INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

Dharwad
ज्ञानेन विकासः

Cryptography and information security

D Hemanth Kumar	22BCS034
G Akash	22BCS042
G Sandeep	22BCS043
M Prabhu kumar	22BCS070

ImgVault

Image Encryption and Decryption Tool, is designed to securely encrypt and decrypt images using a method based on the XOR operation. The project employs Cipher Block Chaining (CBC) mode, where each pixel's color value is XORed with both a key and an initial value (IV) to enhance security. The IV is updated dynamically as the encryption progresses, adding an extra layer of complexity

Usage:

1.ENCRYPTING THE IMAGE:

```
# try block to handle exception
try:
    # take path of image as a input
    path = input(r'Enter path of Image : ')

    # taking encryption key as input
    key = int(input('Enter Key for encryption of Image : '))
```

```
    # print path of image file and encryption key that
    # we are using
    print('The path of file : ', path)
    print('Key for encryption : ', key)

    # open file for reading purpose
    fin = open(path, 'rb')

    # storing image data in variable "image"
    image = fin.read()
    fin.close()

    # converting image into byte array to
    # perform encryption easily on numeric data
    image = bytearray(image)

    # performing XOR operation on each value of bytearray
    for index, values in enumerate(image):
        image[index] = values ^ key

    # opening file for writing purpose
    fin = open(path, 'wb')

    # writing encrypted data in image
    fin.write(image)
    fin.close()
    print('Encryption Done...')


except Exception:
    print('Error caught : ', Exception.__name__)
```

2.DECRYPTING THE IMAGE:

```
# try block to handle the exception
try:
    # take path of image as a input
    path = input(r'Enter path of Image : ')
```

```
    # taking decryption key as input
    key = int(input('Enter Key for encryption of Image : '))

    # print path of image file and decryption key that we are using
    print('The path of file : ', path)
    print('Note : Encryption key and Decryption key must be same.')
    print('Key for Decryption : ', key)

    # open file for reading purpose
    fin = open(path, 'rb')

    # storing image data in variable "image"
    image = fin.read()
    fin.close()

    # converting image into byte array to perform decryption easily on
numeric data
    image = bytearray(image)

    # performing XOR operation on each value of bytearray
    for index, values in enumerate(image):
        image[index] = values ^ key

    # opening file for writing purpose
    fin = open(path, 'wb')

    # writing decryption data in image
    fin.write(image)
    fin.close()
    print('Decryption Done...')


except Exception:
    print('Error caught : ', Exception.__name__)
```

3.Graphical user interface(GUI):

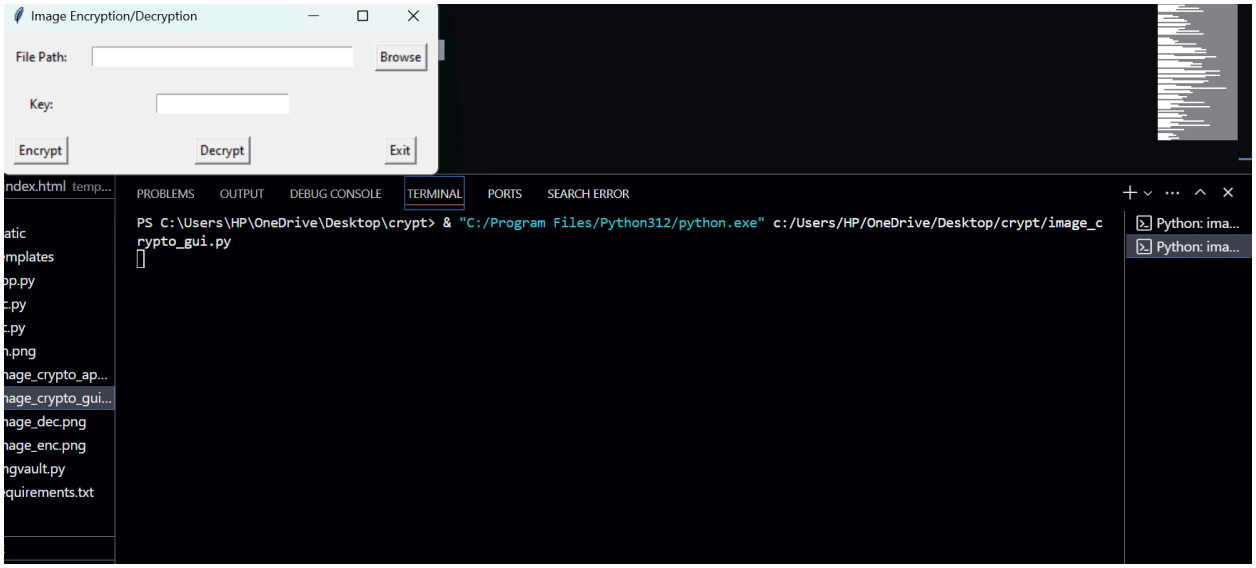
```
import os
```

```python
from tkinter import Tk, Label, Button, Entry, filedialog, messagebox


# Encryption function
def encrypt_image(path, key):
    try:
        with open(path, 'rb') as fin:
            image = bytearray(fin.read())

        for index, values in enumerate(image):
            image[index] = values ^ key

        with open(path, 'wb') as fout:
            fout.write(image)
        messagebox.showinfo("Success", "Encryption Done!")
    except Exception as e:
        messagebox.showerror("Error", f"Encryption failed: {e}")


# Decryption function
def decrypt_image(path, key):
    try:
        with open(path, 'rb') as fin:
            image = bytearray(fin.read())

        for index, values in enumerate(image):
            image[index] = values ^ key

        with open(path, 'wb') as fout:
            fout.write(image)
        messagebox.showinfo("Success", "Decryption Done!")
    except Exception as e:
        messagebox.showerror("Error", f"Decryption failed: {e}")


# GUI Application
class ImageCryptoApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Image Encryption/Decryption")
```

```
        # File path label and entry
        self.file_path_label = Label(root, text="File Path:")
        self.file_path_label.grid(row=0, column=0, padx=10, pady=10)
        self.file_path_entry = Entry(root, width=40)
        self.file_path_entry.grid(row=0, column=1, padx=10, pady=10)

        # Browse button
        self.browse_button = Button(root, text="Browse",
command=self.browse_file)
        self.browse_button.grid(row=0, column=2, padx=10, pady=10)

        # Key label and entry
        self.key_label = Label(root, text="Key:")
        self.key_label.grid(row=1, column=0, padx=10, pady=10)
        self.key_entry = Entry(root, width=20)
        self.key_entry.grid(row=1, column=1, padx=10, pady=10)

        # Encrypt and Decrypt buttons
        self.encrypt_button = Button(root, text="Encrypt",
command=self.encrypt_action)
        self.encrypt_button.grid(row=2, column=0, padx=10, pady=10)

        self.decrypt_button = Button(root, text="Decrypt",
command=self.decrypt_action)
        self.decrypt_button.grid(row=2, column=1, padx=10, pady=10)

        # Exit button
        self.exit_button = Button(root, text="Exit", command=root.quit)
        self.exit_button.grid(row=2, column=2, padx=10, pady=10)

    def browse_file(self):
        file_path = filedialog.askopenfilename(
            filetypes=[("Image Files", "*.png;*.jpg;*.jpeg;*.bmp;*.tiff"),
("All Files", "*.*")]
        )
        self.file_path_entry.delete(0, 'end')
        self.file_path_entry.insert(0, file_path)

    def encrypt_action(self):
```

```
        path = self.file_path_entry.get()
        key = self.key_entry.get()

        if not os.path.exists(path):
            messagebox.showerror("Error", "File does not exist!")
            return
        if not key.isdigit():
            messagebox.showerror("Error", "Key must be a numeric value!")
            return

        encrypt_image(path, int(key))

    def decrypt_action(self):
        path = self.file_path_entry.get()
        key = self.key_entry.get()

        if not os.path.exists(path):
            messagebox.showerror("Error", "File does not exist!")
            return
        if not key.isdigit():
            messagebox.showerror("Error", "Key must be a numeric value!")
            return

        decrypt_image(path, int(key))


# Run the application
if __name__ == "__main__":
    root = Tk()
    app = ImageCryptoApp(root)
    root.mainloop()
```

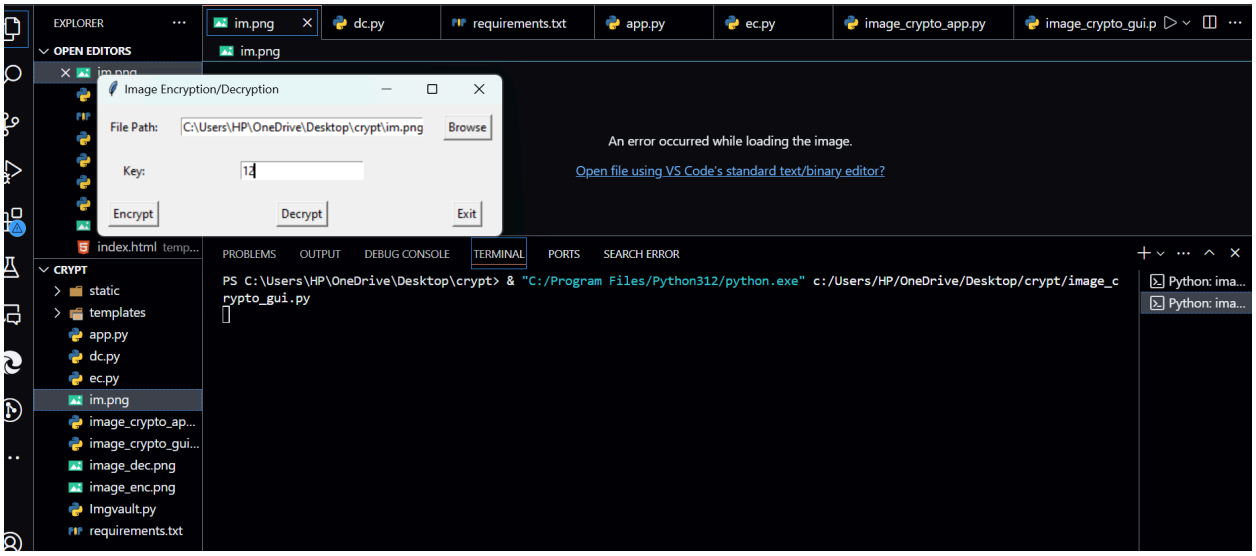
RUN THE CODE IN THE TERMINAL:



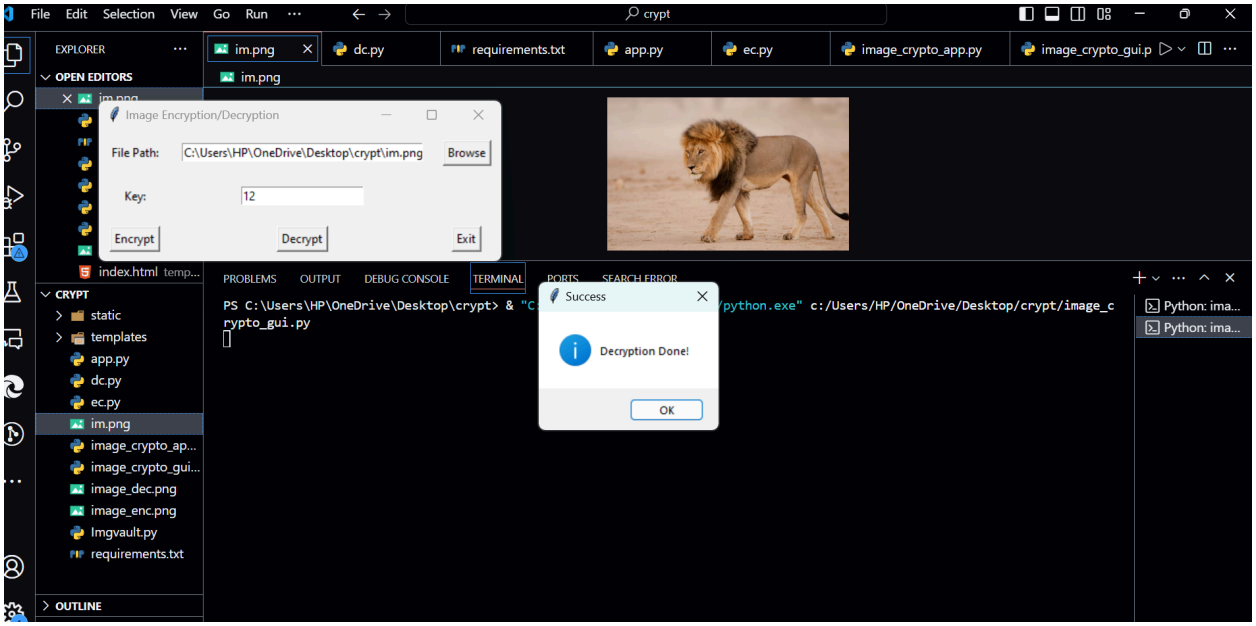
FILE PATH : Give the file path of the respective image
After giving the key it will encrypt the respective image.
Original image:

Encrypted image:

After encrypting the image:



DECRYPTED IMAGE:
After decrypting the image it will display the prompt that decryption was done.


File Edit Selection View Go Run
crypt
EXPLORER
OPEN EDITORS
im.png
dc.py
requirements.txt
app.py
ec.py
image_crypto_app.py
image_crypto_gui.p
Image Encryption/Decryption
File Path:
C:\Users\HP\OneDrive\Desktop\crypt\im.png
Browse
Key:
12
Encrypt
Decrypt
Exit
index.html
CRYPT
static
templates
app.py
dc.py
ec.py
im.png
image_crypto_ap...
image_crypto_gui...
image_dec.png
image_enc.png
Imgvault.py
requirements.txt
OUTLINE
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
SEARCH ERROR
PS C:\Users\HP\OneDrive\Desktop\crypt> & "C
python.exe" c:/Users/HP/OneDrive/Desktop/crypt/image_crypto_gui.py
Success
Decryption Done!
OK
Python: ima...
Python: ima...