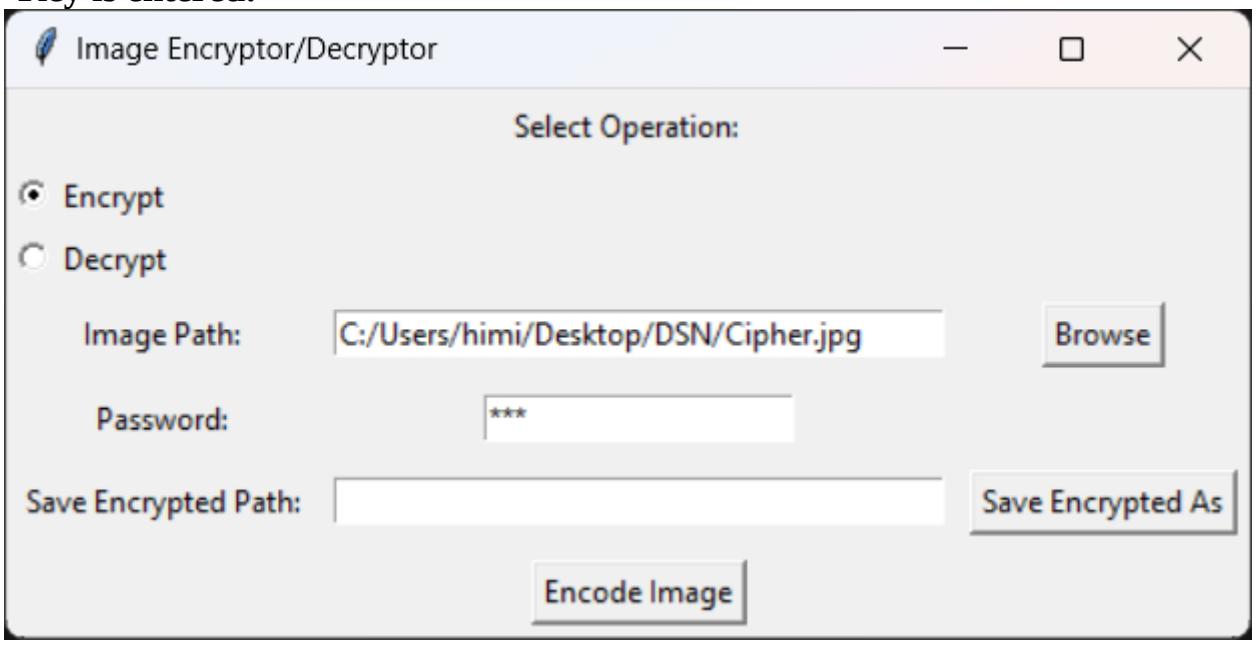
Original Image and GUI Interface:



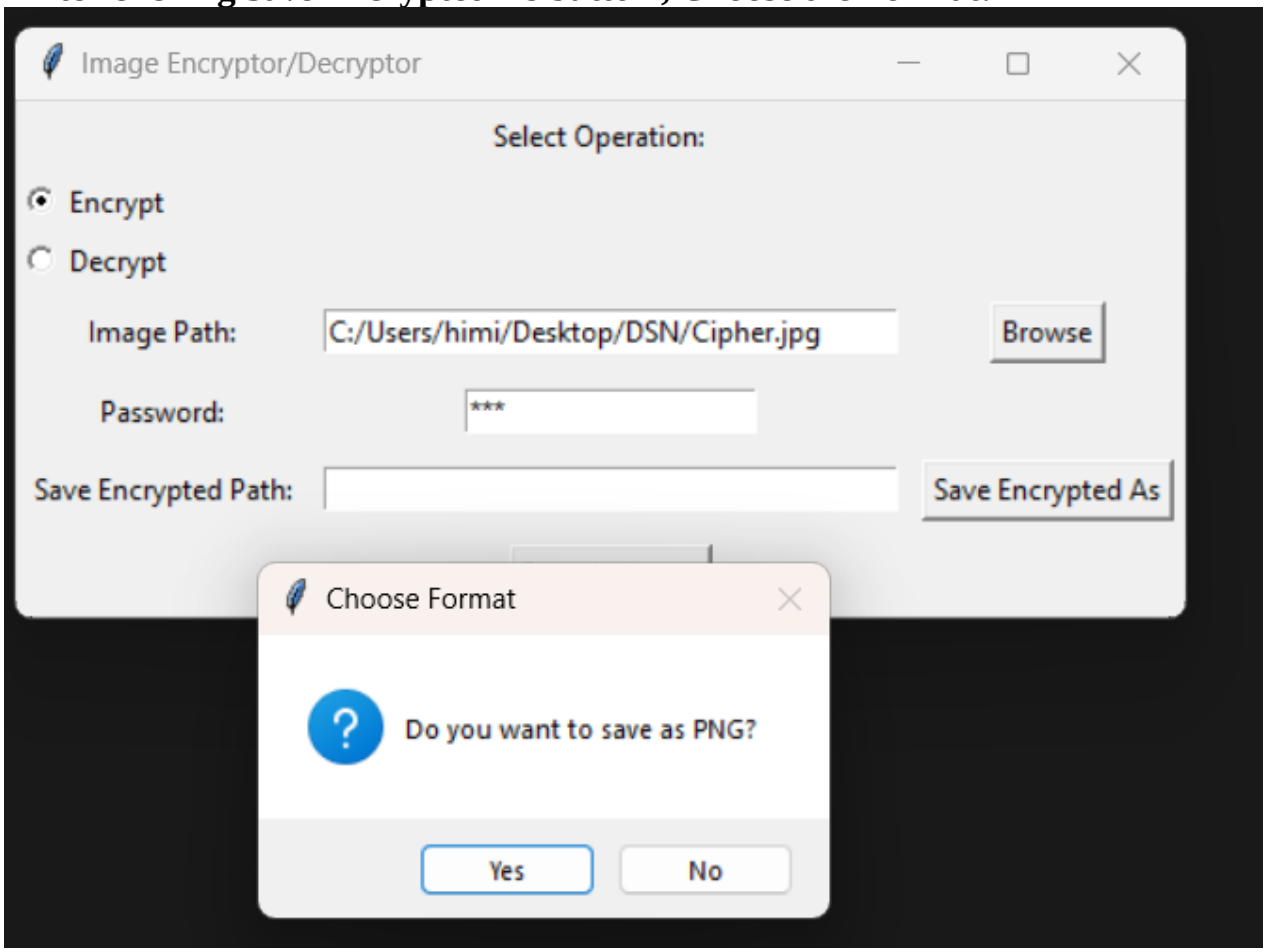
Application's first screen asking for path of image:



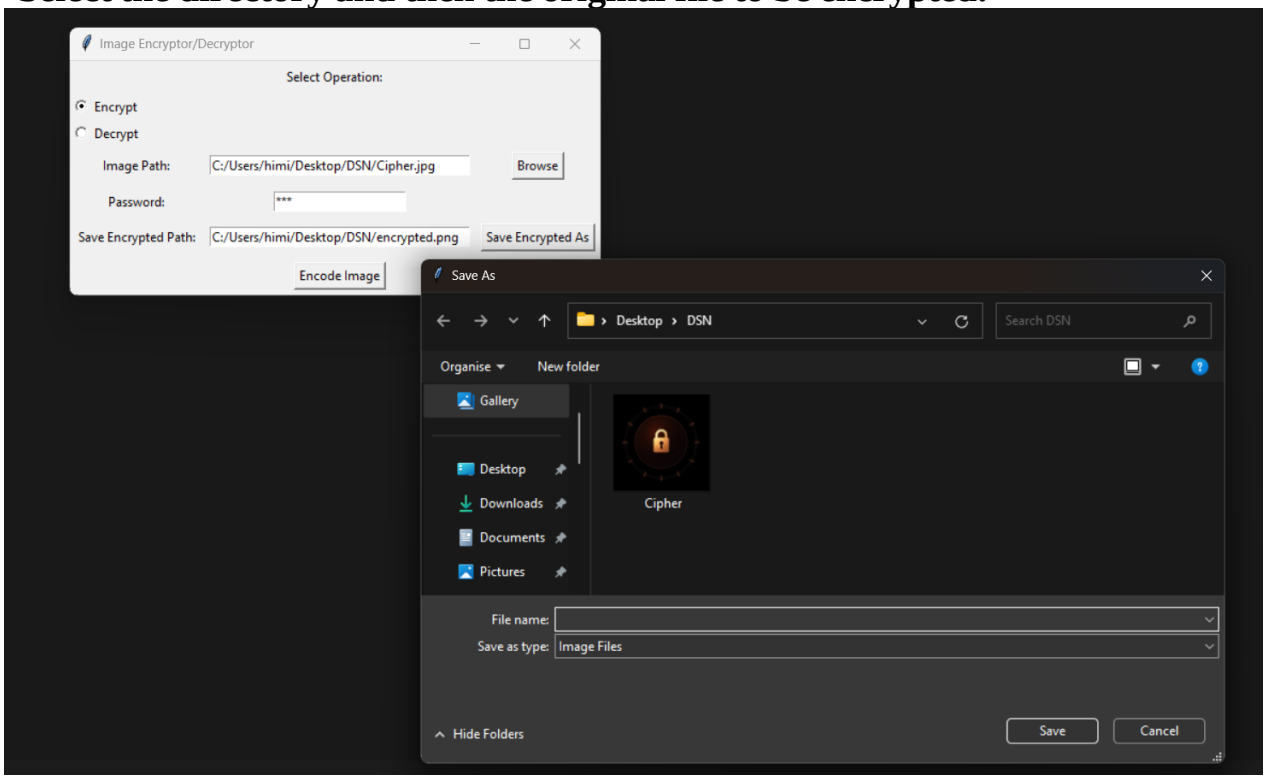
Key is entered:



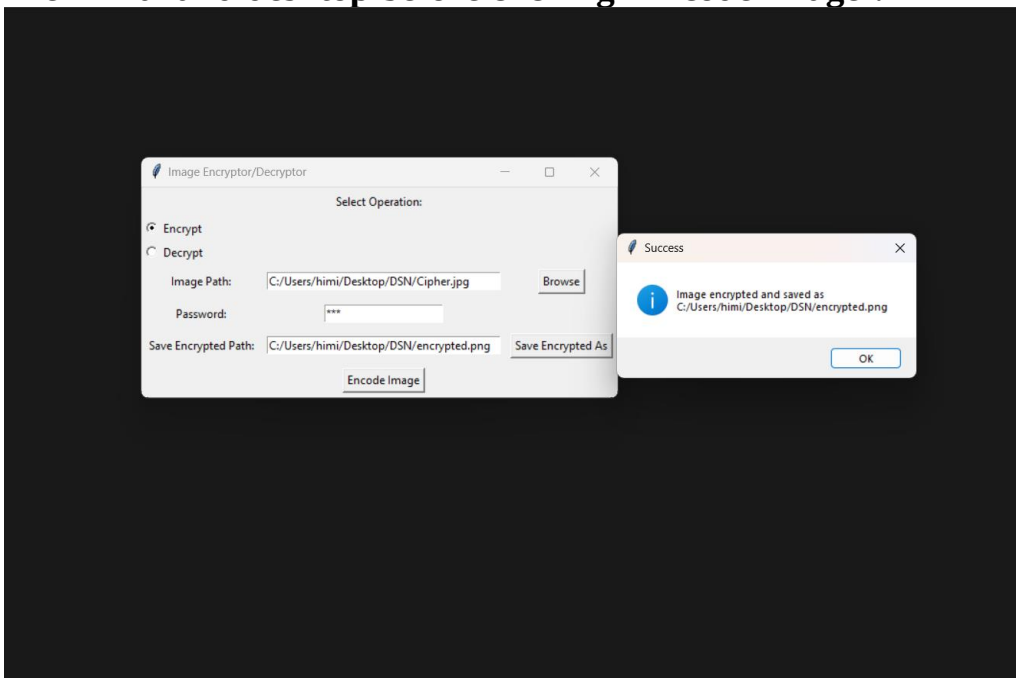
After clicking Save Encrypted As button , Choose the Format:



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

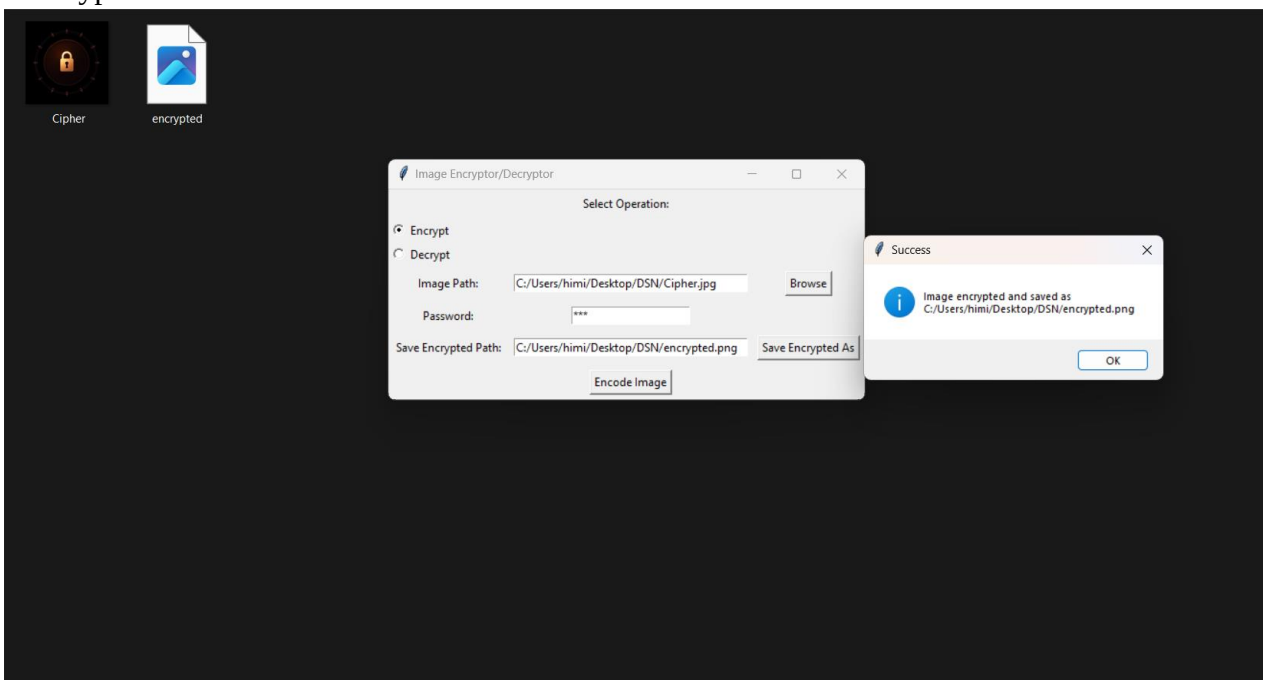


Terminal and desktop before clicking ‘Encode Image’:

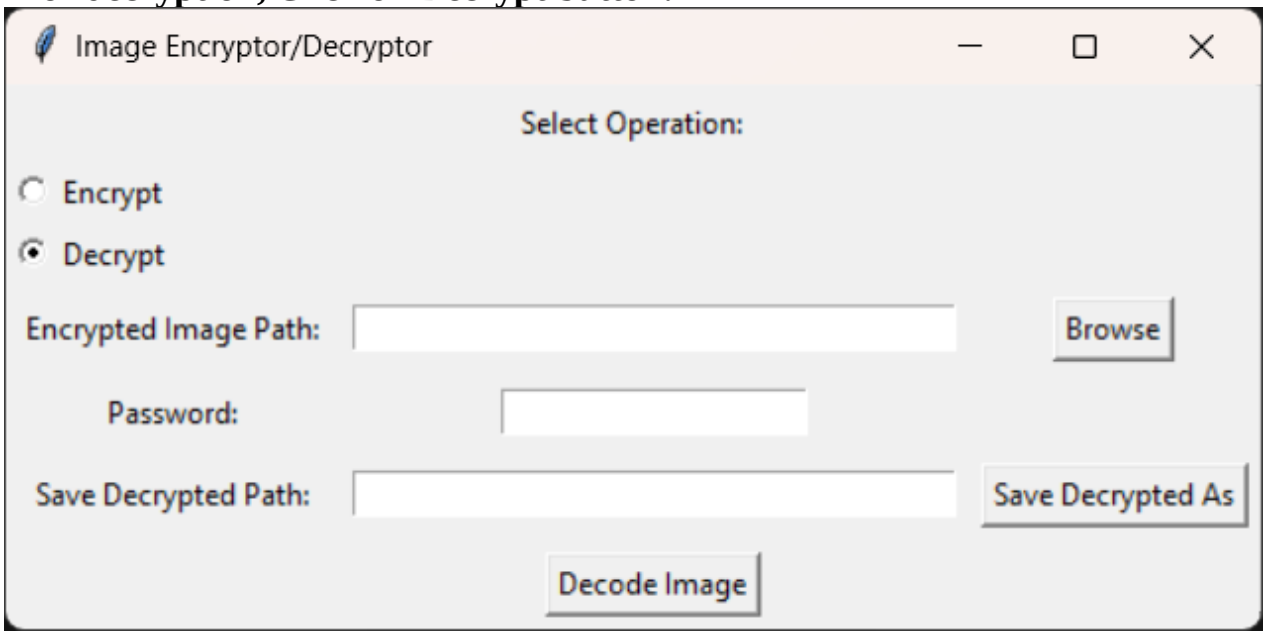


Desktop screen after clicking ‘Encode Image’:

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

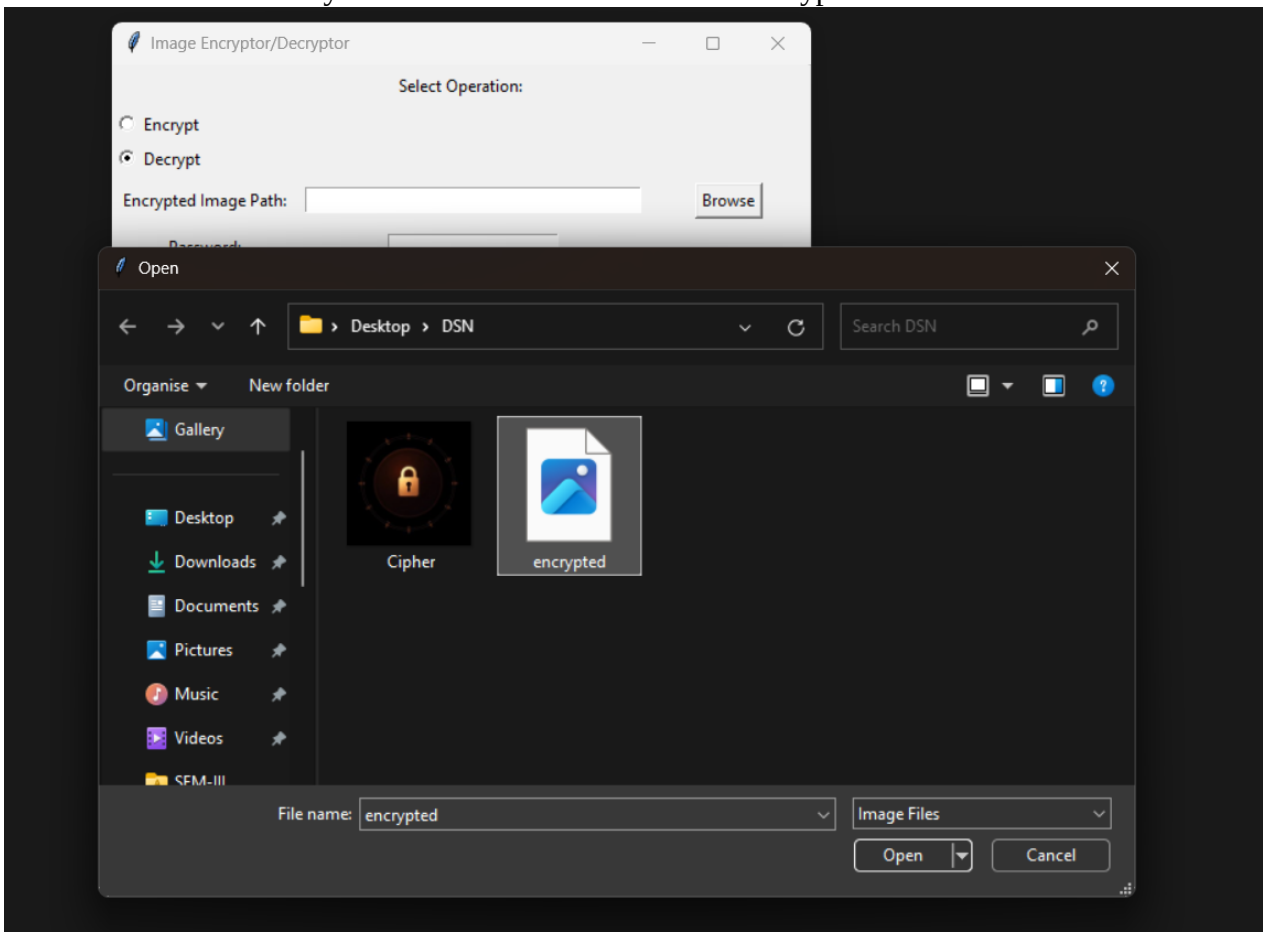


For decryption, Click on Decrypt button:

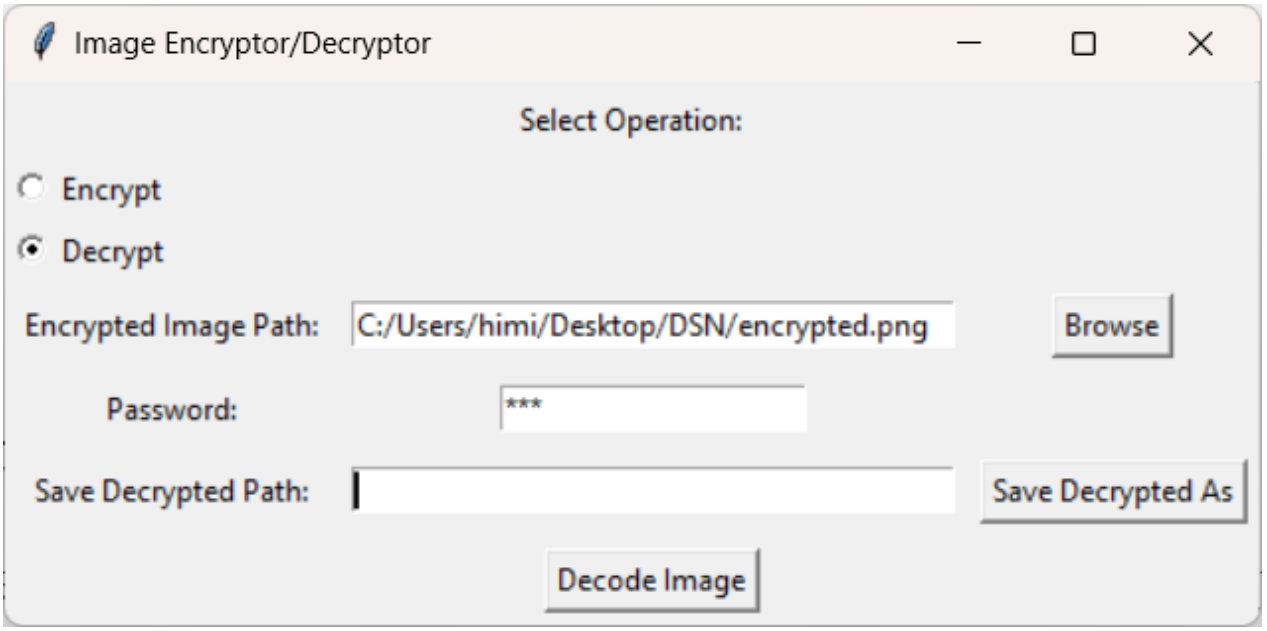


After clicking decryption button:

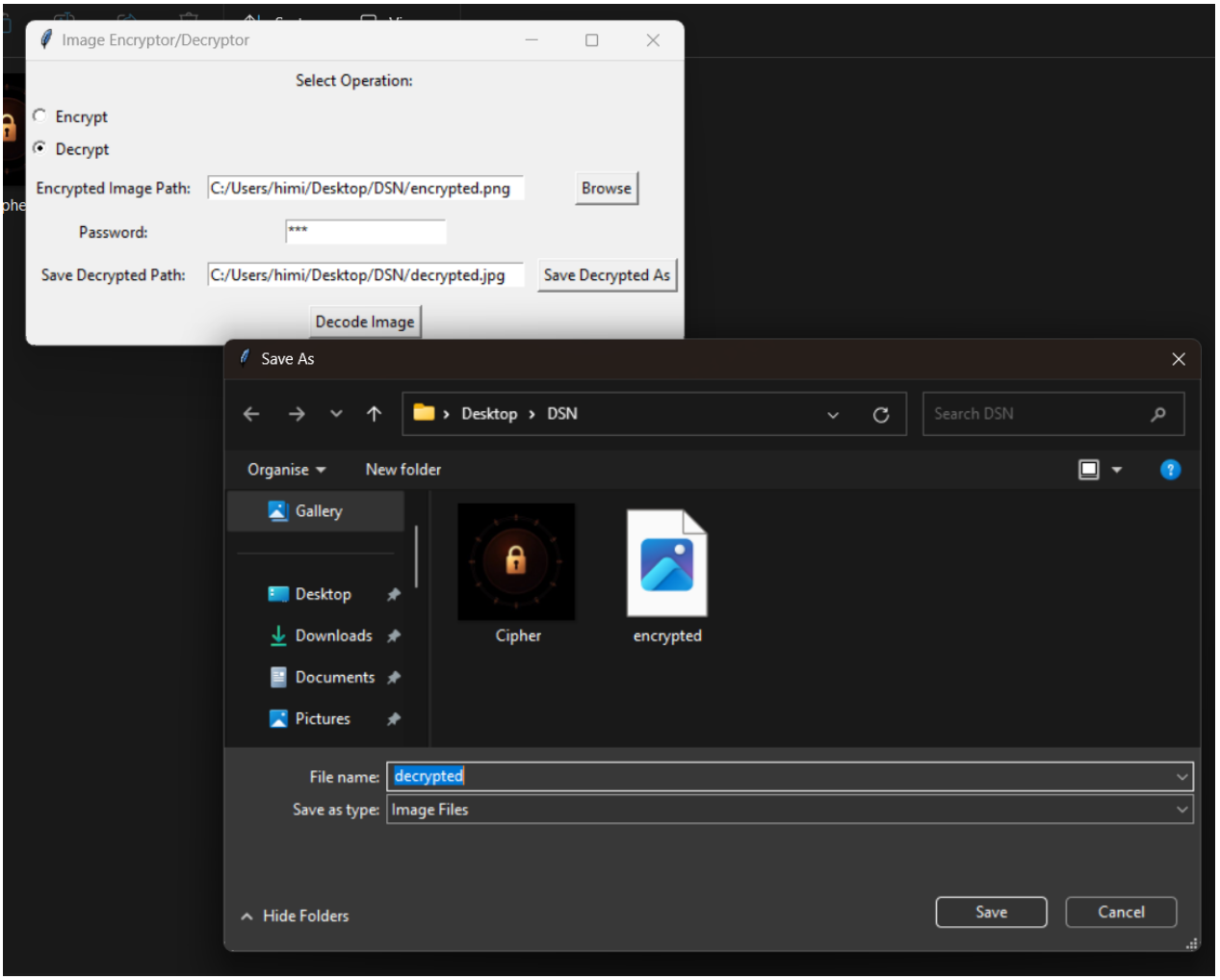
Select the same directory and then select the file named ‘encrypted’.



Enter the Key:

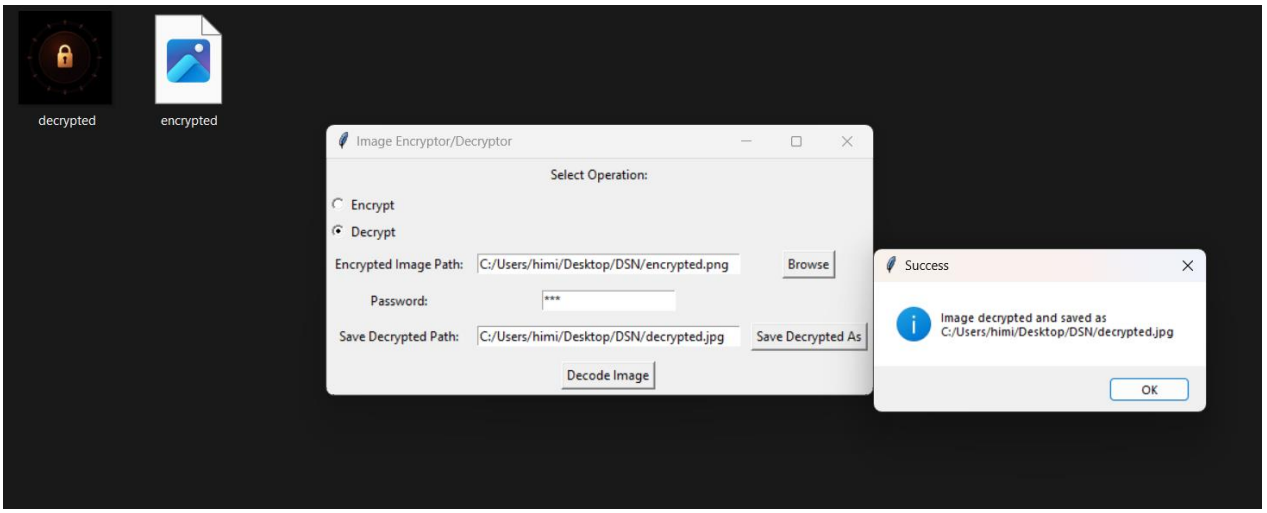


Select the directory and then the encrypted image to be the decrypted image:

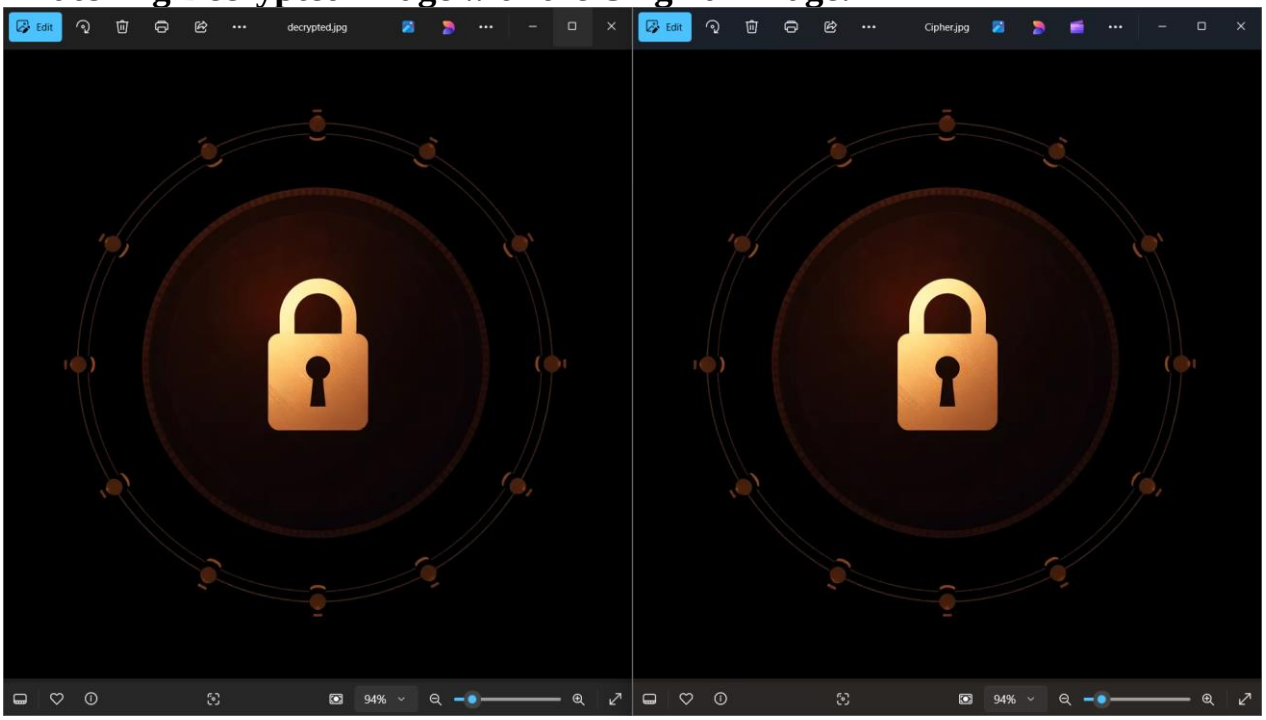


Desktop screen after clicking ‘Decode Image’:

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



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 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.