Prodigy Infotech Cybersecurity Internship Report

Task 3: Password Complexity Checker

Submitted by: Saira Arshad

STUDENT ID: CA/AU1/8075

Date: August 2025

Duration : 1 month

Table of Contents

Introduction

This project is part of the Prodigy Infotech internship program. The objective of Task 02 is to create a simple image encryption tool using pixel manipulation. The tool encrypts and decrypts images using a mathematical operation (XOR) on the pixel values, making it possible to secure image data with a secret key.

Objective

The main objectives of this project are:
1. To develop a Python program for encrypting and decrypting images.
2. To implement pixel manipulation using the XOR operation.
3. To create a user-friendly interface using Tkinter for image selection and processing.

How it Works (Pixel XOR Method)

The XOR (Exclusive OR) operation is applied to the RGB pixel values of an image using a secret key (an integer between 0 and 255). When XOR is applied with the same key twice, the original value is restored. This property is used for both encryption and decryption:

- Encryption: Original pixel value XOR Key → Encrypted pixel
- Decryption: Encrypted pixel XOR Key → Original pixel

Full Python Code

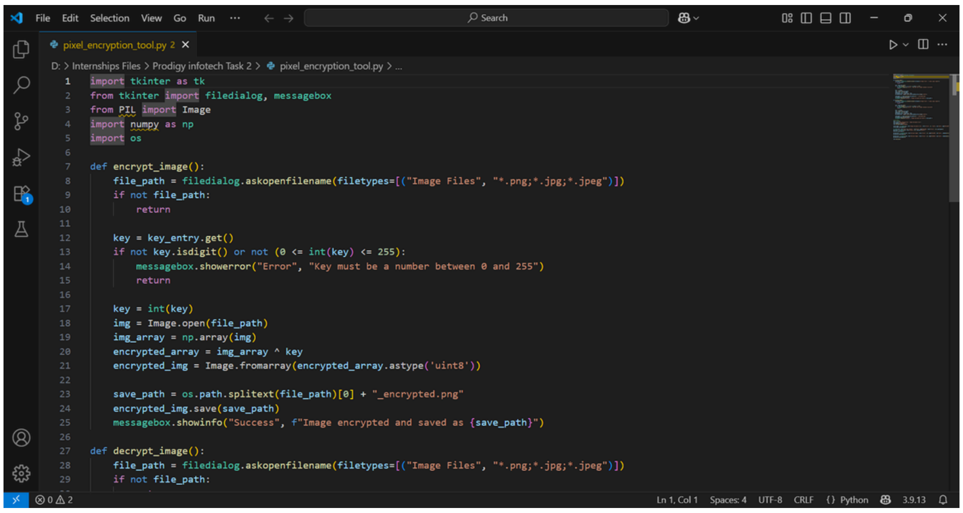
```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image
import numpy as np
import os

def encrypt_image():
 file_path = filedialog.askopenfilename(filetypes=[("Image Files", "*.png;*.jpg;*.jpeg")])
 if not file_path:
 return

 key = key_entry.get()
 if not key.isdigit() or not (0 <= int(key) <= 255):
 messagebox.showerror("Error", "Key must be a number between 0 and 255")
 return

 key = int(key)
 img = Image.open(file_path)
 img_array = np.array(img)
 encrypted_array = img_array ^ key
 encrypted_img = Image.fromarray(encrypted_array.astype('uint8'))
 save_path = os.path.splitext(file_path)[0] + "_encrypted.png"
 encrypted_img.save(save_path)
 messagebox.showinfo("Success", f"Image encrypted and saved as {save_path}")

def decrypt_image():
 file_path = filedialog.askopenfilename(filetypes=[("Image Files", "*.png;*.jpg;*.jpeg")])
 if not file_path:
 return

 key = key_entry.get()
 if not key.isdigit() or not (0 <= int(key) <= 255):
 messagebox.showerror("Error", "Key must be a number between 0 and 255")
 return

 key = int(key)
 img = Image.open(file_path)
 img_array = np.array(img)
 decrypted_array = img_array ^ key
 decrypted_img = Image.fromarray(decrypted_array.astype('uint8'))

 save_path = os.path.splitext(file_path)[0] + "_decrypted.png"
 decrypted_img.save(save_path)
 messagebox.showinfo("Success", f"Image decrypted and saved as {save_path}")

root = tk.Tk()
root.title("Pixel Manipulation - Image Encryption Tool")
root.geometry("400x250")
root.configure(bg="#2c3e50")

title_label = tk.Label(root, text="Image Encryption Tool", font=("Arial", 16, "bold"), fg="white", bg="#2c3e50")
title_label.pack(pady=10)

tk.Label(root, text="Enter Key (0-255):", fg="white", bg="#2c3e50", font=("Arial", 12)).pack(pady=5)
key_entry = tk.Entry(root, font=("Arial", 12), justify="center")
key_entry.pack(pady=5)

encrypt_btn = tk.Button(root, text="Encrypt Image", font=("Arial", 12), bg="#27ae60", fg="white",
command=encrypt_image)
encrypt_btn.pack(pady=10)

decrypt_btn = tk.Button(root, text="Decrypt Image", font=("Arial", 12), bg="#c0392b", fg="white",
command=decrypt_image)
decrypt_btn.pack(pady=10)
```

root.mainloop()

Step-by-Step Usage

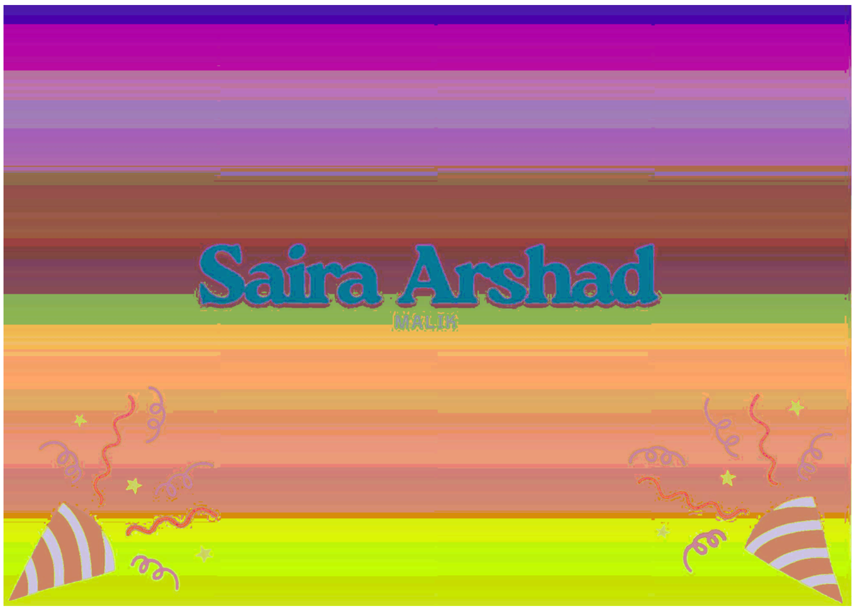
1. Run the Python file (pixel_encryption_tool.py).
2. Enter a secret key (0–255) in the input box.
3. Click 'Encrypt Image' and select the original image.
4. The encrypted image will be saved with '_encrypted' in its filename.
5. Click 'Decrypt Image' and select the encrypted image.
6. The decrypted image will be saved with '_decrypted' in its filename.

Program Running


File Edit Selection View Go Run ···
Search
pixel_encryption_tool.py 2
D: > Internships Files > Prodigy infotech Task 2 > pixel_encryption_tool.py > ...
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.9.13

```python
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image
import numpy as np
import os

def encrypt_image():
    file_path = filedialog.askopenfilename(filetypes=[("Image Files", "*.png;*.jpg;*.jpeg")])
    if not file_path:
        return

    key = key_entry.get()
    if not key.isdigit() or not (0 <= int(key) <= 255):
        messagebox.showerror("Error", "Key must be a number between 0 and 255")
        return

    key = int(key)
    img = Image.open(file_path)
    img_array = np.array(img)
    encrypted_array = img_array ^ key
    encrypted_img = Image.fromarray(encrypted_array.astype('uint8'))

    save_path = os.path.splitext(file_path)[0] + "_encrypted.png"
    encrypted_img.save(save_path)
    messagebox.showinfo("Success", f"Image encrypted and saved as {save_path}")

def decrypt_image():
    file_path = filedialog.askopenfilename(filetypes=[("Image Files", "*.png;*.jpg;*.jpeg")])
    if not file_path:
```

Encrypted Image



Decrypted Image

Conclusion

This project successfully demonstrates how pixel manipulation using the XOR operation can be used for image encryption and decryption. The method is simple, effective, and reversible when the same key is used. With the added Tkinter GUI, the program becomes user-friendly and suitable for real-world basic encryption tasks.