Table of Contents

1. CHAPTER

INTRODUCTION

1.1 What Is Steganography?

Steganography is a Greek word which means concealed writing. The word steganos means covered and graphial means writing. Thus, steganography is not only the art of hiding data but also hiding the fact of transmission of secret data. Steganography hides the secret data in another file in such a way that only the recipient knows the existence of message. In ancient time, the data was protected by hiding it on the back of wax, writing tables, stomach of rabbits or on the scalp of the slaves. But todays most of the people transmit the data in the form of text, images, video, and audio over the medium. In order to safely transmission of confidential data, the multimedia object like audio, video, images are used as a cover sources to hide the data. Steganography is defined as the study of invisible communication. Steganography usually deals with the ways of hiding the existence of the communicated data in such a way that it remains confidential. It maintains secrecy between two communicating parties. In image steganography, secrecy is achieved by embedding data into cover image and generating a stego-image. There are different types of steganography techniques each have their strengths and weaknesses. In this paper, we review the different security and data hiding techniques that are used to implement a steganography such as LSB , ISB ,etc.

In todays world, the communication is the basic necessity of every growing area. Everyone wants the secrecy and safety of their communicating data. In our daily life, we use many secure pathways like internet or telephone for transferring and sharing information, but it's not safe at a certain level. In order to share the information in a concealed manner two techniques could be used. These mechanisms are cryptography and steganography. In cryptography,the message is modified in an encrypted form with the help of encryption key which is known to sender and receiver only. The message cannot be accessed by anyone without using the encryption key. However, the transmission of encrypted message may easily arouse attackers suspicion, and the encrypted message may thus be intercepted, attacked or decrypted violently. In order to overcome the shortcomings of cryptographic techniques, steganography techniques have been developed. Steganography is the art and science of communicating in such a way that it hides the existence of the communication. Thus, steganography hides the existence of data so that no one can detect its presence. In steganography the process of hiding information content inside any multimedia content like image, audio, video referred as a Embedding. For increasing confidentiality of communicating data both techniques may combined. Application of Steganeography:

i) Confidential Communication

ii) Protection of Data Alteration

iii) Access Control System for Digital Content Distribution

iv) E-Commerce

v) Media

vi) Database Systems.

1.2 History

The first recorded uses of steganography can be traced back to 440 BC in Greece, when Herodotus mentions two examples in his Histories. Histiaeus sent a message to his vassal, Aristagoras, by shaving the head of his most trusted servant, "marking" the message onto his scalp, then sending him on his way once his hair had regrown, with the instruction, "When thou art come to Miletus, bid Aristagoras shave thy head, and look thereon." Additionally, Demaratus sent a warning about a forthcoming attack to Greece by writing it directly on the wooden backing of a wax tablet before applying its beeswax surface. Wax tablets were in common use then as reusable writing surfaces, sometimes used for shorthand.

In his work Polygraphiae, Johannes Trithemius developed his so-called "Ave-Maria-Cipher" that can hide information in a Latin praise of God. "Auctor Sapientissimus Conseruans Angelica Deferat Nobis Charitas Potentissimi Creatoris" for example contains the concealed word VICIPEDIA.

.

2. CHAPTER

WHAT IS STEGANOGRAPHY?

Much of the information communicated on a daily basis must be kept confidential. Information such as financial reports, employee data and medical records needs to be communicated in a way that ensures confidentiality and integrity. This makes good business sense and may even be regulated by legislation like the Health Insurance Portability and Accountability Act (HIPAA). The problem of unsecure communication is compounded by the fact that much of this information is sent over the public Internet and may be processed by third parties, as in
e-mail or instant messaging (IM).

2.1 Steganography Basics

Steganography aims to hiding information in a cover data in such a way that non-participatingpersons are not able to detect the presence of this information by analyzing the information detection. Unlike watermarking, steganography does not intended to prevent the hidden in- formation by opponents of removing or changing the hidden message, which is embedded in the cover data but it emphasizes on remains it undetectable. Steganography is particularly interesting for applications in which the encryption can not used to protect the
confidentiality of communication.

2.2 Steganography with LSB Algorithm

Bytes of pixels are sufficient to hold one message byte. Rest of the bits in the pixels remains the same. Steganography is the art and science of communicating in a way which hides the existence of the communication. Steganography plays an important role in information security. It is the art of invisible communication by concealing information inside other information. The term

steganography is derived from Greek and literally means covered writing. A Steganography system consists of three elements: coverimage (which hides the secret message), the secret message and the stegano-image (which is the cover object with message embedded inside it). A digital image is described using a 2-D matrix of the color intestines at each grid point (i.e. pixel). Typically gray images use 8 bits, whereas colored utilizes 24 bits to describe the color model, such as RGB model. The Steganography system which uses an image as the cover, there are several techniques to conceal information inside cover-image. The spatial domain techniques manipulate the cover-image pixel bit values to embed the secret information. The secret bits are written directly to the cover image pixel bytes. Consequently, the spatial domain techniques are simple and easy to implement. The Least Significant Bit (LSB) is one of the main techniques in spatial domain image Steganography.

The concept of LSB Embedding is simple. It exploits the fact that the level of precision in many image formats is far greater than that perceivable by average human vision. Therefore, an altered image with slight variations in its colors will be indistinguishable from the original by a human being, just by looking at it. In conventional LSB technique, which requires eight bytes of pixels to store 1byte of secret data but in proposed LSB technique.

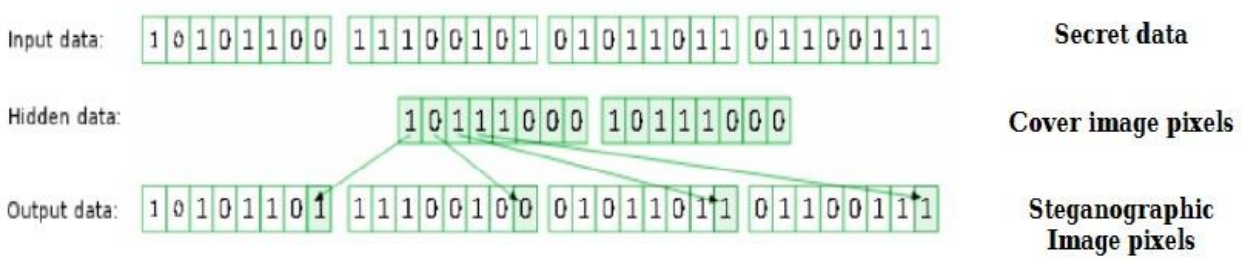


Fig 1.1

3. CHAPTER

REQUIREMENT ANALYSIS

3.1 Functional Requirements

Functional requirements are the requirements that that define specific behavior or function of the system.

•Ask User : It will ask user whether he/she wants to encode or decode the image .

If user chooses Encode:

•Secret Text Message : In this you will have to write secret message to hide.

If user chooses Decode:

Stego Encryption LSB implementation is performed on cover image to hide secret text message by replacing bits of cover image by the bits of message.

•Message Display : Message is displayed on the terminal.

3.2 Software Requirements

- Operating System : Windows 7 or above
- Programming Language : Python
- Python Libraries : Numpy and Pillow
- Python IDE : VS Code , Jupyter etc.

4. CHAPTER

Types of Steganography

Depending on the nature of the cover object (actual object in which secret data is embedded), steganography can be divided into five types:

1. Text Steganography
2. Image Steganography
3. Video Steganography
4. Audio Steganography
5. Network Steganography

4.1 Text Steganography

Text Steganography is hiding information inside the text files. It involves things like changing the format of existing text, changing words within a text, generating random character sequences or using context-free grammars to generate readable texts. Various techniques used to hide the data in the text are:

- Format Based Method
- Random and Statistical Generation
- Linguistic Method

4.2 Image Steganography

Hiding the data by taking the cover object as the image is known as image steganography. In digital steganography, images are widely used cover source because there are a huge number of bits present in the digital representation of an image. There are a lot of ways to hide information inside an image. Common approaches include:

- Least Significant Bit Insertion
- Masking and Filtering
- Redundant Pattern Encoding
- Encrypt and Scatter
- Coding and Cosine Transformation

4.3 Video Steganography

In Video Steganography you can hide kind of data into digital video format. The advantage of this type is a large amount of data can be hidden inside and the fact that it is a moving stream of images and sounds. You can think of this as the combination of Image Steganography and Audio Steganography. Two main classes of Video Steganography include:

- Embedding data in uncompressed raw video and compressing it later
- Embedding data directly into the compressed data stream

4.4 Audio Steganography

In audio steganography, the secret message is embedded into an audio signal which alters the binary sequence of the corresponding audio file. Hiding secret messages in digital sound is a much more difficult process when compared to others, such as Image Steganography. Different methods of audio steganography include:

- Least Significant Bit Encoding
- Parity Encoding
- Phase Coding
- Spread Spectrum

This method hides the data in WAV, AU, and even MP3 sound files.

4.5 Network Steganography

It is the technique of embedding information within network control protocols used in data transmission such TCP, UDP, ICMP etc. You can use steganography in some covert channels that you can find in the OSI model. For Example, you can hide information in the header of a TCP/IP packet in some fields that are either optional.
In today's digitalized world, various software tools are available for Steganography. In the remainder of this Steganography Tutorial, we will explore some of the popular steganographic tools and their capabilities.

5. CHAPTER

How It Works

5.1 Implementation

- Importing image module from Pillow(PIL) library
  Importing Numpy

```
from PIL import Image
import numpy as np
```

- Creating a Class named “ImagSteg”:

```
class ImagSteg:
```

- Creating a function named “encrypt_text_in_image” : It will take image path , message to enter and the target path added with folder name that I have created named “static”. Then we converted the image to array format and flattened it. Then I added the message with a word which would be useful later. Following it we call an function “__fillMSB” that gives me the binary value of my message letter by letter in array format. Then I have encrypted the image by replacing it each least significant bit of image pixel binary value with my message bits one by one. Then we formed a variable to save the path with and added extension in name to change the name from original one and after this I have reshaped it and saved the image and returned it to the calling area.

```
def encrypt_text_in_image(self, image_path, msg, target_path=""):

  img = np.array(Image.open(image_path))
  imgArr = img.flatten()
  msg += "<-END->"
  msgArr = [self.__fillMSB(bin(ord(ch))) for ch in msg]
  idx = 0
  for char in msgArr:
    for bit in char:
      if bit==1:
        if imgArr[idx]==0:
          imgArr[idx] = 1
        else:
          imgArr[idx] = imgArr[idx] if imgArr[idx]%2==1 else imgArr[idx]-1
      else:
        if imgArr[idx]==255:
          imgArr[idx] = 254
        else:
          imgArr[idx] = imgArr[idx] if imgArr[idx]%2==0 else imgArr[idx]+1
      idx += 1
  savePath = target_path+ image_path.split(".")[0] + "_encrypted.png"

  resImg = Image.fromarray(np.reshape(imgArr, img.shape))
  resImg.save(savePath)
  return
```

- Creating the "__fillMSB" function : It takes the binary value of the string that is entered by the user and converts it in array from and returns to the calling function.

```
def __fillMSB(self, inp):
  '''
  0b01100 -> [0,0,0,0,1,1,0,0]
  '''
  inp = inp.split("b")[-1]
  inp = '0'*(7-len(inp))+inp
  return [int(x) for x in inp]
```

- Creating the function “decrypt_text_in_image” which decrypt the image in which the message is hidden i.e Stego Image . If we get the least significant bits of each 7 pixels we will get a letter of our word . We used a for loop of step size that picks up 7 pixels at once from the image array and sends it to a function where the LSB is extracted and change to the character that the 7 pixels taken forms. The we go on appending the letters one by one to form the message.As soon as we find the message appended while encrypting the message we end it and return the original message.

```
def decrypt_text_in_image(self, image_path,target_path=""):
  '''
  Read image -> Extract Text -> RetuC:/Users/souma/AppData/Local/Programs/Python/Python310/rn
  Read image -> Extract Text -> RetuC:/Users/souma/AppData/Local/Programs/Python/Python310/rn
  '''
  img = np.array(Image.open(image_path))
  imgArr = np.array(img).flatten()

  decrypted_message = ""
  for i in range(7,len(imgArr),7):
    decrypted_char = self.__decrypt_pixels(imgArr[i-7:i])
    decrypted_message += decrypted_char

    if len(decrypted_message)>10 and decrypted_message[-7:] == "<-END->":
      break
  return decrypted_message[:-7]
```

- We create the “__decrypt_pixels” function that Join seven 0/1s to form binary then integer and from that ASCII value it forms characters.

```
def __decrypt_pixels(self, pixels):
  '''
  Given list of 7 pixel values -> Determine 0/1 -> Join 7 0/1s to form binary -> integer -> character
  '''

  pixels = [str(x%2) for x in pixels]
  bin_repr = "".join(pixels)
  return chr(int(bin_repr,2))

def encrypt_text_in_image(self, image_path, msg, target_path=""):
```

- Now we create another python file where we call the class from the different file and then go on to call the functions to encode and decode the picture as instructed by the user .

```python
from numpy.core.arrayprint import printoptions
from test1 import ImagSteg

img = ImagSteg()

choice = input("enter 1 to encode //// Enter 0 to decode \t")
if choice == "1":
    # msginput=input("enter a msg")
    # img.encrypt_text_in_image("exp1.jpg",msginput,"static/")
    # img.encrypt_text_in_image("exp2.png",msginput,"static/")
    typeofimage = input(
        "\n You Chose to encode the image. \n 1.for jpg type image type 'jpg' \n 2.for png type image type 'png'\n")

    match typeofimage:

        case "jpg":
            msginput = input("enter a msg \n")
            print(
                "\n you chose JPG format plss follow the instructions below and proceed  as told")
            imagename = input(
                "\n copy and paste the image in the project file location and then type just the name of the image with .jpg extension For exaple: test1.jpg \n")
            img.encrypt_text_in_image(imagename, msginput, "static/")
        case "JPG":
            msginput = input("enter a msg")
            print(
                "\n you chose JPG format plss follow the instructions below and proceed  as told")
            imagename = input(
                "\n copy and paste the image in the project file location and then type just the name of the image with .jpg extension For exaple: test1.jpg \n")

            img.encrypt_text_in_image(imagename, msginput, "static/")

        case "png":
            msginput = input("enter a msg")
            print(
                "\n you chose PNG format plss follow the instructions below and proceed  as told")
            imagename = input(
                "\n copy and paste the image in the project file location and then type just the name of the image with .png extension For exaple: test1.png \n")
            img.encrypt_text_in_image(imagename, msginput, "static/")
```

```python
        case "PNG":
            msginput = input("enter a msg")
            print(
                "\n you chose PNG format plss follow the instructions below and proceed  as told")
            imagename = input(
                "\n copy and paste the image in the project file location and then type just the name of the image with .png extension For exaple: test1.png \n")
            img.encrypt_text_in_image(imagename, msginput, "static/")

        case _:
            print("\n Image type not supported")


elif choice == "0":
    typeofimage = input(
        "\n You Chose to decode the image. \n 1.for jpg type image type 'jpg' \n 2.for png type image type 'png' \n")

    match typeofimage:

        case "jpg":
            print(
                "\n you chose JPG format plss follow the instructions below and proceed  as told")
            imagename = input(
                "\n copy and paste the image in the project file location and then type just the name of the image and add '_encrypted.jpg' extension For exaple: test1_encrypted.jpg \n")
            imagename = "static/"+imagename
            res = img.decrypt_text_in_image(imagename)
            print(res)
        case "JPG":
            print(
                "\n you chose JPG format plss follow the instructions below and proceed  as told")
            imagename = input(
                "\n copy and paste the image in the project file location and then type just the name of the image and add '_encrypted.jpg' extension For exaple: test1_encrypted.jpg \n")
            imagename = "static/"+imagename
            res = img.decrypt_text_in_image(imagename)
            print(res)

        case "png":
            print(
                "\n you chose JPG format plss follow the instructions below and proceed  as told")
            imagename = input(
                "\n copy and paste the image in the project file location and then type just the name of the image and add '_encrypted.jpg' extension For exaple: test1_encrypted.jpg \n")
            imagename = "static/"+imagename
            res = img.decrypt_text_in_image(imagename)
            print(res)
```

```
        case "PNG":
            print(
                "\n you chose JPG format plss follow the instructions below and proceed  as told")
            imagename = input(
                "\n copy and paste the image in the project file location and then type just the name of the image and add '_encrypted.jpg' extension For exaple: test1_encrypted.jpg \n")
            imagename = "static/"+imagename
            res = img.decrypt_text_in_image(imagename)
            print(res)

        case _:
            print("\n Image type not supported")
else:
    print("wrong input!! start again")
```

RESULTS & DISCUSSION

JPG type test image

Figure 1.2

JPG type encrypted image

Figure 1.3

PNG type test image

Figure 1.4

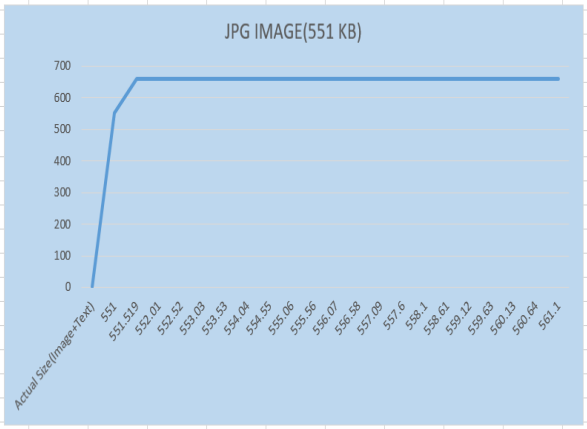
PNG type encrypted image

Figure 1.5

I have encrypted 2600 words of 3 letters and here are the results of the above images:

Graph analyzation

Image size	Text Size	Actual Size(Image+Text	Size of Encrypted Imag
551	0	551	551
551	0.519	551.519	660
551	1.01	552.01	660
551	1.52	552.52	660
551	2.03	553.03	660
551	2.53	553.53	660
551	3.04	554.04	660
551	3.55	554.55	660
551	4.06	555.06	660
551	4.56	555.56	660
551	5.07	556.07	660
551	5.58	556.58	660
551	6.09	557.09	660
551	6.6	557.6	660
551	7.1	558.1	660
551	7.61	558.61	660
551	8.12	559.12	660
551	8.63	559.63	660
551	9.13	560.13	660
551	9.64	560.64	660
551	10.1	561.1	660

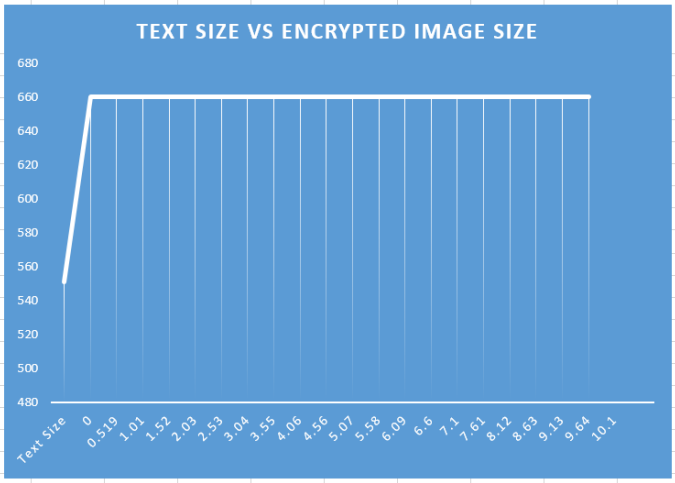


ENCRYPTED IMAGE SIZE

ACTUAL IMAGE(IMAGE+TEXT)

Figure 1.6

Text Size	Size of Encrypted Image
0	551
0.519	660
1.01	660
1.52	660
2.03	660
2.53	660
3.04	660
3.55	660
4.06	660
4.56	660
5.07	660
5.58	660
6.09	660
6.6	660
7.1	660
7.61	660
8.12	660
8.63	660
9.13	660
9.64	660
10.1	660



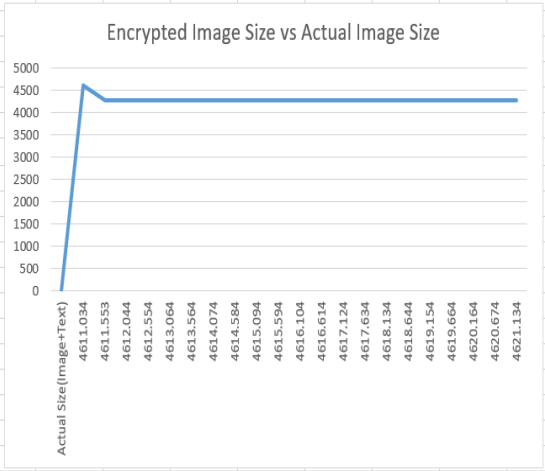
ENCRYPTED IMAGE SIZE

TEXT SIZE

Figure 1.7

JPG type picture analysis

Image size	Text Size	Actual Size(Image+Text	Encrypted Image Siz
4611.034	0	4611.034	4611.034
4611.034	0.519	4611.553	4276.991
4611.034	1.01	4612.044	4277.126
4611.034	1.52	4612.554	4277.343
4611.034	2.03	4613.064	4277.37
4611.034	2.53	4613.564	4277.467
4611.034	3.04	4614.074	4277.461
4611.034	3.55	4614.584	4277.509
4611.034	4.06	4615.094	4277.507
4611.034	4.56	4615.594	4277.498
4611.034	5.07	4616.104	4277.492
4611.034	5.58	4616.614	4277.424
4611.034	6.09	4617.124	4277.527
4611.034	6.6	4617.634	4277.668
4611.034	7.1	4618.134	4277.967
4611.034	7.61	4618.644	4278.04
4611.034	8.12	4619.154	4278.009
4611.034	8.63	4619.664	4277.964
4611.034	9.13	4620.164	4278.032
4611.034	9.64	4620.674	4277.977
4611.034	10.1	4621.134	4276.991

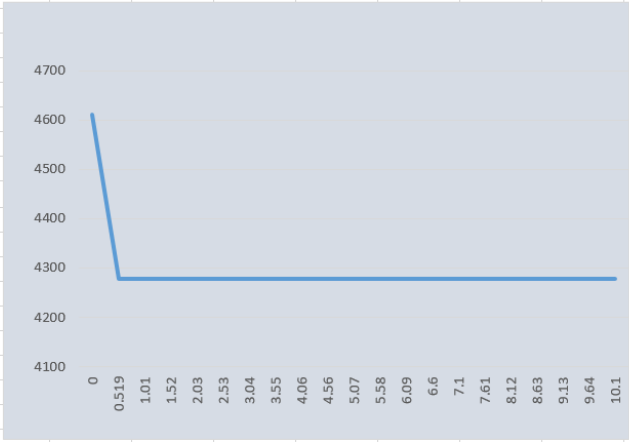


Encrypted Image Size

Actual Image Size

Figure 1.8

Text Size	Encrypted Image Size
0	4611.034
0.519	4276.991
1.01	4277.126
1.52	4277.343
2.03	4277.37
2.53	4277.467
3.04	4277.461
3.55	4277.509
4.06	4277.507
4.56	4277.498
5.07	4277.492
5.58	4277.424
6.09	4277.527
6.6	4277.668
7.1	4277.967
7.61	4278.04
8.12	4278.009
8.63	4277.964
9.13	4278.032
9.64	4277.977
10.1	4276.991



Encrypted image size

text size

Figure 1.9

PNG type picture analysis

As we can see from the charts and above given graph we can see that there is a sudden change in the image size by few killo-bytes on its first word encryption but after that the change is almost constant.

In case of out PNG type picture the image size reduces from 4611.034 kb to 4276.991 kb and then as we go on adding words upto 2600 the changes are in few bytes only and even after encrypting such a large text file one can not identify at first sight that such a large number of words are hidden inside it.

In case of JPG type picture the image size increases from 551 kb to 660 kb for it's first word encryption and after that change is almost constant.

CONCLUSION

It is observed that through LSB Substitution Steganographic method, the results obtained in data hiding are pretty impressive as it utilizes the simple fact that any image could be broken up to individual bit-planes each consisting of different levels of information. It is to be noted that as discussed earlier, this method is only effective for bitmap images as these involve lossless compression techniques. But this process can also be extended to be used for color images where, bitplane slicing is to be done individually for the top four bit-planes for each of R, G, B of the message image.

It is also important to discuss that though steganography was once undetected, with the various methods currently used, it is not only easy to detect the presence but also retrieving them is easier. For instance, without having to use a software or complex tools for detection, simple methods to observe if an image file has been manipulated are: 1. Size of the image: A Steganographic image has a huge storage size when compared to a regular image of the same dimensions. I.e. if the original image storage size would be few KBs, the Steganographic image could be several MBS in size. This again varies with the resolution and type of image used. 2. Noise in image: A Steganographic image has noise when compared to a regular image. This is the reason why initially little noise is added to the cover image, so that the Steganographic image doesnt appear very noisy when compared to the original cover image.Further our project is just at the first stage of building a secure encryption through which we can encrypt messages and other can not decode it easily.

APPENDIX

1.Main programme where all the works are performed:

```
from PIL import Image
import numpy as np

class ImagSteg:

  def __fillMSB(self, inp):
    '''
    0b01100 -> [0,0,0,0,1,1,0,0]
    '''
    inp = inp.split("b")[-1]
    inp = '0'*(7-len(inp))+inp
    return [int(x) for x in inp]

  def __decrypt_pixels(self, pixels):
    '''
    Given list of 7 pixel values -> Determine 0/1 -> Join 7 0/1s to form binary -> integer ->
character
    '''

    pixels = [str(x%2) for x in pixels]
    bin_repr = "".join(pixels)
    return chr(int(bin_repr,2))

  def encrypt_text_in_image(self, image_path, msg, target_path=""):

    img = np.array(Image.open(image_path))
    imgArr = img.flatten()
    msg += "<-END->"
    msgArr = [self.__fillMSB(bin(ord(ch))) for ch in msg]
    idx = 0
    for char in msgArr:
      for bit in char:
        if bit==1:
          if imgArr[idx]==0:
            imgArr[idx] = 1
          else:
            imgArr[idx] = imgArr[idx] if imgArr[idx]%2==1 else imgArr[idx]-1
        else:
          if imgArr[idx]==255:
```

```
            imgArr[idx] = 254
          else:
            imgArr[idx] = imgArr[idx] if imgArr[idx]%2==0 else imgArr[idx]+1
        idx += 1
    savePath = target_path+ image_path.split(".")[0] + "_encrypted.png"

    resImg = Image.fromarray(np.reshape(imgArr, img.shape))
    resImg.save(savePath)
    return

  def decrypt_text_in_image(self, image_path,target_path=""):
    '''
    Read image -> Extract Text ->
RetuC:/Users/souma/AppData/Local/Programs/Python/Python310/rn
    Read image -> Extract Text ->
RetuC:/Users/souma/AppData/Local/Programs/Python/Python310/rn
    '''
    img = np.array(Image.open(image_path))
    imgArr = np.array(img).flatten()

    decrypted_message = ""
    for i in range(7,len(imgArr),7):
      decrypted_char = self.__decrypt_pixels(imgArr[i-7:i])
      decrypted_message += decrypted_char

      if len(decrypted_message)>10 and decrypted_message[-7:] == "<-END->":
        break
    return decrypted_message[:-7]
            imgArr[idx] = 254
          else:
            imgArr[idx] = imgArr[idx] if imgArr[idx]%2==0 else imgArr[idx]+1
        idx += 1
    savePath = target_path+ image_path.split(".")[0] + "_encrypted.png"

    resImg = Image.fromarray(np.reshape(imgArr, img.shape))
    resImg.save(savePath)
    return

  def decrypt_text_in_image(self, image_path,target_path=""):
    '''
    Read image -> Extract Text ->
RetuC:/Users/souma/AppData/Local/Programs/Python/Python310/rn
    Read image -> Extract Text ->
RetuC:/Users/souma/AppData/Local/Programs/Python/Python310/rn
    '''
```

```
    img = np.array(Image.open(image_path))
    imgArr = np.array(img).flatten()

    decrypted_message = ""
    for i in range(7,len(imgArr),7):
      decrypted_char = self.__decrypt_pixels(imgArr[i-7:i])
      decrypted_message += decrypted_char

      if len(decrypted_message)>10 and decrypted_message[-7:] == "<-END->":
        break
    return decrypted_message[:-7]
```

2 . Code which calls this class to perform the actions:

```
from numpy.core.arrayprint import printoptions
from test1 import ImagSteg

img = ImagSteg()

choice = input("enter 1 to encode //// Enter 0 to decode \t")
if choice == "1":
    # msginput=input("enter a msg")
    # img.encrypt_text_in_image("exp1.jpg",msginput,"static/")
    # img.encrypt_text_in_image("exp2.png",msginput,"static/")
    typeofimage = input(
        "\n You Chose to encode the image. \n 1.for jpg type image type 'jpg' \n
2.for png type image type 'png'\n")

    match typeofimage:

        case "jpg":
            msginput = input("enter a msg \n")
            print(
                "\n you chose JPG format plss follow the instructions below and
proceed  as told")
            imagename = input(
                "\n copy and paste the image in the project file location and
then type just the name of the image with .jpg extension For exaple: test1.jpg
\n")
            img.encrypt_text_in_image(imagename, msginput, "static/")
        case "JPG":
            msginput = input("enter a msg")
            print(
```

```
                "\n you chose JPG format plss follow the instructions below and
proceed  as told")
            imagename = input(
                "\n copy and paste the image in the project file location and
then type just the name of the image with .jpg extension For exaple: test1.jpg
\n")

            img.encrypt_text_in_image(imagename, msginput, "static/")

        case "png":
            msginput = input("enter a msg")
            print(
                "\n you chose PNG format plss follow the instructions below and
proceed  as told")
            imagename = input(
                "\n copy and paste the image in the project file location and
then type just the name of the image with .png extension For exaple: test1.png
\n")
            img.encrypt_text_in_image(imagename, msginput, "static/")

        case "PNG":
            msginput = input("enter a msg")
            print(
                "\n you chose PNG format plss follow the instructions below and
proceed  as told")
            imagename = input(
                "\n copy and paste the image in the project file location and
then type just the name of the image with .png extension For exaple: test1.png
\n")
            img.encrypt_text_in_image(imagename, msginput, "static/")

        case _:
            print("\n Image type not supported")


elif choice == "0":
    typeofimage = input(
        "\n You Chose to decode the image. \n 1.for jpg type image type 'jpg' \n
2.for png type image type 'png' \n")

    match typeofimage:

        case "jpg":
            print(
```

```
                "\n you chose JPG format plss follow the instructions below and
proceed  as told")
            imagename = input(
                "\n copy and paste the image in the project file location and
then type just the name of the image and add '_encrypted.jpg' extension For
exaple: test1_encrypted.jpg \n")
            imagename = "static/"+imagename
            res = img.decrypt_text_in_image(imagename)
            print(res)
        case "JPG":
            print(
                "\n you chose JPG format plss follow the instructions below and
proceed  as told")
            imagename = input(
                "\n copy and paste the image in the project file location and
then type just the name of the image and add '_encrypted.jpg' extension For
exaple: test1_encrypted.jpg \n")
            imagename = "static/"+imagename
            res = img.decrypt_text_in_image(imagename)
            print(res)

        case "png":
            print(
                "\n you chose JPG format plss follow the instructions below and
proceed  as told")
            imagename = input(
                "\n copy and paste the image in the project file location and
then type just the name of the image and add '_encrypted.jpg' extension For
exaple: test1_encrypted.jpg \n")
            imagename = "static/"+imagename
            res = img.decrypt_text_in_image(imagename)
            print(res)

        case "PNG":
            print(
                "\n you chose JPG format plss follow the instructions below and
proceed  as told")
            imagename = input(
                "\n copy and paste the image in the project file location and
then type just the name of the image and add '_encrypted.jpg' extension For
exaple: test1_encrypted.jpg \n")
            imagename = "static/"+imagename
            res = img.decrypt_text_in_image(imagename)
            print(res)
```

```
        case _:
            print("\n Image type not supported")
else:
    print("wrong input!! start again")
```

BIBLIOGRAPHY

Ramadhan J.Mstafa , University of Zakho & Christian Bach , Univerity of Bridgeport (2013) . Information Hiding in Images Using Steganography Techniques.

Mohammed A.Saleh , College of Sciences and Arts in Ar Rass, Qassim University, Kingdom of Saudi Arabia(2018). Image Steganography Techniques – A Review Paper

Dr. Amarendra K, Venkata Naresh Mandhala, B.Chetan gupta, G.Geetha Sudheshna, V.Venkata Anusha . Image Steganography Using LSB.