Name: **Richard Olu Jordan**

Class: **Information Security II**

Professor**: Remi Chou**

Date: **12/3/2024**

Report: **Steganography Assignment**

**Table of Contents**

1. Summary

* Purpose of the Assignment
* Key Objectives

1. Approach

* Overview of the Methodology
* Steps for Hiding the Image
* Steps for Extracting the Image

1. Code

* Key Operations for Hiding the Image
* Key Operations for Extracting the Image
* Error Handling and Validation

1. Results

* Test Case 1: Hiding a Secret Image
* Test Case 2: Extracting the Hidden Image
* Observations

**Summary**

Steganography is the practice of **concealing information within non-secret objects** or files to securely transmit sensitive data. This assignment focuses on implementing a steganography technique using Python to hide and retrieve an image. The objective is to **embed a smaller, secret image into a larger**, visible cover image by manipulating specific bits in the cover image's pixel data. The resulting image, known as the stego image, appears visually similar to the cover image but contains encoded hidden data.

The process is achieved through **bitwise operations**, which allow embedding the secret image into the cover image without significantly altering its appearance. The stego image can then be shared, and the secret image can later be extracted from it using a reverse decoding process. This implementation highlights the effectiveness of bitwise operations in encoding and decoding data, offering a clear understanding of how steganography works in a digital context.

The assignment aims to demonstrate the practical application of steganography for **secure communication**, focusing on usability, efficiency, and robustness. By resizing images to ensure compatibility and validating inputs to avoid errors, the program is designed to handle common challenges in digital steganography effectively.

**Approach**

To solve the assignment, **bitwise operations** were used to embed and extract the secret image into/from the cover image. The process is divided into two main steps:

1. **Hiding the Image**:
   * Load the **cover image** and **secret image** in grayscale.
   * Resize the secret image to match the dimensions of the cover image, ensuring they are compatible.
   * Modify the least significant bits (LSBs) of the cover image to encode the most significant bits (MSBs) of the secret image.
   * Save the resulting stego image.
2. **Extracting the Image**:
   * Load the **stego image** in grayscale.
   * Extract the encoded bits from the stego image's LSBs.
   * Reconstruct the hidden secret image from these bits.
   * Save the reconstructed secret image.

This approach uses **masking** and **bit-shifting** for embedding and decoding, ensuring minimal distortion of the cover image.

**Code**

Key parts of the code include:

1. **Hiding the Image**:

stego\_image = (cover\_image & 0xF0) | (secret\_image >> 4)

cv2.imwrite(output\_path, stego\_image)

* + The cover\_image & 0xF0 operation clears the LSBs of the cover image.
  + The secret\_image >> 4 shifts the MSBs of the secret image into a position suitable for embedding.
  + The | operation combines these two to create the stego image.

1. **Extracting the Image**:

secret\_image = (stego\_image & 0x0F) << 4

cv2.imwrite(output\_path, secret\_image)

* + The stego\_image & 0x0F operation isolates the embedded bits.
  + The << 4 operation restores these bits to their original positions.
  + This reconstructs the hidden secret image.

1. **Error Handling**:

if cover\_image is None or secret\_image is None:

print("Error: One or both images could not be loaded. Check the paths.")

return

1. **Resizing the Secret Image**:

if cover\_image.shape != secret\_image.shape:

secret\_image = cv2.resize(secret\_image, (cover\_image.shape[1], cover\_image.shape[0]))

**Results**

**Test Case 1: Hiding a Secret Image**

* **Inputs**:
  + Cover Image: cover\_image.png (grayscale)
  + Secret Image: secret\_image.png (grayscale)
  + Output Path: stego\_image.png
* **Output**:
  + Stego Image: stego\_image.png successfully saved.

**Test Case 2: Extracting the Hidden Image**

* **Inputs**:
  + Stego Image: stego\_image.png
  + Output Path: extracted\_secret\_image.png
* **Output**:
  + Extracted Image: extracted\_secret\_image.png successfully saved.

**Usage**

1. Recreating the Environment:
   * + Open a terminal and navigate to the project folder.
2. Create a new virtual environment:
   * + python -m venv .venv
3. Activate the virtual environment:
   * + .venv\Scripts\activate
4. Install the required dependencies:
   * + pip install -r requirements.txt

**Correct Workflow A screenshot of a computer program

Description automatically generated**

**Error Handling Workflow A screenshot of a computer program

Description automatically generated**

**Conclusion**

This assignment successfully demonstrates the embedding and extraction of a secret image using bitwise operations. The solution ensures compatibility by resizing the secret image to match the cover image, providing accurate and reliable results. The program effectively handles errors such as invalid paths and mismatched image dimensions, making it user-friendly and robust.