INDIAN INSTITUTE OF TECHNOLOGY, BHUBANESWAR

SECURITY AND FORENSICS LAB 1

ASSIGNMENT-10

CHANDAN KESHARI
24CS06022
MTECH(CSE)

Questions

Steganography - LSB(Least Significant Bit)

1. Develop a program to embed a short text message within an image using the Least Significant Bit (LSB) technique

Goal of Assignment:

The objectives of your assignment are:

1. Develop a program to embed a short text message within an image using the Least Significant Bit (LSB) technique.
2. Allow the user to:
3. Specify the image file and the text message to be embedded.
4. Choose the color channel (red, green, or blue) in which to embed the message.
5. Implement functionality to:
6. Hide the text message within the image.
7. Retrieve and display the hidden message from the image.
8. Ensure the program works with short, unencrypted text messages and begins embedding from pixel (1,1).

Steganography

- Steganography is the practice of hiding secret information within an ordinary, non-secret medium to avoid detection. The goal is to conceal the fact that communication is taking place, unlike encryption, which protects the content of communication but does not hide its existence.

Key Features of Steganography:

- **Mediums:** Common carriers for steganography include images, audio files, videos, and even text.
- **Hidden Data:** Data such as text, images, or files are embedded into the chosen medium without making noticeable changes.
- **Stealth:** The changes are subtle enough that they are imperceptible to a casual observer.

How Steganography is Done (Using Images)**:**

- **Least Significant Bit (LSB) Technique:**
  - Digital images are composed of pixels, where each pixel's color is represented by numbers (often in binary).
  - The Least Significant Bit (LSB) of each color channel (red, green, blue) in a pixel is replaced with bits of the secret data.
  - For example:
    - Original Pixel Color (Red): *10110011*
    - LSB Modified Pixel (with secret data): *10110010*
  - This change is minimal and often imperceptible to the human eye.
- **Embedding Process:**
  - Choose a medium (e.g., an image file like .png or .bmp).
  - Convert the secret message into binary format.
  - Embed each bit of the message into the LSB of the chosen pixel channel (e.g., red, green, or blue).
- **Extraction Process:**
  - Extract the LSBs from the carrier file and reconstruct the hidden data from these bits.
  - Convert the binary data back into its original format (e.g., text).

Practical Applications:

- **Digital Watermarking:** Embedding ownership information into images or videos.
- **Secure Communication:** Hiding messages to transmit them covertly.
- **Forensics and Authentication:** Verifying the integrity of digital content.

While steganography can be effective, its security relies on keeping the embedding method secret. If the technique is discovered, the hidden data can be easily retrieved.

Prerequisites and Setup:

Required Libraries and Tools

- **OpenCV Library:** For handling image processing in C++.
- **Programming Environment:** Any C++ IDE or command-line environment.
- **Image Format:** Use .png or .bmp images as they preserve pixel data better than compressed formats like .jpg.
- Sample Image

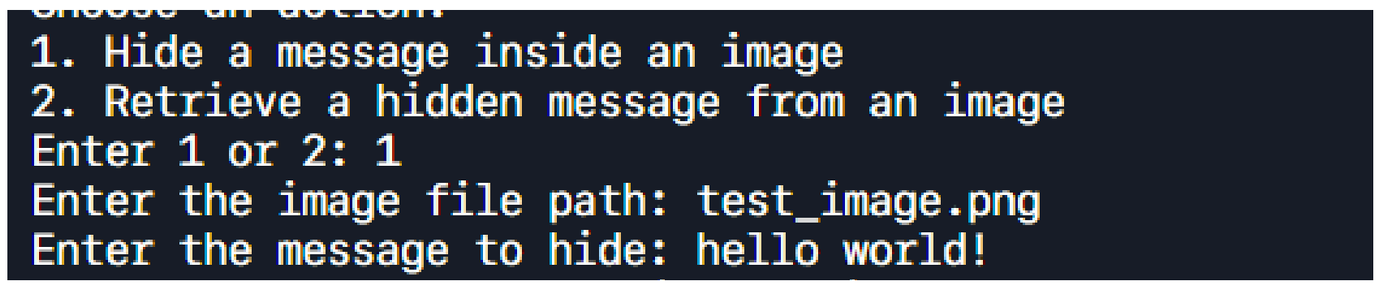
Process for Embedding a Message:

- For this project, we used the message: "Hello, World!" This short message will be converted to binary and embedded into the selected color channel of each pixel, starting from the top-left corner (pixel (1,1)).

- The program allows the user to select which color channel (red, green, or blue) will hold the hidden data. This feature provides flexibility in choosing the color channel with the least impact on the image.

- We used this image for hiding the data. It looked like this before embedding the message.

Then run main.cpp file. Then the program asks for the action you want to perform i.e. whether you want to hide a message or you want to retrieve it from an image.



Since we want to hide message inside the image so we choose option 1
then it prompts us to enter the path of the image and the message we want to embed in it.

Image name/path: **test_image.png**
Message to be embedded: **hello world!**

Then we have to select the **color channel**. The color channel is being asked because the steganography technique being used embeds the secret message into one of the RGB color channels (Red, Green, or Blue) of the image. Each channel is represented by a single byte in an image's pixel data, and you can manipulate the least significant bit (LSB) of those values to hide the message.

```
Select the color channel (r, g, b): r
libpng warning: iCCP: known incorrect sRGB profile
Message successfully hidden! The image is saved as hidden_msg.png
```

The image looks like this after embedding the text inside it, (There are only subtle changes to it)

The differences between the original image and the modified image are minimal and often visually imperceptible.

Now if we need to retrieve the embedded message from the image so we choose option 2 then it prompts us to enter the path of the image and the message we want to embed in it.

```
Choose an action:
1. Hide a message inside an image
2. Retrieve a hidden message from an image
Enter 1 or 2: 2
Enter the image file path: hidden_msg.png
Enter the length of the hidden message: 12
Select the color channel (r, g, b): r
Extracted hidden message: hello world!Ä>"bKbkPÒ
hð4g`bÓsæÔPB#=+.c¶ä¾Õ8ÕGm+@♥à☺à°
PS C:\Users\ACER\Downloads\SF_lab10>
```

Image name/path: **hidden_msg.png**

Length of the embedded message: **12**

Then we have to select the **color channel** as well.

The embedded message “hello world!” is extracted successfully.

The extracted message also includes some more bits while extracting the text from image.

These text is encountering where the extracted hidden message contains not only the intended "hello world!" message but also additional bits from the image data, occurs because the LSB (Least Significant Bit) technique only modifies the least significant bits of the image pixels. This means that when you extract the hidden message, the bits that were altered for the message are still mixed with the original pixel values of the image.

Conclusion:

In this implementation, we utilized the Least Significant Bit (LSB) technique to embed and extract hidden messages within an image's pixel values. This approach modifies the least significant bit of the selected color channel (Red, Green, or Blue) for each pixel in the image, which allows for the encoding of a message without significantly altering the visual appearance of the image.