Indian Institute of Technology Bhubaneswar

Security & Forensics Lab II

Laboratory Experiment No. 1

(Held on 07/01/2022)

Submitted By:

Ayush Modi

21CS06001

M. Tech, CSE (1st year)

Aim

The aim of the experiment is to hide the message in an image and extract the hidden message from the encoded image.

Theory

Steganography is the technique of hiding data secretly within an ordinary, non-secret, file or message in order to avoid detection. The use of steganography can be combined with encryption as an extra step for hiding or protecting data. Steganography can be used to conceal almost any type of digital content, including text, image, video or audio content; the data to be hidden can be hidden inside almost any other type of digital content.

In modern digital steganography, data is often encrypted and then inserted, using a special algorithm, into data that is part of a particular file such as a JPEG image, audio, or video file. The secret message can be embedded into ordinary data files in many different ways. One technique is to hide data in bits that represent the same color pixels repeated in a row in an image file. By applying the encrypted data to this redundant data in some inconspicuous way, the result will be an image file that appears identical to the original image but that has “noise” patterns of regular, unencrypted data.

Steganography with LSB –

Input

1. **Filename -** which is to be used to hide the message**.**
2. **Key -** An alpha-numeric key is used to determine the location of the message bits.
3. **Message -** Alpha-numeric message (up to 4000 characters in Grey-scale images and up to 12000 characters in colour images).

Hiding the message

This is a symmetric key algorithm, i.e. the same key is used to hide the data and then extract the hidden data.

Every character of the message (including symbols and numbers) is converted to its 8-bit binary code using ASCII values. Similarly, every character of the key is converted to its 8-bit binary code using ASCII values.

For example Binary equivalent of ***Hello*** is:

0100100001100101011011000110110001101111

Now, we traverse the pixels from left to right (along the width) from top to bottom first in dimension-1. If it is an RGB image, we traverse in a similar fashion to dimension-2 and dimension-3. While traversing, we check the value of the key at key index. If it's 1, we replace the LSB of the pixel with the value of the message at message index. When the message is over, we stop traversing and give the output image.

For example: Let's say we wish to hide the below data in an RGB image

Message - *Steganography is the art of concealing information. In computer science, it refers to hiding data within a message or file. It serves a similar purpose to cryptography, but instead of encrypting data, steganography simply hides it from the user.*
(Binary - 01000011……)

Key - *Hello (Binary -* **0100100001100101011011000110110001101111***)*

While traversing the image, we ignore the first pixel as key[0] is 0 and we move to the next pixel. Key[1] is 1. So, we put the message[0] bit (i.e. 0) in the LSB of pixel 219 (at image[0][1][0]).

216	**219**	217		216	**218**	217
176	**177**	175	**>>>**	176	**177**	175
218	216	216		218	216	216
(Original Image)				(Image with message encoded)		

Binary Representation:

11011000	**11011011**	11011001		11011000	**11011010**	11011001
10110000	**10110001**	10101111	**>>>**	10110000	**10110001**	10101111
11011010	11011000	11011000		11011010	11011000	11011000
(Original Image)				(Image with message encoded)		

We continue putting the message bits in the image until all the message is encoded.

Algorithm -

Convert the key and message into their corresponding binary representation.
Initialize message_index to 0
Initialize key_index to 0

```
for(k from 0 to dim) {          // k = 1 for Grey-scale image and 3 for RGB image
   for(i from 0 to height) {
      for(j from 0 to width) {
        Increment key_index by 1
        If(key[key_index % key_length] == 1) {
            If(message_index < message_length) {
              LSB of the current pixel is replaced with the message[message_index]
              Increment the message_index by 1
            }
            Else
               Exit      // All the message has been successfully hidden in the image.
        }
        Else
           Continue;  // Skip the current pixel
      }
   }
}
```

Extracting the hidden message from the image -

This is a symmetric key algorithm, so the same key is required to extract the hidden data as the one which was used to encode the data in the image.

Every character of the key is converted to its 8-bit binary code using ASCII values.

For example Binary equivalent of **Hello** is:

0100100001100101011011000110110001101111

Now, we traverse the pixels from left to right (along the width) from top to bottom first in dimension-1. If it is an RGB image, we traverse in a similar fashion to dimension-2 and dimension-3. While traversing, we check the value of the key at key index. If it's 1, we append the LSB of the pixel to the newChar. If the length of the newChar is 8, a new character is converted to an integer and then to Character. This character is added to the result.

For example: Let's say the output image pixels are as follows:

216	**218**	217		11011000	**11011010**	11011001
176	**177**	175	>>>	10110000	**10110001**	10101111
218	216	216		11011010	11011000	11011000

While traversing the image, we ignore the first pixel as key[0] is 0 and we move to the next pixel. Key[1] is 1. So, the newChar is added with 0 as the LSB of the pixel is **0**. Similarly, we add 1 to newChar - **01** for 177.

Algorithm -

```
Convert the key into its binary representation.
Initialize message_index to 0
Initialize key_index to 0
Initialize extracted_message to empty string                // initially the message is empty
Initialize newPixel as 0
Initialize length_newPixel to 0

for(k from 0 to dim) {          // k = 1 for Gray-scale image and 3 for RGB image
   for(i from 0 to height) {
      for(j from 0 to width) {
        Increment key_index by 1
        If(key[key_index % key_length] == 1) {
            If(message_index < message_length) {
              Multiply the newPixel with 10 and add the LSB of the current pixel
              Increment the message_index by 1
            }
            Else
               Exit      // Complete message has been successfully extracted.
        }
        Else
           continue;   // Skip the current pixel
      }
   }
}
```

Result

Below are the screenshots of the output image (i.e. encoded image) along with the extracted message from the decoding function.

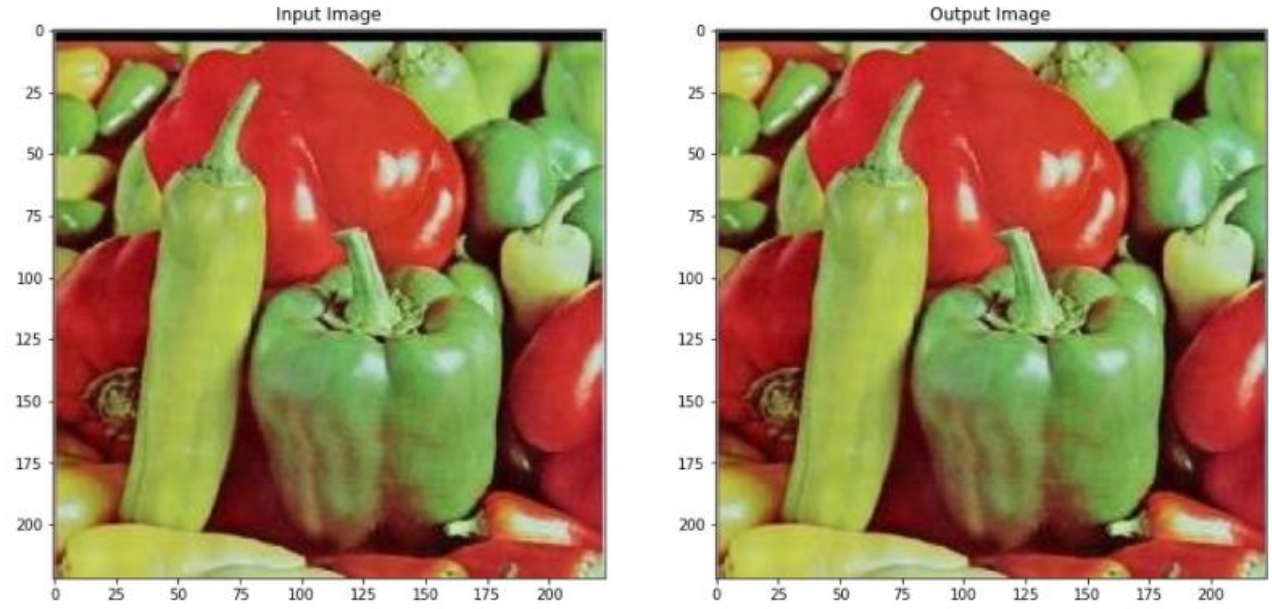
RGB image –

```
fileName = input("Enter the filename: (eg- Cover_1.png or Cover_2.png)")
key = input("Enter the key:")
message = input("Enter the message to be hidden:")

inputImage, outputImage = encodingMessage(fileName, key, message)
extractedMessage = extractData(key)

plt.figure(figsize=(15,15));
plt.subplot(1,2,1),plt.imshow(cv2.cvtColor(inputImage, cv2.COLOR_BGR2RGB)), plt.title("Input Image")
plt.subplot(1,2,2),plt.imshow(cv2.cvtColor(outputImage, cv2.COLOR_BGR2RGB)), plt.title("Output Image")
print("Extracted message = {}".format(extractedMessage))
```

```
Enter the filename: (eg- Cover_1.png or Cover_2.png)Cover_2.png
Enter the key:auhwdihawhdiawi
Enter the message to be hidden:hauwdhiuahw ahuihwfiu ahfah giahugh hgauihgihaghiuahwigha789a789wy789fa9wd a9w9da w
Extracted message = hauwdhiuahw ahuihwfiu ahfah giahugh hgauihgihaghiuahwigha789a789wy789fa9wd a9w9da w
```



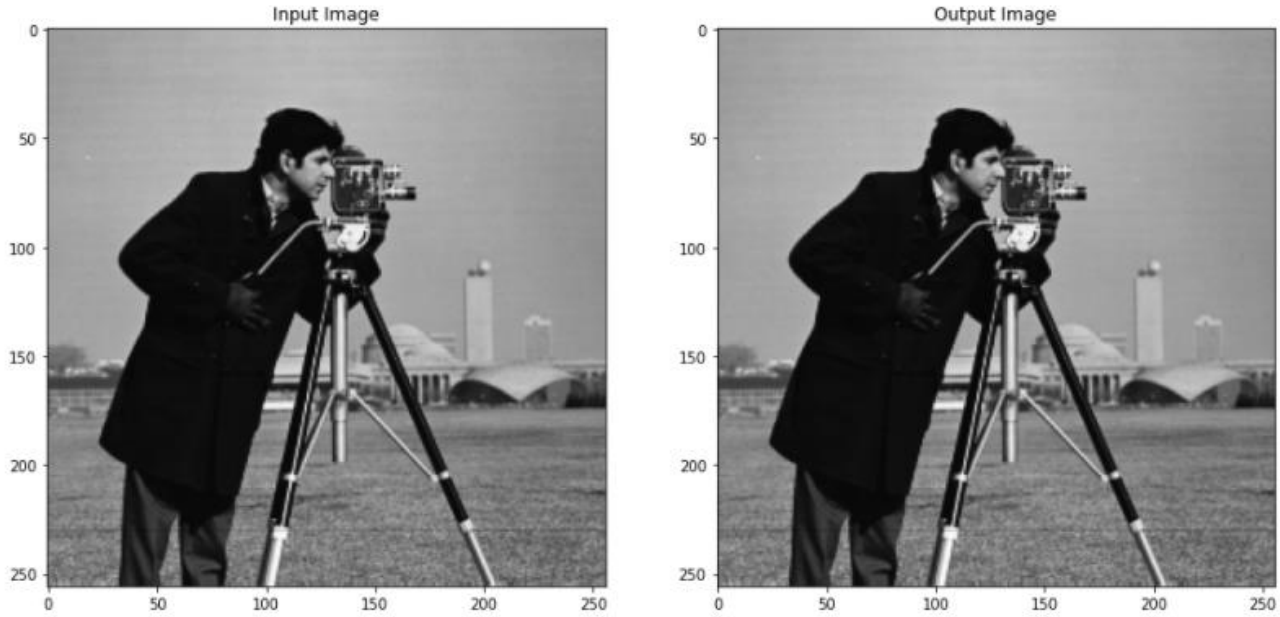
Gray-scale image -

```
fileName = input("Enter the filename: (eg- Cover_1.png or Cover_2.png)")
key = input("Enter the key:")
message = input("Enter the message to be hidden:")

psnr, inputImage, outputImage = encodingMessage(fileName, key, message)
extractedMessage = extractData(key)

plt.figure(figsize=(15,15));
plt.subplot(1,2,1),plt.imshow(cv2.cvtColor(inputImage, cv2.COLOR_BGR2RGB)), plt.title("Input Image")
plt.subplot(1,2,2),plt.imshow(cv2.cvtColor(outputImage, cv2.COLOR_BGR2RGB)), plt.title("Output Image")
print("Extracted message = {}".format(extractedMessage))
```

```
Enter the filename: (eg- Cover_1.png or Cover_2.png)Cover_1.png
Enter the key:Hello this is sflab
Enter the message to be hidden:Hello, this is iit. First lab of second semester. 576890
Extracted message = Hello, this is iit. First lab of second semester. 576890
```



Conclusion

The message was hidden in the image. Then using the same key, data was extracted from the encoded image.

Code

https://drive.google.com/drive/folders/1j6qGxEai4qfkVwNmG2EB0Ss4IVyt1LX9?usp=sharing