




OCTOBER 17, 2024

# **IMAGE ENCRYPTION AND DECRYPTION TOOL USING PIXEL MANIPULATION**

**"A SECURE METHOD FOR IMAGE OBFUSCATION USING SIMPLE PIXEL-BASED TECHNIQUES"**

JAGADISH TRIPATHY

Jagadishtripathy144@gmail.com



## Table of Contents

1. Introduction .....	2
2. Objectives.....	2
3. Technical Approach .....	2
3.1 Tools and Technologies .....	2
3.2 Encryption Method .....	2
3.3 Decryption Method.....	3
4. Implementation.....	3
4.1 Encryption Process.....	3
4.2 Decryption Process .....	3
4.3 User Input .....	4
5. Usage.....	4
5.1 Prerequisites .....	4
5.2 How to Use the Tool.....	4
6. Testing and Validation .....	4
7. Future Improvements .....	5
8. Conclusion.....	5

# Image Encryption Tool Using Pixel Manipulation

## 1. Introduction

This report details the development and functionality of an image encryption tool using pixel manipulation techniques. The tool is designed to encrypt and decrypt images, rendering the encrypted image blank until the correct decryption process is applied. This ensures visual obfuscation, making the original content inaccessible without proper decryption.

The tool is built using Python and the Pillow library and allows users to securely encrypt and decrypt images. This solution demonstrates simple image encryption through pixel manipulation, which can be further enhanced or adapted for more advanced cryptographic applications.

## 2. Objectives

The main objectives of this project are:

- To implement a simple image encryption method using pixel manipulation.
- To visually obscure the image during encryption by setting all pixel values to black.
- To provide users with an intuitive interface to encrypt and decrypt images.
- To ensure that the encrypted image remains completely unviewable until decrypted.

## 3. Technical Approach

### 3.1 Tools and Technologies

- **Programming Language:** Python
- **Library:** Pillow (Python Imaging Library)
- **Platform:** Any platform with Python and Pillow support, such as Linux (Kali), Windows, or macOS.

### 3.2 Encryption Method

The encryption algorithm used in this project works by setting all pixel values of an image to black (RGB values (0, 0, 0)), making the image visually unreadable. This simple method visually encrypts the image without using any complex cryptographic techniques, ensuring that the image cannot be viewed until the correct decryption process is applied.

### 3.3 Decryption Method

During decryption, the original pixel values of the image are restored to their initial state, allowing the image to be viewed normally.

## 4. Implementation

### 4.1 Encryption Process

The encryption process traverses through each pixel of the image and sets its RGB value to (0, 0, 0), making it fully black. Below is an example of the code used for encryption:

```
def encrypt_image(input_path, output_path):  
    img = Image.open(input_path)  
    pixels = img.load()  
    width, height = img.size  
  
    for x in range(width):  
        for y in range(height):  
            pixels[x, y] = (0, 0, 0) # Set to black  
  
    img.save(output_path)  
    print(f"Image encrypted and saved as {output_path}.")
```

### 4.2 Decryption Process

The decryption process restores the pixel values of the image to their original state, revealing the image content. The code for decryption is:

```
def decrypt_image(original_path, encrypted_path, output_path):  
    original_img = Image.open(original_path)  
    encrypted_img = Image.open(encrypted_path)  
    original_pixels = original_img.load()  
    encrypted_pixels = encrypted_img.load()  
  
    width, height = encrypted_img.size  
  
    for x in range(width):  
        for y in range(height):  
            encrypted_pixels[x, y] = original_pixels[x, y]  
  
    encrypted_img.save(output_path)  
    print(f"Image decrypted and saved as {output_path}.")
```

## 4.3 User Input

The user is prompted to provide the paths for the original image, encrypted image, and decrypted image:

```
def main():  
    original_image_path = input("Enter the path of the original image to encrypt: ")  
    encrypted_output_path = input("Enter the output path for the encrypted image: ")  
    decrypted_output_path = input("Enter the output path for the decrypted image: ")  
  
    encrypt_image(original_image_path, encrypted_output_path)  
    decrypt_image(original_image_path, encrypted_output_path, decrypted_output_path)
```

## 5. Usage

### 5.1 Prerequisites

- **Python 3.x** must be installed
- **Pillow library** needs to be installed, which can be done using:

```
pip3 install pillow
```

### 5.2 How to Use the Tool

1. **Run the Python script** on your system.
2. The script will ask you to input:
  - The path of the original image.
  - The path where the encrypted image should be saved.
  - The path where the decrypted image should be saved.
3. Once the process is completed, the image will be encrypted (all-black image), and the decryption process will restore the original image.

## 6. Testing and Validation

1. **Test Case 1:** Encrypt a sample image.
  - Input: Image file sample\_image.jpg
  - Output: Encrypted image, which appears completely black.
2. **Test Case 2:** Decrypt the encrypted image.
  - Input: Encrypted image and original image file.
  - Output: The original image is restored to its original state.

The tool was successfully tested with several images of varying sizes and formats (e.g., PNG, JPG), and it accurately encrypted and decrypted images in each case.

## 7. Future Improvements

- **Advanced Encryption:** Implement more advanced encryption algorithms, such as AES or XOR encryption, to provide stronger security.
- **Password Protection:** Add a feature where the user needs to enter a password or key to decrypt the image.
- **Support for Other Image Formats:** Extend the tool to support additional image formats (e.g., BMP, TIFF).
- **Graphical User Interface (GUI):** Create a user-friendly GUI to make the tool more accessible for non-technical users.

## 8. Conclusion

This project demonstrates the use of pixel manipulation for simple image encryption. While the encryption method used here is basic, it serves as a foundation for further enhancements in security and usability. The tool successfully encrypts images by visually obscuring their content, making them unreadable until decrypted.