**IMAGE ENCRYPTION TOOL USING PIXEL MANIPULATION**

# INTRODUCTION

This project presents a simple yet functional image encryption and decryption tool built using Python. The tool leverages the Tkinter library for the graphical user interface (GUI) and Pillow (PIL) for image processing. It demonstrates how fundamental pixel manipulation techniques can be applied to secure image files.

# WHAT IS PIXEL MANIPULATION?

Pixel manipulation involves changing the individual pixel values of an image. Pixels are the smallest units of an image, and modifying their color values (typically represented as RGB – Red, Green, Blue) alters the visual representation of the image. This concept is crucial in various domains such as image processing, steganography, cryptography, and computer vision.

# HOW THE TOOL WORKS

The image encryption tool operates by:  
1. Applying XOR and basic mathematical operations (addition/subtraction) to each pixel's RGB values.  
2. Swapping adjacent pixels in each row to add an additional layer of complexity.  
3. Reversing the above steps during decryption to restore the original image.

# TECHNOLOGIES USED

Python: Primary programming language.  
 Tkinter: Used to develop the GUI.  
 Pillow (PIL): Used for image loading, reading pixel values, and saving.  
 PyInstaller (optional): For converting the Python script into a standalone executable file.

# KEY FEATURES

Select an image file (PNG, JPG).  
Enter a numeric encryption key.  
Encrypt and decrypt images using reversible operations.  
Simple and lightweight GUI with no external dependencies apart from Pillow.

# CORE ENCRYPTION LOGIC (CODE SNIPPET)

r = (r + delta \* 30) % 256 ^ (key % 256)  
g = (g + delta \* 20) % 256 ^ (key \* 2 % 256)  
b = (b + delta \* 10) % 256 ^ (key \* 3 % 256)

# CODE

import tkinter as tk

from tkinter import filedialog, messagebox

from PIL import Image

import os

def process\_image(path, key, mode):

    img = Image.open(path)

    pix = img.load()

    w, h = img.size

    if mode == "decrypt":

        for i in range(0, w - 1, 2):

            for j in range(h):

                pix[i, j], pix[i+1, j] = pix[i+1, j], pix[i, j]

    for i in range(w):

        for j in range(h):

            r, g, b = pix[i, j]

            delta = -1 if mode == "decrypt" else 1

            r = (r + delta \* 30) % 256 ^ (key % 256)

            g = (g + delta \* 20) % 256 ^ (key \* 2 % 256)

            b = (b + delta \* 10) % 256 ^ (key \* 3 % 256)

            pix[i, j] = (r, g, b)

    if mode == "encrypt":

        for i in range(0, w - 1, 2):

            for j in range(h):

                pix[i, j], pix[i+1, j] = pix[i+1, j], pix[i, j]

    out = os.path.splitext(path)[0] + f"\_{mode}.png"

    img.save(out)

    return out

def select\_file(): entry.delete(0, tk.END) or entry.insert(0, filedialog.askopenfilename(filetypes=[("Image", "\*.png;\*.jpg")]))

def run(mode):

    path, key = entry.get(), entry\_key.get()

    if not path or not key.isdigit():

        return messagebox.showwarning("Missing Info", "Select image and enter numeric key")

    try:

        out = process\_image(path, int(key), mode)

        messagebox.showinfo("Done", f"{mode.title()}ed image saved:\n{out}")

    except Exception as e:

        messagebox.showerror("Error", str(e))

root = tk.Tk()

root.title("Image Encryptor")

root.geometry("400x220")

tk.Label(root, text="Select Image:").pack(pady=5)

entry = tk.Entry(root, width=50); entry.pack()

tk.Button(root, text="Browse", command=select\_file).pack(pady=5)

tk.Label(root, text="Enter Key:").pack()

entry\_key = tk.Entry(root); entry\_key.pack(pady=5)

tk.Button(root, text="Encrypt", width=15, command=lambda: run("encrypt")).pack(pady=5)

tk.Button(root, text="Decrypt", width=15, command=lambda: run("decrypt")).pack(pady=5)

root.mainloop()

# FUTURE SCOPE

- Add support for image preview within the GUI.  
- Implement drag-and-drop functionality.  
- Support batch encryption of multiple images.  
- Add secure key generation and password protection.  
- Expand to video or audio file encryption using similar principles.

# CONCLUSION

This project provides a foundational understanding of how simple image encryption can be achieved using pixel manipulation techniques. It combines the power of GUI-based Python applications with core concepts from image processing and cryptography, making it an ideal beginner-friendly project with real-world relevance.