Name
Ashish Ramling Patil
(PGDCS414)

Project Name – SecureStego

Image-Based Text Encryption using Steganography
Steganography with Automatic Key Generation

Introduction:

This document provides a step-by-step guide to performing steganography using Python, where secret text is hidden within an image. The encryption process generates a key automatically, which is required for decryption.

🔑 Understanding the Automatically Generated Key

Your steganography project uses **Fernet encryption**, which automatically generates a **secure key** when encrypting the message. Let's break it down:

What is the Generated Key?

- The key is a **randomly generated 32-byte (256-bit) secret key**.
- It is used for both **encryption** and **decryption**.
- It follows the **Fernet standard**, which is based on **AES-128 encryption (CBC mode) with HMAC authentication**.

Step 1: Install Required Libraries:

Before running the script, ensure you have the necessary Python libraries installed. Use the following command:

```
pip install cryptography pillow numpy opencv-python
```

Step 2: Encryption (Hiding Text in Image)

Python Script: encryption.py

```
from cryptography.fernet import Fernet
from PIL import Image
import numpy as np
import cv2

# Generate and save a key
```

```
def generate_key():
    key = Fernet.generate_key()
    with open("key.key", "wb") as key_file:
        key_file.write(key)
    return key

# Encrypt a message using the key
def encrypt_message(message, key):
    cipher = Fernet(key)
    return cipher.encrypt(message.encode()).decode()

# Hide encrypted text inside an image
def hide_text_in_image(image_path, text, output_path):
    img = cv2.imread(image_path)
    flat_img = img.flatten()
    binary_text = ''.join(format(ord(char), '08b') for char in text)

    if len(binary_text) > len(flat_img):
        raise ValueError("Text is too long to hide in this image")

    for i in range(len(binary_text)):
        flat_img[i] = (flat_img[i] & ~1) | int(binary_text[i])

    img_stego = flat_img.reshape(img.shape)
    cv2.imwrite(output_path, img_stego)

# Main function
def main():
    print("\n🔐 IMAGE STEGANOGRAPHY - ENCRYPTION 🔐\n")
    image_path = input("🖼 Enter the path of the image: ")
    secret_text = input("✍ Enter the secret text to hide: ")
    key = generate_key()
    print("\n🔑 Key generated and saved as 'key.key'. Keep it safe!\n")

    encrypted_message = encrypt_message(secret_text, key)
    output_image = "stego_image.png"
    hide_text_in_image(image_path, encrypted_message, output_image)
    print(f"✅ Encrypted message hidden in {output_image}\n")

if __name__ == "__main__":
    main()
```

Step 3: Decryption (Extracting Hidden Text)

Python Script: decryption.py

```
from cryptography.fernet import Fernet
from PIL import Image
import numpy as np
import cv2

# Load the encryption key
def load_key():
    key_path = input("🔑 Enter the path of the key file: ")
    with open(key_path, "rb") as key_file:
        return key_file.read()

# Decrypt the hidden message

def decrypt_message(encrypted_message, key):
    cipher = Fernet(key)
    return cipher.decrypt(encrypted_message.encode()).decode()

# Extract hidden text from an image
def extract_text_from_image(stego_image_path, key):
    img = cv2.imread(stego_image_path)
    flat_img = img.flatten()

    binary_text = ''
    for i in range(0, len(flat_img)):
        binary_text += str(flat_img[i] & 1)
        if len(binary_text) % 8 == 0 and binary_text[-8:] == "00000000":
            break

    encrypted_message = ''.join(chr(int(binary_text[i:i+8], 2)) for i in range(0,
len(binary_text), 8))
    decrypted_text = decrypt_message(encrypted_message, key)
    print(f"🔒 Decrypted Message: {decrypted_text}\n")

# Main function
def main():
    print("\n🔍 IMAGE STEGANOGRAPHY - DECRYPTION 🔍\n")
    stego_image_path = input("🖼 Enter the path of the stego image: ")
    key = load_key()
    extract_text_from_image(stego_image_path, key)
```

```
if __name__ == "__main__":
    main()
```

Step 4: Running the Scripts

To encrypt a message and hide it in an image, run:

```
python encryption.py
```

To decrypt and extract the hidden message, run:

```
python decryption.py
```

Conclusion

This guide explains how to perform image steganography using Python. The encryption process securely hides the text within an image, and the decryption process extracts the hidden message using the generated key. Ensure that the `key.key` file is kept secure, as it is required for decryption.