**MO' Steganography Tool**

Name: Ahmed Mo’men Ahmed Ali

ID: 202301532

Email address: [aa2301532@tkh.edu.eg](mailto:aa2301532@tkh.edu.eg)

1. Introduction

Steganography, or the act of concealing information within other information, has been used from the beginning of time whenever information needed to be concealed.

The basic goal of steganography is to prevent unwanted access to a message by any other method. With the current technological advancements, steganography has developed in its scope since its application now extends from concealing messages within text documents to images, sounds and videos.

The spread and use of digital technologies has provided opportunities for people to embed information within digital content thus making the application more useful in the areas of cyber security and digital investigation.

The present report explains the process of development of the Python application that conceals a secret message in a BMP image file using the Least Significant Bit method. The application has the defouchers allowing to type a message which is placed within an image file.

Two main functions are performed by the application, and they are being: the insertion of the secret message into the image file and the retrieval of the secret message when required. The report will hence seek to explain the application its features, algorithms, testing methods, and problems experienced in the course of developing the application.

1. Application Overview

The steganography application developed for this coursework uses the Least Significant Bit (LSB) method that is encoding message with images. LSB steganography is based on the least significant bit modification of the individual pixel in the image. This modification does alter a bit and such a least change in the image involves a visible undetectable message that is concealed into the image. There is a wide spectrum of such techniques because they are simple and effective.

How it works:

1. Encoding Process:

- The user inputs image(s) in BMP format and message that is to be encoded.

- The application is able to transform the message into binary.

- Then the images or picture’s pixels are sequentially embedded with a message in the form of binary bits into the LSB (the least significant bit) of the image.

- Finally, a BMP file is generated containing the new image with the message in the file.

2. Decoding Process:

- the user supplies the picture which bears the concealed message.

- The software regains from the image the Scalar Quantization (number) of the non-significant bits and puts the bits together which form a message in binary bits.

- The reverse process is applicable here; the binary format is transformed back into the text which is understandable.

There exists a need to perform the above ritualistic processes so as to ensure that the message is intact within the picture and can be retrieved when required without much ease or disturbance.

1. Application Use

The aim of the application is to be simple and user-friendly. The following is the procedure which is useful for both encoding as well as decoding messages.

\* Message Hiding (Encryption):

- make sure to keep the image in bmp file format (only allow the bmp files at this application).

- copy the image path from the BMP file image.

- Enter the secret message that you would like to embed in the image.

- Enter the name of the output file which is going to be used for saving the image containing the secret message.

- After that, the software will have completed its task on the image and the file name that was selected for the output will be used to store the image.

\* View the massage (decryption):

-Find the application and the picture file location which possesses the concealed secret message.

- It will be attained or possible for the application to fetch or get the embedded secret message in the picture.

- The text message that includes the decoded portion will be displayed on the screen

1. Application Pseudocode

\*Start

\* Text to Binary Conversion

* Take a secret text the user wants to hide as input.
* Convert this massage letters a bin sequence (1,0)
* Return the converted massage (which is binary format) as result\_1.

\* Convert Binary to Text

* accept the converted the binary text consisting of (1s and 0s) as input.
* Divide the binary number into eight groups.
* Change each group into a letter.
* enter this letter in the outcome.
* Return the outcome, which is the original message.

\* Hide a Message in a Picture

* accept only a bmp Picture path as an impute
* convert and read the Picture into a data format
* use the data at result\_1
* add a special end after the converted secret message for easy recognition(01001010)
* skip the picture’s header at 54 indexes
* make a loop for each bit of the massage
* edit the least significant bit (LSB) of the byte to match the message bit
* move the edited byte back to the picture data
* take a name for the new picture from the user as input.
* save the new picture name followed by (.bmp)

\*Find a Hidden Message in a Picture

* Take the picture path as an input for the user.
* Read the pictures as data.
* Start from the 55th index of the picture data (after the header).
* Look for the LSB of data until you see the special end (01001010).
* Remove the special end (01001010).
* convert the bin massage into a string again.
* Show hidden message to user.

\*End

1. Algorithm Description

As can generally be deduced, the basis of this steganography application is the Least Significant Bit (LSB) technique. This is because the least significant bit of a color pixel in an image may only be altered in an image without any visual changes occurring. In an 8-bit color model, a single image pixel has three primary color components, i.e. Red, Green, and Blue (RGB). Now, we can cover some amount of binary data into the image by replacing only the least significant bits from each of these channels.

--Steps in the Algorithm:

1.Message Encoding:

- The binary message should be trusted to the message as it is not readable by a human.

- Starting from the first pixel, which is the video file in sequence, the binary message is divided again, and it is put into the image’s least significant bits on the perimeter.

o the program shows a pop-up message apologizing for the error when it attempts to conceal the entire message because there are not sufficient bounds.

2.Message Decoding:

- With the image now open, the key message is contained in the LSB, which is cut out from the various pixel values.

- These logic level states are combined to make up a structural binary message.

- Finally, and relaxingly, the osteological message is changed from the encoding back to the decimal system and presented.

Time Complexity:

•Encoding: O(n), where n is the number of pixels in the image. The application updates the least significant bit of each pixel once for every bit in the secret message.

•Decoding time: O(n), where n is the number of pixels in the image. To get the message, the application reads the least significant bit from each pixel once.

1. Unit Test Table

The unit tests that were carried out to verify functioning are listed in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | input | Expected output | Status |
| Encrypt a short massage | Image: flour.bmp  message: "Hi :)" | Your secret has been saved successfully and saved as: secret flour.bmp  (Small bmp file = short message) | pass |
| Encrypt a long massage | Image: dinosaur.bmp  message: "I won't be at university tomorrow. Take my attendance and tell him I'm in the bathroom." | Your secret has been saved successfully and saved as: secret dinosaur.bmp  )large bmp file = large message( | pass |
| Decrypt an Encrypted message | secret dinosaur.bmp path | The decrypted message: “"I won't be at university tomorrow. Take my attendance and tell him I'm in the bathroom." | Pass |
| Decrypt a non-encrypted message | flour.bmp path | Error | pass |
| Invalid format | Star.jbg | Error | Pass |
| Too long message | Image: flour.bmp  message: "I won't be at university tomorrow. Take my attendance and tell him I'm in the bathroom." | Shorten your message, please!. | pass |

1. **Repository Link**

The project is available in the repository. Here is the <https://github.com/ahmed-momen61/Introduction-to-Programming---EHC-LVL4.git>

 (Ensure the commit or tag is before the deadline).

1. References

Steganography (Wikipedia): <https://en.wikipedia.org/wiki/Steganography>

**Bitwise Operations (geeksforgeeks)**: <https://www.geeksforgeeks.org/bitwise-operators-in-python/>

Understanding LSB Steganography **(geeksforgeeks):** [**https://www.geeksforgeeks.org**](https://www.geeksforgeeks.org/)

**LSB** Steganography(Steganography Wiki): <https://steganography.wikia.org/wiki/Least_significant_bit>

**String Encoding and Decoding at python:** <https://realpython.com/python-string-methods/>

What is Steganography? **(TechTarget):** <https://www.techtarget.com/>

Steganography Techniques: **(ResearchGate):** <https://www.researchgate.net/>

S. Katzenbeisser and F. Petitcolas, "Information Hiding Techniques for Steganography and Digital Watermarking," Artech House, 1999.

1. Source Code

* interface part

print("Welcome to [BMP MO' Steganography]\n \*\*Use it wisely:)\*\*")

print("Make your choice:")

print("1. Encrypt message")

print("2. decrypt message")

choice = input("Enter your choice: ")

if choice == "1":

    path = input("Enter the BMP image path: ")

    msg = input("Enter Your secret message: ")

    output = input("Enter the output image name: ")

    encrypt\_part(path, msg, output)

elif choice == "2":

    path = input("Enter the image path to decrypt: ")

    decrypt\_part(path)

else:

    print("Invalid choice!")

* conversion part

def String\_to\_bin(text):

    bin\_MSG = ""

    for char in text:

        bin\_MSG += format(ord(char), '08b')

    return bin\_MSG

def bin\_to\_String(binary):

    text = ""

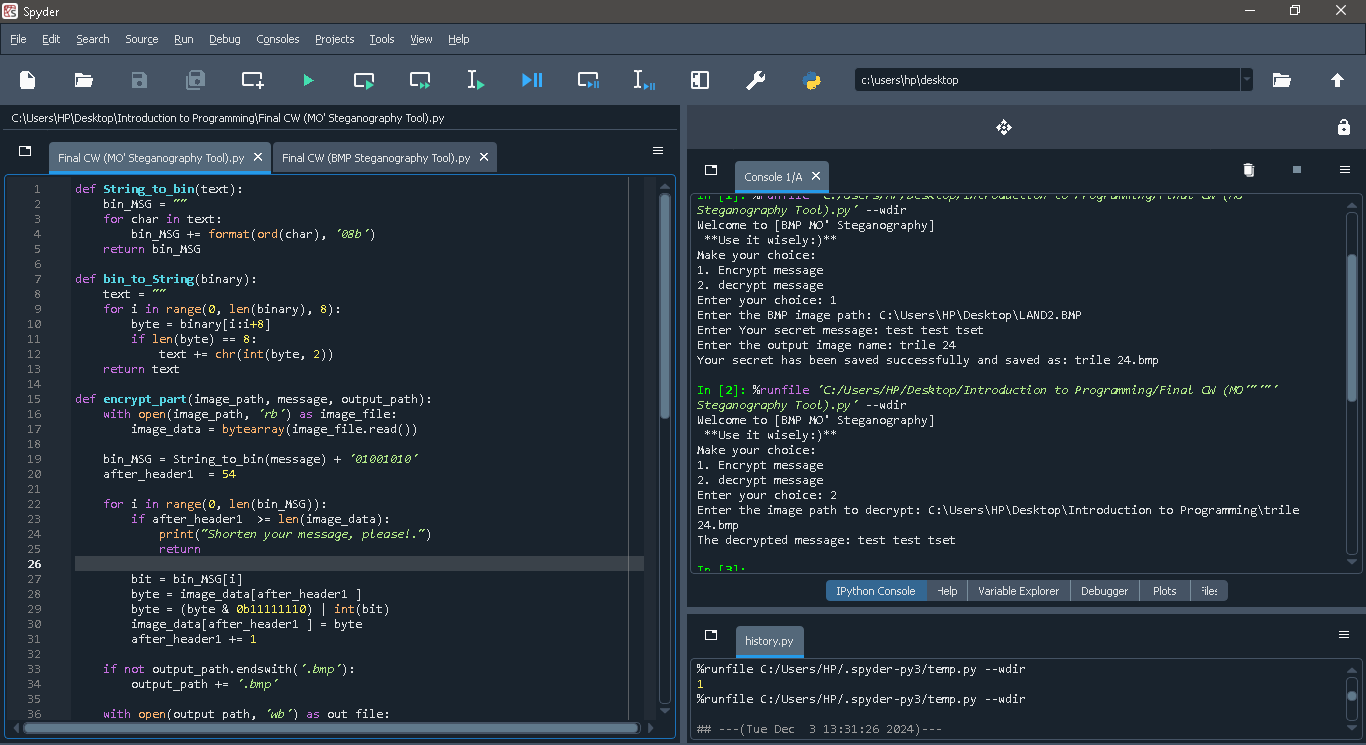
    for i in range(0, len(binary), 8):

        byte = binary[i:i+8]

        if len(byte) == 8:

            text += chr(int(byte, 2))

    return text

1. Test.24
2. Conclusion

The developed steganography tool based on Python is able to embed and extract messages from BMP format image files. Through the use of the Least Significant Bit (LSB) technique, the application guarantees that the concealed messages will not be visible to the naked eye while at the same time easy retrieval through encoding is possible. In addition, there is no need to use sophisticated libraries as ordinary features of Python are utilized which promotes ease of use but still durable for real life use.