NEW HORIZON
COLLEGE OF ENGINEERING
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

A MINI PROJECT REPORT ON

IMAGE STEGANOGRAPHY

Submitted in partial fulfillment for the award of the degree of Bachelor of Engineering
In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

PREKSHA SHRIDHAR

1NH17CS102

V 'B'

DURING
ODD SEMESTER 2019-2020
For
COURSE CODE 'CSE56'

Reviewed by

DR. KAMATCHI PRIYA
ASSOCIATE PROFESSOR, DEPT. OF CSE

NEW HORIZON
COLLEGE OF ENGINEERING
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

CERTIFICATE

This is to certify that the mini project work titled

IMAGE STEGANOGRAPHY

Submitted in partial fulfillment for the award of the degree of Bachelor of Engineering

PREKSHA SHRIDHAR
1NH17CS102

During the academic year
2018-2019

Signature of Reviewer **Signature of HOD**

Semester End Examination

Name of the Examiner	*Signature with date*
1. ……	*……*
2. ……	*……*

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be, but impossible without the mention of the people who made it possible, whose constant guidance and encouragement crowned my efforts with success.

I thank the management, **Dr. Mohan Manghnani**, Chairman of NEW HORIZON EDUCATIONAL INSTITUTIONS for providing necessary infrastructure and creating good environment.

I also record here the constant encouragement and facilities extended to me by **Dr. Manjunatha**, Principal, NHCE, **Dr.Prashanth. C. S. R**, Dean Academics, **Dr.B. Rajalakshmi**, Head of the Department of Computer Science and Engineering. I extend my sincere gratitude to them.

I express my gratitude to **Dr. Kamatchi Priya,associate professor** my project reviewer for constantly monitoring the development of the project and setting up precise deadlines. Her valuable suggestions were the motivating factors in completing the work.

Finally a note of thanks to all the teaching and non-teaching staff of Computer Science and Engineering Department for their cooperation extended to me and my friends, who helped me directly or indirectly in the course of the project work.

Preksha Shridhar
1NH17CS102

ABSTRACT

Image steganography is an algorithm-based project on hiding the secret data inside an image. It is the method of secret communication between the communicators. The main goal of this project is to hide the existence of the secret message from unauthorized party. For hiding the information in the images, there exists a large variety of steganographic techniques. In this project I have implemented image steganography, which uses the least significant bits(LSB) algorithm. The basic idea of this algorithm is to replace the least bit of the each pixel with the bit of the secret data that is to be hidden. After extracted the encoded image, there is no much difference in the quality of the encoded image and the original image.

TABLE OF CONTENTS

CHAPTERS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In this project, LSB algorithm is used to hide the data inside an image using steganography method.

Issues related to the privacy of the information can arise in different fields such as healthcare records, criminal justice investigations and proceedings, financial institutions and transactions, biological traits, residence, and geographic records and ethnicity. Privacy of the data or data security has become increasingly important as more systems are connected to the Internet. There are information privacy laws that cover the security or privacy of the information on private individuals from intentional or unintentional disclosure or misuse.

Steganography is another method for privacy and security. Instead of encrypting we can hide the secret data in another innocuous-looking carrier so that their existence is hidden and unknown from the unauthorized people or group.

The word 'Steganography' is made up of the two Greek words steganos and graphei meaning "concealed writing". In essence, Hiding things in plain sight. Steganography is a way of exchange of information in which the exchange remains unseen. The focus on data security is to ensure privacy while protecting personal data or data related to corporate life. Privacy, on the other hand, is the ability of an individual or group to hide them or information about themselves and thereby selectively revealing them.

There are different types of steganography. Like image steganography, video steganography, audio steganography, etc.

Although this method began many centuries ago, this field became popular recently due to an increase in the interest of one in the field of technology. People are communicating and transferring secret information via computers and the internet.

Hence, in this report, one of the methods of steganography used is explained in detail.

1.1 COURSE OBJECTIVES

- Acquiring knowledge of python programming language.
- Acquiring knowledge of algorithm design.
- Learn to implement time complexity.
- Develop problem solving ability and gain debugging skills.

1.2 PROBLEM DEFINITION

In this project the image steganography is implemented using the LSB(least significant bit) method. This project is used for hiding and unhiding the secret message.

The terms used in this project are:

- Cover image
- Stego image
- Message conversion to binary
- Altering the LSB of the pixel of the image

LSB method is one of the simplest algorithms used by worldwide for the steganography technique. This method changes the least bit value of few of the pixels. There are encoding and decoding part in this method. The message sender encodes the message inside the image and sends it to the decoder via internet. The decoder decodes the image and receives the secret message that was to be transferred. In the encoding part the message to be used is chosen. The message to be transferred is typed and this message will be converted to the binary form. There will be traversing through the pixels of the image and will convert the RGB of image pixels to the binary form. Now each of the least bit of the RGB of pixels are replaced by one bit of the binary form of the secret message.

For example:

A grid for **3 pixels** of a image can be as follows:

(00101101 00011100 11011100)

(10100110 11000100 00001100)

(11010010 10101101 01100011)

Let the binary form of the message to be hidden be **1100100.**

The after substitution using this project method the output will be:

(0010110**1** 0001110**1** 1101110**0**)

(1010011**0** 1100010**1** 0000110**0**)

(1101001**0** 10101101 01100011)

This will be the pixel value of the image after encoding.

In the decoding part the stego image will be chosen. And LSB of each RGB value of the pixel will be extracted. And that extracted binary form will be converted to the string form. And the message will be displayed.

For example:

Let's say the extracted LSB will be 011001001000101111010010101111

When this extracted binary form is converted to string form. The message displayed will be: 'HELLO'

The main advantage of this project I have done if that it can used for any type of image and stored in any format.

1.3 OUTCOME OF THE PROJECT

There is encoding and decoding part of the project.

Encoding has different functions. To extract the image, to extract the message. To convert the message and RGB values to binary form. And substitute the message bits into the LSB of the pixel bits.

When this is executed, there will be a message which will ask for the cover image location or name. enter the cover image location or name with the image format. The message which asks for the secret data will be displayed. Enter the secret message. Scanning of image and converting of pixels to binary processing takes place. And then it will ask for the stego image name to be saved. So enter the stego image name with the format of the image. And message will be displayed which says ‘ the message encoded into the image stego_name.format . The image in which the message is encode will also be displayed.

Decoding has two functions. One to extract list of LSB bit of the RGB of pixel. And the other function to convert the binary form to the string form.

When it is executed, the message will be displayed asking for the stego image to decode. Enter the stego image name with the format. And it decodes the image and displays the message that was hidden inside the image.

CHAPTER 2

REQUIREMENTS

2.1 HARDWARE REQUIREMENTS:

- PIV 2.8 GHz Processor and Above
- RAM 512MB and Above
- HDD 20 GB Hard Disk Space and Above

2.2 SOFTWARE REQUIREMENTS:

- Windows
- Python IDLE
- Image folder

CHAPTER 3

DESIGN AND IMPLEMENTATIONS

3.1 ALGORITHM

Encoding:

Step-1: Extract the cover image and secret text information which is to be embedded in to the cover image.

Step-2: Enter the secret message and check if the image capacity is greater than the message capacity. If true then continue. Else print 'not sufficient'.

Step-3: Read the secret message and encrypt it by converting it to binary format.

Step-4: Get the pixels of cover image and convert to binary format.

Step-5: Find LSBs of each RGB pixels of the cover image.

Step-6: The message bits are taken sequentially and then are placed in LSB bit of pixels.

Step-7: Same procedure is followed till all the message bits are placed in the cover image.

Step-8: The image generated is called stego image and is ready to transmit through the internet.

Decoding:

Step-1: Extract the stego image.

Step-2: Find LSBs of each RGB pixel of the stego image.

Step-3: Find and retrieve the LSBs of each RGB pixel of the stego image.

Step-4: Continue the process until the message is fully extracted from stego image.

Step-5: Decompress the extracted secret data by converting back to characters from binary.

Step-6: Reconstruct the secret information and display the message on the screen.

3.2 PSEUDO CODE

ENCODE:

```
i=0
//GET THE SIZE OF THE COVER IMAGE
for X <- 0 TO width
    for y <- 0 to height
       pixel <- get_list_of_pixels(x,y)
       for n <- 0 to 3  // (RGB values)
         if  i < len(data)
            pixel[n] <- pixel[n] & ~1 | data[i] //Doing the LSB operation
            i+=1
```

DECODE:

```
//GET THE SIZE OF THE STEGO IMAGE
for x <-0 to width
   for y <-0 to height
       //GET THE PIXEL VALUES
         for n <-0 to 3 //(RBG VALUES)
            //GET PIXEL LIST AFTER ANDING WITH 1
     //CONVERT THE EXTRACTED PIXEL VALUE TO STRING TO GET THE SECRET DATA
```

3.3 TIME COMPLEXITY

The complexity of an algorithm is the measure of the amount of time required by an algorithm for an input of a given size. The most significant factors are a complexity of underlying algorithm and size of the input by which the time complexity of the algorithm depends.

ASYMPTOTIC ANALYSIS

Asymptotic analysis is the computation of the running time of any piece of code or the operation. Its operation is computed in terms of a functions like f(n).

The time required by the algorithm are of three types:

Worst case -> It is the maximum time required by an algorithm and it is mostly used or done while analyzing the algorithm.

Best case -> It is the minimum time required for the algorithm or piece of code and it is not normally calculated while analyzing the algorithm.

Average case -> It is the average time required for an algorithm or portion of code and it is sometimes done while analyzing the algorithm.

ASYMPTOTIC NOTATION

The commonly used notation for calculating the running time complexity of any algorithm is as follows:

- Big O notation
- Big θ notation
- Big Ω notation

Big Oh Notation, O

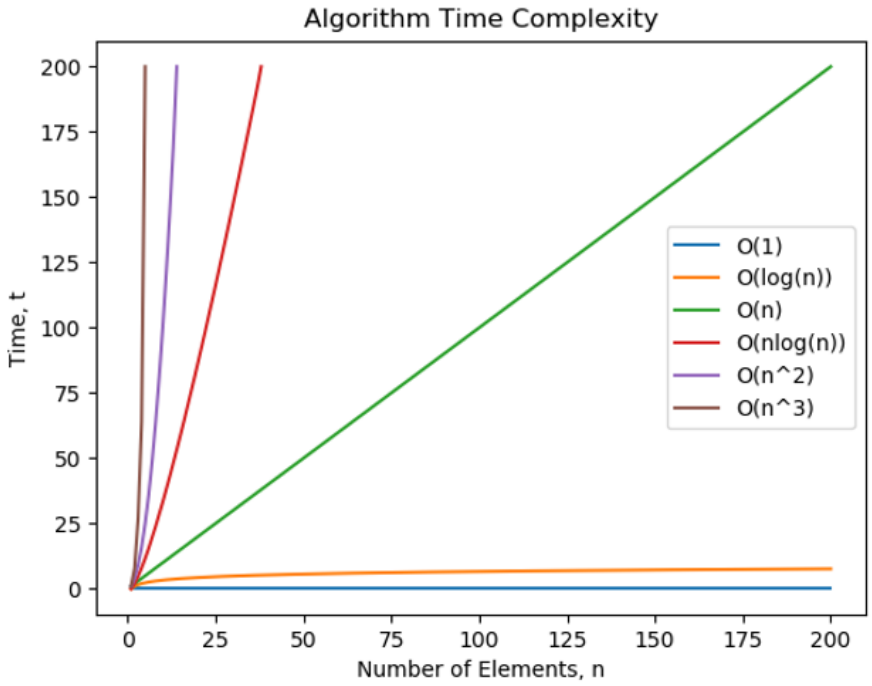
Big-O -> It is used to measure the performance or complexity of an algorithm. In more mathematical term, it is the upper bound of the growth rate of a function, or that if a function g(x) grows no faster than a function f(x), then g is said to be a member of O(f).In general, it is used to express the upper bound of an algorithm and which gives the measure for the worst time complexity or the longest time in which an algorithm might take to complete.

Big Omega Notation, Ω

The notation Ω(n) ->It is the formal way which expresses the lower bound running time of an algorithm. this is used to measures the best case time complexity or the best amount of time possibly taken by an algorithm to complete.

Big Theta Notation, θ

The notation θ(n) -> It is the formal way to express both the lower bound and the upper bound of an algorithm's running time.



3.3a. Algorithm time complexity

FINDING THE TIME COMPLEXITY OF THE ALGORITHM USED IN THE PROJECT CODE:

Encode:

i=0----------1

//GET THE SIZE OF THE COVER IMAGE----------1

for X <- 0 TO width ----------W

for y <- 0 to height----------H

pixel <- get_list_of_pixels(x,y)----------1

for n <- 0 to 3 // (RGB values)----------3

if i < len(data)----------1

pixel[n] <- pixel[n] & ~1 | data[i] //Doing the LSB operation-------1

i+=1----------1

Decode:

```
//GET THE SIZE OF THE STEGO IMAGE----------------------------------------------1
for x <-0 to width-----------------------------------------------------------------------W
   for y <-0 to height----------------------------------------------------------------H
      //GET THE PIXEL VALUES-------------------------------------------------------1
        for n <-0 to 3 //(RBG VALUES)-----------------------------------------3
          //GET PIXEL LIST AFTER ANDING WITH 1-----------------------------1
```

TIME COMPLEXITY OF THE ALGORITHM USED IN THIS PROJECT:

The time analysis for the algorithm used is O(W*H*3).

There are three for loops. The first for loop runs for width times. The second for loop runs for height times. And the third for loop runs for three times. Since these for loop are inside the other for loop the code runs for W*H*3 times. So the time complexity will be **O(W*H*3)**

CHAPTER 4

CODE SNAPSHOTS

```
def canEncode(message, new_img):
       width, height = new_img.size
       imageCapacity = width * height * bitsPerPixel
       messageCapacity = (len(message) * bitsPerChar) - (bitsPerChar + maxBitStuffing)
       if imageCapacity <messageCapacity:
           print("not sufficient")
       else:
           return imageCapacity >= messageCapacity
```

4.1 CHECKING SIZE

This function calculates the image capacity and the message capacity. Compares both the capacity. If the capacity of message exceeds the image capacity then it prints 'not sufficient'

```
def str2bina(a):
    c =''.join(['0'*(8-len(bin(ord(i))[2:]))+(bin(ord(i))[2:]) for i in a])
    return c
```

4.2 STRING TO BINARY

This code snippet is of the function which converts the given secret message to binary form and returns the value.

```
def encode(img,data):
    i=0

    width, height = img.size
    for x in range(0, 50):
        for y in range(0, 50):
            pixel = list(img.getpixel((x, y)))

            for n in range(0,3):
                if(i < len(data)):

                    pixel[n] = pixel[n] & ~1 | int(data[i])

                    """pixel[n] & ~1 will clear the
                    LSB from the channel and then we only need
                    to change to a 1 when our data says so, running | 1 will tur
                    that LSB into a 1 and | 0 will keep it as is."""
                    i+=1

            img.putpixel((x,y), tuple(pixel))

    filename=input("enter the steg image name: ")

    new_img.save("{}".format(filename), "bmp")
    print("Message encoded in the file {}".format(filename))
```

4.3 ENCODING FUNCTION

This is the snippet of the encoding from the code. All the pixel values are stored in list and by ANDing with 1 the least bit is removed and by ORing with the message bit the value in the least bit of pixel is filled. By this the message will be stored in the pixel of the image.

```
def decode(img):
    extracted_bin = []

    width, height = img.size
    byte = []
    for x in range(0,width):
        for y in range(0,height):
            pixel = list(img.getpixel((x, y)))
            for n in range(0,3):
                extracted_bin.append(pixel[n]&1)

    data = "".join([str(x) for x in extracted_bin])
    return data
```

4.4 DECODING FUNCTION

This is the decoding part of the code. In this the list of pixels are collected and the least bit of each RGB values are extracted. And the list of least bit of every pixel values is extracted from the encoded image.

```
def steg_find(bin_str):

    for i in range(0,1):

        mes=''.join(chr(int(bin_str[i*8:i*8+8],2)) for i in range(len(bin_str)//8))
    return mes
```

4.5 BINARY TO STRING

This is the snippet of part of code where the extracted least bit value of pixels of stego image which was in binary form is converted to the string form. This is done to extract the actual image.

CHAPTER 5

OUTPUT SNAPSHOTS

EXECUTING THE DECODE FUNCTION

```
==== RESTART: C:/Users/Preksha/Desktop/python/Aproject/ENC_ALTER_2.py ====

              ENTER THE COVER IMAGE WITH FORMAT :
```

5.1 COVER IMAGE

Enter the location or the name of the cover image that is to be used to encode the message in it by altering the pixel value of the image according to the binary format of the message.

```
ENTER THE COVER IMAGE WITH FORMAT : rose.JPEG
ENTER THE MESSAGE TO BE ENCODED AND END WITH DELIMITER '%' :HELLO! WELCOME TO
```

5.2 SECRET DATA

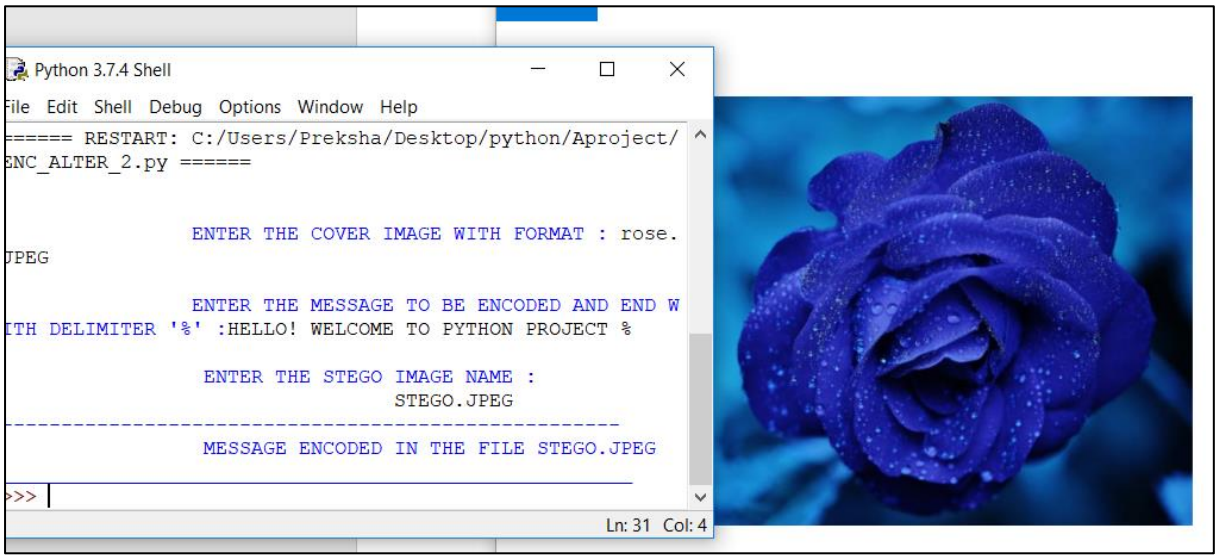
Enter the secret data that is to be hidden in the image that is to be transferred via net. Data can be related to anything.

```
ENTER THE COVER IMAGE WITH FORMAT : rose.JPEG

ENTER THE MESSAGE TO BE ENCODED AND END WITH DELIMITER '%' :HELLO! WELCOME TO PYTHON PROJECT %
```

5.3 END OF MESSAGE

At the end of the message, mention the delimiter '%'. This signifies the end of the message that is to be hidden. This to know that all the message has been extracted from the image.



5.4 STEGO IMAGE

After encoding it'll display the name of the image which is encoded also known as stego image and displays the same image in which the message is hidden to confirm.

EXECUTING THE DECODE FUNCTION

```
ENTER THE STEGO IMAGE TO EXTRACT THE SECRET MESSAGE: STEGO.JPEG
```

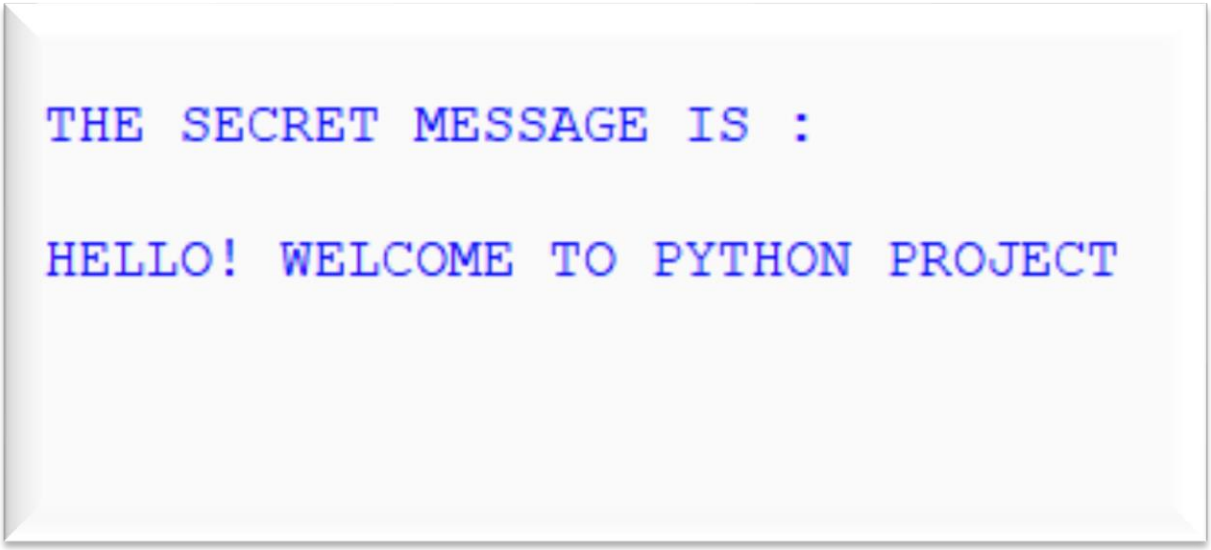
5.5 STEGO IMAGE LOCATION

While executing the decoding part, it'll request for the location of the stego image or the image name. The name with the format of the image should be given to decode the image. This is done to extract the message hidden inside the image that has been secretly sent via net.

```
                ENTER THE STEGO IMAGE TO EXTRACT THE SECRET MESSAGE: STEGO.JPEG
searching for image...
decoding in process...
```

5.6 PROCESSING

The process of searching for the image and decoding it takes little time. While it's loaded the decoder goes to the particular location mentioned and searches for the image that has been asked for to decode. And while processing the decode, it runs through the pixel to extract the binary form for further process.



5.7 FINAL DISPLAY

Finally the message is extracted from the image and is displayed on the screen.

This is done by extracting all the pixel values. Then converting the pixel values to binary form. And all the least bit of the binary form of the image is taken in the list. With the help of the function which converts binary form to string. The list of least bit of pixel values extracted are converted to string form. And the string which is the secret message is displayed to the decoder on the screen.

- BELOW IS THE REPRESENTATION OF THE ORIGINAL IMAGE AND THE ENCODED IMAGE

5.8 ORIGINAL IMAGE (COVER IMAGE)

This is the representation of the original JPEG image file.

5.9 ENCODED IMAGE (STEGO IMAGE)

This is the representation of the encoded image.

CHAPTER 6

CONCLUSION

In this report it's all discussed in the detail the study of LSB method for image Steganography. In LSB the least significant bits of pixel values of the cover image are replaced with the binary format of the secret data. Since LSB doesn't contain any information there will be no loss of information and secret image recovering back become undistorted. In this report It's also mentioned about an approach through which security of data is improved by using LSB method algorithm. This method can be used to hide the information in any type of image format. But as the size of image increases the time required to traverse through the image pixel also increases. The quality of the stego(encoded) image almost remains the same as the cover(original) image. This can be observed by the image represented above. This method is very useful to hide the information regarding the message passing. As more and more the technology is getting developed this technique would be very useful for transferring secret messages. This can be used to share messages across country. Using this technique would even increase the privacy and hide the information shared by two communicators.

CHAPTER 7

REFERENCE

http://www.ijaerd.com

https://pdfs.semanticscholar.org/132c/f16fb1e764b3f443b2cd14a941111e820e29.pdf

https://www.semanticscholar.org/paper/A-Secure-Image-Steganography-Based-on-RSA-Algorithm-Kumar-Sharma/3a6b804a11700a74a6623312b61fd164f2d2214d

https://www.researchgate.net/profile/Osamah_Al-qershi/publication/292310394_Image_Steganography_Techniques_An_Overview/links/57f642ac08ae8da3ce574080/Image-Steganography-Techniques-An-Overview.pdf

https://pdfs.semanticscholar.org/45a1/f9a9cf2d4ead6a8675c506202d33255534cc.pdf

http://www.ijarcsse.com/index.php/ijarcsse