STENOGRAPHY PROJECT REPORT

E Barkavi

1. Introduction

Overview of Steganography and Its Applications

Steganography is the practice of hiding secret information within digital media, such as images, audio, or video files. Unlike cryptography, which encrypts the message itself, steganography conceals the existence of the message. This makes it an effective tool for secure communication.

Applications of Steganography:

Data Security: Used for covert communication to prevent unauthorized access.

Digital Watermarking: Protects copyrights by embedding hidden data within media files.

Forensic Investigation: Helps in embedding forensic marks for identifying digital evidence.

Cybersecurity: Used to embed security keys or authentication details within images.

This project implements image-based steganography, where a secret message is encoded into an image and later decrypted using a passcode.

2. Objectives

The primary objectives of this project are:

To implement a basic steganography technique that embeds a text message into an image.

To ensure message security by using a passcode for decryption.

To demonstrate how ASCII values can be stored within pixel data to hide messages.

To develop a simple encryption and decryption system using Python and OpenCV.

To explore possible enhancements, such as encrypting messages before embedding them.

By achieving these goals, this project demonstrates how steganography can be used for secure and hidden communication.

3. Methodology

Encryption Process:

1. Read the Image: Load the image using OpenCV.
2. Convert Message to ASCII: Each character in the message is converted into its ASCII value.
3. Embed into Image Pixels: The ASCII values are stored in the pixel values of the image.
4. Save the Encrypted Image: The modified image is saved as an encrypted file.

Decryption Process:

1. Read the Encrypted Image: Load the modified image.

2. Retrieve ASCII Values: Extract the embedded ASCII values from the pixels.
3. Convert ASCII to Characters: Convert the retrieved values back to text.
4. Verify Passcode: Decryption is only allowed if the correct passcode is provided.

This ensures that the message remains hidden unless the correct passcode is entered.

4. Implementation

Code Breakdown

1. Importing Libraries and Loading the Image

import cv2 import os cv2 (OpenCV) is used

for image processing.

os is used to open the encrypted image after saving.

2. Encoding the Message into the Image

```
img = cv2.imread("raaki.jpg")  # Load the image
msg = input("Enter secret message:")  password
= input("Enter a passcode:")
```

The user provides a secret message and a passcode for security.

3. Creating Character to ASCII Mappings

d = {chr(i): i for i in range(255)} # Character to ASCII mapping c

= {i: chr(i) for i in range(255)} # ASCII to Character mapping

These dictionaries are used for encoding and decoding messages.

4. Hiding the Message in Image Pixels

n, m, z = 0, 0, 0 for i in

range(len(msg)): img[n, m, z] =

d[msg[i]] n, m, z = n + 1, m + 1,

(z + 1) % 3

The ASCII values are embedded into the pixel values of the image.

The (z + 1) % 3 ensures that the values are stored in different color channels (Red, Green, or Blue).

5. Saving and Displaying the Encrypted Image

cv2.imwrite("encryptedImage.jpg", img)

os.system("start encryptedImage.jpg") # Opens the saved image

The modified image is saved and displayed.

6. Decryption Process

```
message = "" n,
m, z = 0, 0, 0
pas = input("Enter passcode for Decryption")


if password == pas:     for
i in range(len(msg)):
        message += c[img[n, m, z]]         n, m,
z = n + 1, m + 1, (z + 1) % 3
print("Decryption message:", message)
else:
   print("YOU ARE NOT auth")
```

The user must enter the correct passcode.

If correct, the message is retrieved from the image and displayed.

5. Results and Conclusion

Results :

The program successfully hides text inside an image and retrieves it when needed.

The use of a passcode ensures that only authorized users can access the hidden message.

The message remains undetectable to anyone viewing the image normally.

Future Improvements:

Better Encoding: Use advanced steganographic techniques for stronger security.

Data Encryption: Encrypt the message before embedding it.

Support for Large Texts: Optimize storage for longer messages.

Graphical User Interface (GUI): Make it user-friendly with a simple interface.

Conclusion :

This project demonstrates how steganography can be used for secure communication by embedding messages into images. While this is a basic implementation, it lays the foundation for more sophisticated techniques in cybersecurity and data protection.