A MINOR PROJECT REPORT ON

IMAGE STEGANOGRAPHY
USING PYTHON AND FLASK

SUBMITTED IN PARTIAL FULFILMENT FOR THE AWARD OF DEGREE OF

BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted By:-

Kushagra Gupta (9917102203)

Isha Srivastava (9917102127)

Mukul Chauhan (9917102220)

Under the supervision of:

Dr. Kapil Dev Tyagi

Department of ECE

Jaypee Institute of Information Technology, Noida

June, 2020

CERTIFICATE

This is to certify that the minor project report entitled, “IMAGE STEGANOGRAPHY USING PYTHON AND FALSK” submitted by **Kushagra Gupta, Isha Srivastava** and **Mukul Chauhan** in partial fulfilment for the award of bachelor of technology degree in electronics and communication engineering of the Jaypee Institute Of Information Technology, Noida is an authentic work carried out by them under my supervision and guideline. The matter embodied in the report is original and has not been submitted for the award of any other degree.

Signature:
Name: Dr. Kapil Dev Tyagi
ECE department
JIIT, sec 128
Noida-201304

Date: 5th June, 2020

DECLARATION

We hereby declare that this written submission represents our own ideas in our own words and where other's ideas or words have been included, have been adequately cited and referred to the original source. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or bricked or falsified the idea/data/fact/source not submission.

Place: Jaypee Institute Of Information Technology, Noida

Date: 5th June, 2020

Name: Kushagra Gupta

Enrollment:9917102203

Name: Isha Srivastava

Enrollment:9917102127

Name: Mukul Chauhan

Enrollment:9917102220

ACKNOWLEDGEMENT

The completion of any inter-disciplinary project depends upon cooperation, coordination and combined efforts of several sources of knowledge. We are grateful to our mentor, **Dr. Kapil Dev Tyagi** for his constant motivation and willingness to give us valuable advice and direction whenever we approached him with a problem. We are thankful to him for providing us with immense guidance for this project. We have been greatly benefited by his guidance on the subject of **"image processing"**. We are highly obliged to him for giving us the opportunity to present a project on a topic of our choice and interest.

We would also like to thank our college authorities for giving us the opportunity to pursue our project in this field and helping us to successfully complete the project

ABSTRACT

Steganography is the science of writing hidden messages in such a way that no one except the sender and the receiver will be able to read. The advantage of steganography over cryptography alone is that the intended secret message does not attract attention to itself as an object of scrutiny. Plainly visible encrypted messages, no matter how unbreakable they are, arouse interest and may in themselves be incriminating in countries in which encryption is illegal.
Steganography includes the concealment of information within computer files. In digital steganography, electronic communications may include steganographic coding inside of a transport layer, such as a document file, image file, program or protocol. Media files are ideal for stegano- graphic transmission because of their large size. For example, a sender might start with an innocuous image file and adjust the colour of every hundredth pixel to correspond to a letter in the alphabet. The change is so subtle that someone who is not specifically looking for it is unlikely to notice the change.

This project is developed for hiding information in any image file. The scope of the project is implementation of steganography tools for hiding information includes any type of information file and image files and the path where the user wants to save Image and extruded file.

INDEX

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

The word **Steganography** is derived from two Greek words- ‘stegos’ meaning ‘to cover’ and ‘grayfia’, meaning ‘writing’, thus translating to ‘covered writing’, or ‘hidden writing’. **Steganography** is a method of hiding secret data, by embedding it into an audio, video, image or text file. It is one of the methods employed to protect secret or sensitive data from malicious attacks.

Cryptography and steganography are both methods used to hide or protect secret data. However, they differ in the respect that cryptography makes the data unreadable, or hides the meaning of the data, while steganography hides the existence of the data.

Image Steganography refers to the process of hiding data within an image file. The image selected for this purpose is called the **cover-image** and the image obtained after steganography is called the **stego-image**.

The basic terminologies used in the steganography systems are: the cover message, secret message, the secret key and embedding algorithm**.** The cover message is the carrier of the message such as image, video, audio, text or some other digital media. The secret message is the information which is needed to be hidden in the suitable digital media. The secret key is usually used to embed the message depending on the hiding algorithms. The embedding algorithm is the way or the idea that usually used to embed the secret information in the cover message**.**

1.1 Problem Statement

In this digital era, the intensive collection of data and the inherent advantages of the new technology have spawned the cynical idea that privacy is dead. Moreover, recent instances of information leak by our beloved social networking sites put us at the risk of privacy breach. This image steganography web application is designed to relieve users of this data breach risk and instill a feeling of security amongst them. The advantage of steganography over cryptography alone is that the intended secret message does not attract attention to itself as an object of scrutiny.

1.2 Objectives

Security and Privacy of the users: Steganography hides the existence of the data and thus protects
the sensitive data from malicious attacks.

Data encoding: Hiding the text into an image’s pixels.

Data Decoding: Displaying the hidden text from the image.

1.3 Software Requirements

- Python
- Flask
- HTML
- CSS
- Bootstrap

1.4 Advantages and Disadvantages of Steganography

Everything has its pros and cons, so as Steganography.

1.4.1 Advantages

- It can be used for safeguarding data, such as in the field of media where copywriting ensures authenticity.
- It can be used by intelligence agencies for sending their secret data.
- Used to ensure that a third party has not tampered with a sent message. This is accomplished by creating a hash of the message using a fixed character length for every item in the message, when the original items are in fact of variable character length. The hash is encrypted and sent with the message. When the recipient receives the message it is decoded. If the hash from the decoded message does not match the hash from the encrypted message, both the sender and recipient of the message know that it has been tampered with.

1.4.2 Disadvantages

Unfortunately most uses of steganography and research around the topic of steganography centre around the illegitimate purposes. The three biggest areas of illegitimate steganography evolve around terrorism, pornography and data theft.

The illegitimate uses of steganography were also found to be on a global scale, involved national security or were done on an academic basis in order to better understand the potential danger of steganography if created by individuals with ill-intentions.

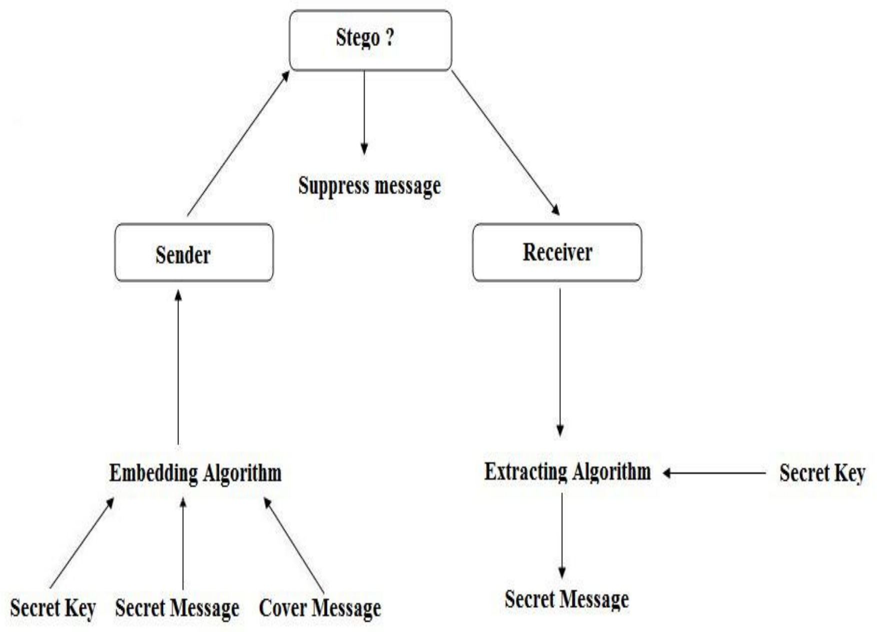


Figure 1.1 General Steganographic Approach

Steganography is the art and science of communicating in a way which hides the existence of the communication. In contrast to cryptography, where the "enemy" is allowed to detect, intercept and modify messages without being able to violate certain security premises guaranteed by a cryptosystem, the goal of steganography is to hide messages inside other "harmless" messages in a way that does not allow any "enemy" to even detect that there is a second secret message present. Steganography is in the (especially military) literature also referred to as transmission security or short TRANSEC.

CHAPTER 2

LITERATURE REVIEW

The network security is becoming indispensable as the volume of data being exchanged over the Internet is escalating day by day. The two important techniques for providing security to information or data being shared are cryptography and steganography. Both are well known and widely used methods in information security.

In a large number of applications, it is desirable that the secrecy of the communicated data or information is maintained. Such secret communication ranges from the apparent cases of bank transfers, corporate communications, and credit card purchases, on down to a large percentage of everyday email. Steganography is an ancient art of embedding a secret message (data embedding) [4] into a seemingly benign message. Most of the newer applications use steganography, for instance, a watermark to protect a copy right on information. The forms of steganography vary, but unsurprisingly, benign spam messages are turning up more often containing embedded text.

This section should situate the team research, which is needed to focus on the wider academic community in the text to image steganography and to identify the gap within that the literature that the research will need to address. The main purpose of literature review is that combine with understanding of each work, point that in which way could fulfilling the need for other research, and located the team own design in the background of existing literature is the most significant point.

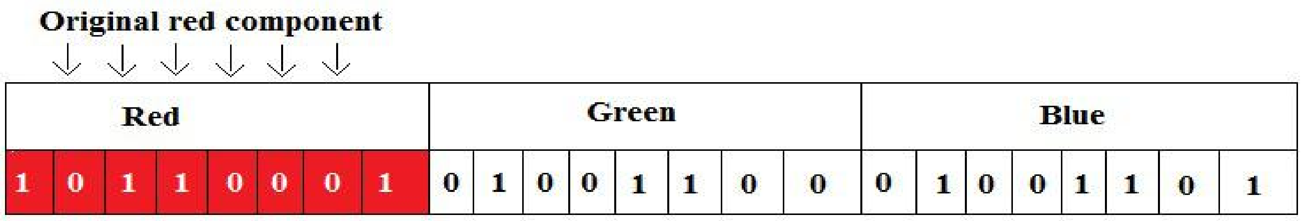
2.1 Research Papers

Research Paper 1: A New Approach To Hide Text in Images Using Steganography

This paper proposes a new algorithm to hide data inside an image using steganographic technique. The algorithm proposed is an enhanced version of LSB technique, that is not very much robust. Also it has implemented a compression technique to increase the hiding capacity. The algorithm proposed in this system is basically an extension of the original LSB which is quite vulnerable. Instead of hiding the data in least significant bits of the RGB components of a pixel, we in this algorithm, would be hiding data as shown

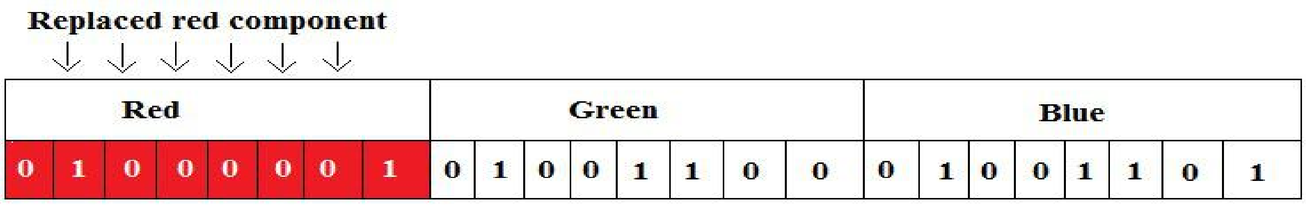
Here its assumed that the text to be hidden here is “ABC”

Convert the ascii value of each character into 8 bit binary value and then replace it with the LSB values of the RGB values.


Original red component

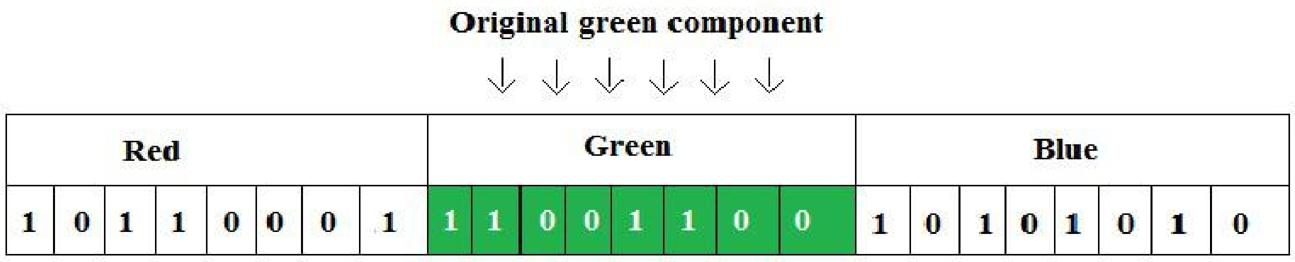
Red								Green								Blue							
1	0	1	1	0	0	0	1	0	1	0	0	1	1	0	0	0	1	0	0	1	1	0	1

Figure 2.1 Original value of red component


Replaced red component

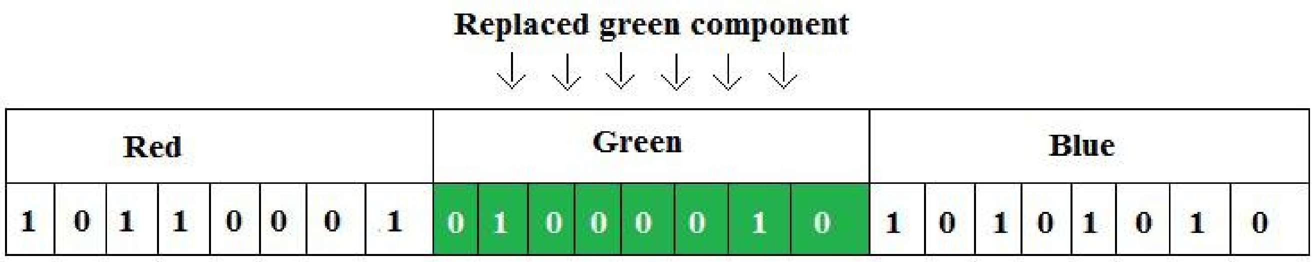
Red								Green								Blue							
0	1	0	0	0	0	0	1	0	1	0	0	1	1	0	0	0	1	0	0	1	1	0	1

Figure 2.2 Red component is replaced with binary of 65 i.e. A.


Original green component

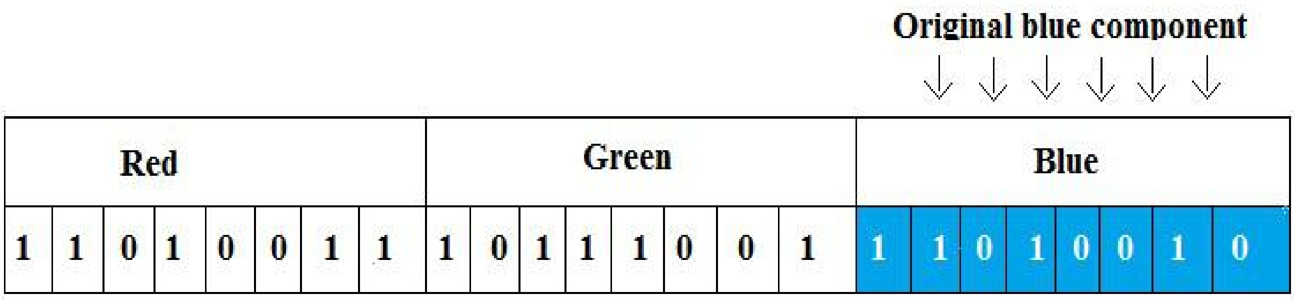
Red								Green								Blue							
1	0	1	1	0	0	0	1	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0

Figure 2.3 Original Value of green component


Replaced green component

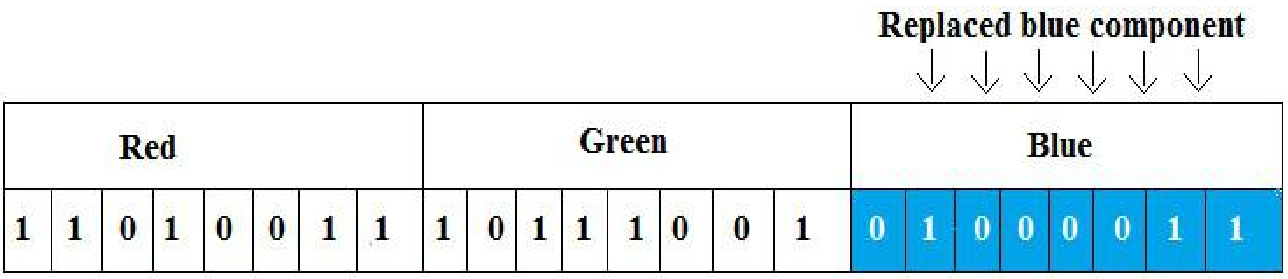
Red								Green								Blue							
1	0	1	1	0	0	0	1	0	1	0	0	0	0	1	0	1	0	1	0	1	0	1	0

Figure 2.4 Green component of second pixel is replaced with binary of 66 (B)


Original blue component

Red								Green								Blue							
1	1	0	1	0	0	1	1	1	0	1	1	1	0	0	1	1	1	0	1	0	0	1	0

Figure 2.5 Original value of blue component


Replaced blue component

Red								Green								Blue							
1	1	0	1	0	0	1	1	1	0	1	1	1	0	0	1	0	1	0	0	0	0	1	1

Figure 2.6 Blue component of third pixel is replaced with binary of 67 i.e. C.

Conclusion : This paper proposed a new steganographic algorithm for hiding text files in images. Here they have also used an underlying compression algorithm with maximum compression ratio of 8 bits/ pixel. They have developed a system in java based on the proposed algorithm. Here they have tested few images with different sizes of text files to be hidden and concluded that the resulting stego images do not have any noticeable changes. Also its found that for .bmp images this algorithm works very efficiently. Hence this new steganographic approach is robust and very efficient for hiding text files in images.

Research Paper 2: An Overview Of Image Steganography

The most popular algorithms for image steganography are discussed and compared in this paper. After the overview it briefly reflects on the suitability of various image steganography techniques for various applications. This reflection is based on a set of criteria that the authors identified for image steganography. Further, the steganographic algorithms are explained in categories according to image file formats and the domain in which they are performed. In the domain of digital images many different image file formats exist, most of them for specific applications. All color variations for the pixels of a 24-bit image are derived from three primary colors: red, green and blue, and each primary color is represented by 8 bits. For these different image file formats, different steganographic algorithms exist. For this reason, LSB steganography has also been developed for use with other image file formats. In image steganography the information is hidden exclusively in images. The JPEG file format is the most popular image file format on the Internet, because of the small size of the images. In image steganography as stated earlier, images are the most popular cover objects used for steganography. In spread spectrum image steganography the message is embedded in noise and then combined with the cover image to produce the stego-image. LSB and Palette Based Images Palette based images, for example GIF images, are another popular image file format commonly used on the Internet. These technologies are mainly concerned with the protection of intellectual property, thus the algorithms have different requirements than steganography. Internet and all the different digital file formats that is has decreased in importance. Image steganography techniques can be divided into two groups: those in the Image Domain and those in the Transform Domain.

<u>Conclusion :</u> Although only some of the main image steganographic techniques were discussed in this paper, one can see that there exists a large selection of approaches to hiding information in images. All the major image file formats have different methods of hiding messages, with different strong and weak points respectively. Where one technique lacks in payload capacity, the other lacks in robustness. For example, the patchwork approach has a very high level of robustness against most type of attacks, but can hide only a very small amount of information. Least significant bit (LSB) in both BMP and GIF makes up for this, but both approaches result in suspicious files that increase the probability of detection when in the presence of a warden.

Thus for an agent to decide on which steganographic algorithm to use, he would have to decide on the type of application he want to use the algorithm for and if he is willing to compromise on some features to ensure the security of others.

2.2 Common Issues

The biggest problem steganography faces is that of size. There is a limit to the size of a file which you can embed information into. For instance if you take a 16 bit image where each pixel is 4 bytes in three colours RGB you can only reliable encode the lower byte before the colour changes become visible in the viewed image. This means that the image you are embedding your data in has to be 1 quarter larger than the encrypted data itself.

2.3 Conclusion Of Literature Review

Its observed that through LSB substitution of steganographic method, the results obtained are pretty impressive as it utilizes the simple fact that any image can be broken into bit-planes each containing different levels of information.

It is also important to discuss that though steganography was once undetected, with the various methods currently used, its not only easy to detect the presence but retrieving them is also easier. For instance, without using a complex software for detection, the other simpler methods for detecting the presence are:

1. Size of the image: A steganographic image has huge storage size as compared to a normal image of same dimensions. If the original image size is in KB's, then the storage size of stego-image will be in a few MB's. This varies with the resolution and type of image used.
2. Noise in the image: A steganographic image has noise when compared to regular image. This is the reason why a little noise is added to the cover image, so that stego-image is less noisy as compared to regular image.

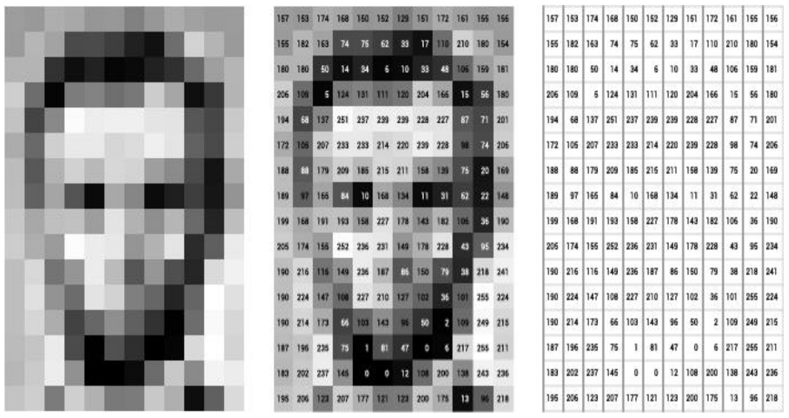
CHAPTER 3

METHODOLOGY

In this chapter , we have explained the approach we used for implementing the project and the working of our program. We have used Embedding Algorithm for hiding the text in an image. The algorithm along with the block diagram are explained further in the chapter.

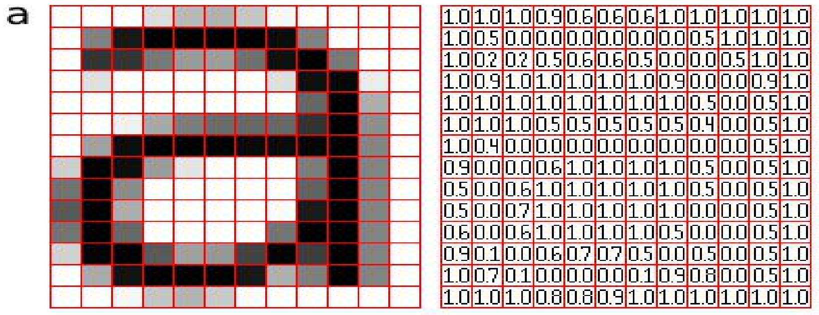
3.1 Matrix Representation Of An Image

An **image is represented as an N*M** (in case of greyscale images) or N*M*3 (in case of color Images) matrix in memory, with each entry representing the **intensity value of a pixel**. In image steganography, a message is embedded into an image by altering the values of some pixels, which are chosen by an encryption algorithm.



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Figure 3.1 Matrix Representation Of A Grayscale Image



1.0	1.0	1.0	0.9	0.6	0.6	0.6	1.0	1.0	1.0	1.0	1.0
1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	1.0	1.0
1.0	0.2	0.2	0.5	0.6	0.6	0.5	0.0	0.0	0.5	1.0	1.0
1.0	0.9	1.0	1.0	1.0	1.0	1.0	0.9	0.0	0.0	0.9	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.4	0.0	0.5	1.0
1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0
0.9	0.0	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.6	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.7	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	1.0
0.6	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.0	0.5	1.0
0.9	0.1	0.0	0.6	0.7	0.7	0.5	0.0	0.5	0.0	0.5	1.0
1.0	0.7	0.1	0.0	0.0	0.0	0.1	0.9	0.8	0.0	0.5	1.0
1.0	1.0	1.0	0.8	0.8	0.9	1.0	1.0	1.0	1.0	1.0	1.0

Figure 3.2 Matrix Representation Of A Coloured Image

3.2 Python Script Using Embedding Algorithm

Our Python Script majorly consists of two functions for 'encoding' and 'decoding'. With a HTML Form displayed on index page of our web application we expect our user to choose between one of the two options displayed, one to Encode and one to Decode. If the **'Encryption' button is clicked** the user is directed to a HTML form where he is required to **enter a Message** that he wishes to hide and **choose a cover image**(image in which message would be hidden). These are **received as objects using Flask** and **passed as parameters to our "Encode()" function**. The **message is converted to an 8-bit binary string**. As a pixel of a image contains viable information about the composition of a image(RGB values),we have e**xpanded these pixels using Python's Pillow module** and **converted them to 8-bit strings** too. We **iterated over these pixels** one by one and **replaced our message** (in a set of 2 bits)with the rightmost bits(LSB) of the image received.
Finally this **encoded image is displayed** and saved with a 'new' string added to the its filename. Similarly, for 'Decode()', data(our message) is retrieved by checking the rightmost bits.

LSB Technique: Least Significant Bit

Least significant bit (LSB) insertion is a common, simple approach to embedding information in a cover image. The least significant bit (in other words, the 8th bit) of some or all of the bytes inside an image is changed to a bit of the secret message. When using a 24-bit image, a bit of each of the red, green and blue color components can be used, since they are each represented by a byte. In other words, one can store 3 bits in each pixel. An 800 × 600 pixel image, can thus store a total amount of 1,440,000 bits or 180,000 bytes of embedded data . For example a grid for 3 pixels of a 24-bit image can be as follows:

(00101101 00011100 11011100)

(10100110 11000100 00001100)

(11010010 10101101 01100011)

When the number 200, which binary representation is 11001000, is embedded into the least significant bits of this part of the image, the resulting grid is as follows:

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

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

(1101001**0** 1010110**0** 01100011)

Although the number was embedded into the first 8 bytes of the grid, only the 3 underlined bits needed to be changed according to the embedded message. On average, only half of the bits in an image will need to be modified to hide a secret message using the maximum cover size [19]. Since there are 256 possible intensities of each primary color, changing the LSB of a pixel results in small changes in the intensity of the colors. These changes cannot be perceived by the human eye - thus the message is successfully hidden. With a well-chosen image, one can even hide the message in the least as well as second to least significant bit and still not see the difference.

3.2 Web Development And Final Workflow

To implement steganography, We have used Python to write a script which contains functions for reusability. To give this Script a web application's functionalities, we have used Flask (which is a Python's Framework) that allows to render HTML Pages and enables communication between the Script and the HTML Pages, hence the users. The process by process shown in fig 3.3

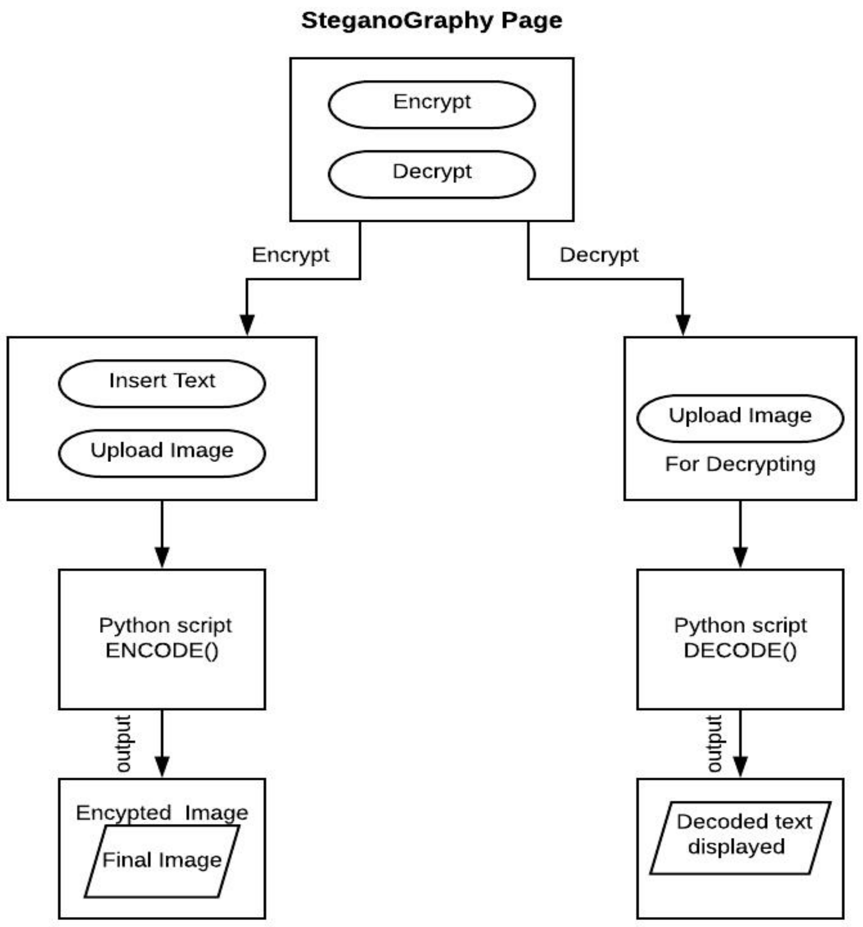


Figure 3.3 Workflow of designed program

CHAPTER 4

RESULTS OBTAINED

After successful running of the python script, the following results were obtained:

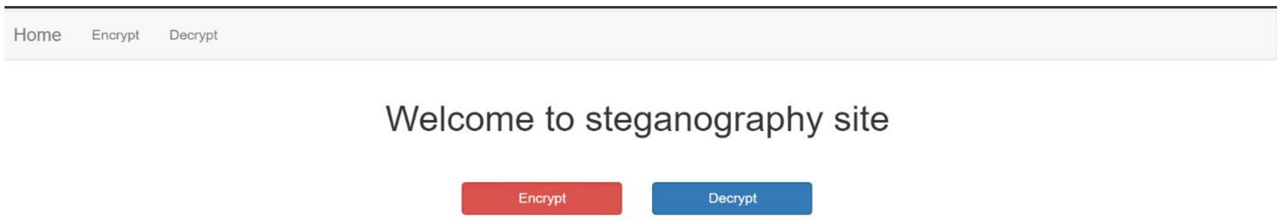


Figure 4.1 Home page which contains encrypt and decrypt options

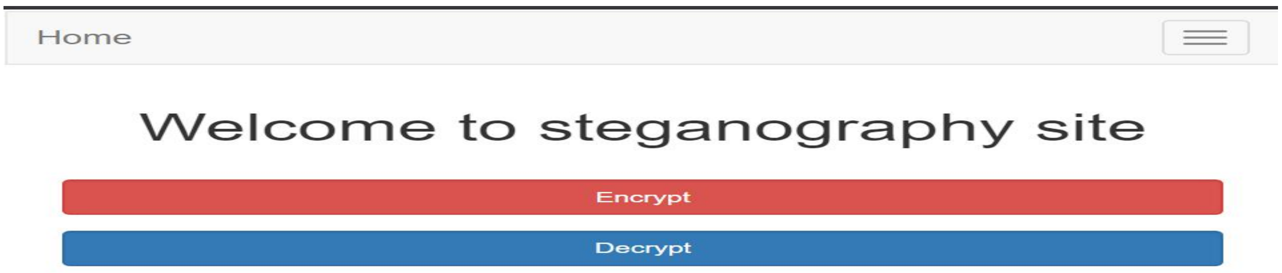


Figure 4.2 Home page for mobile view

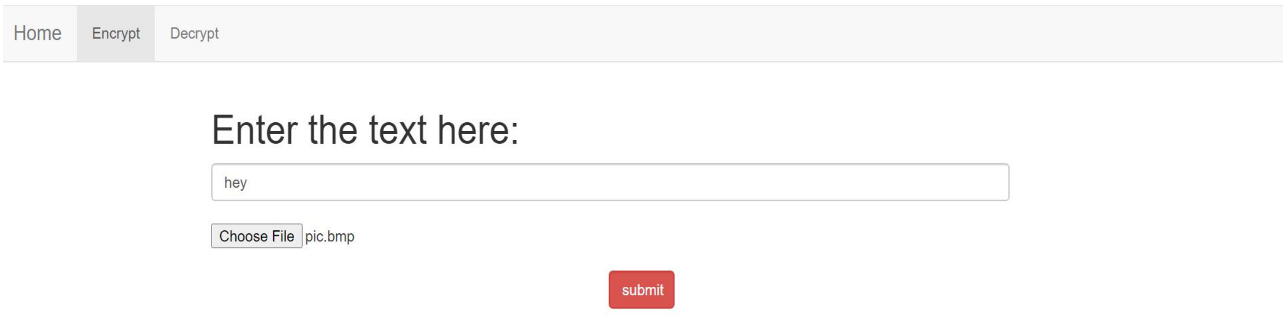


Figure 4.3 Encryption page asking for text to hidden and the cover image

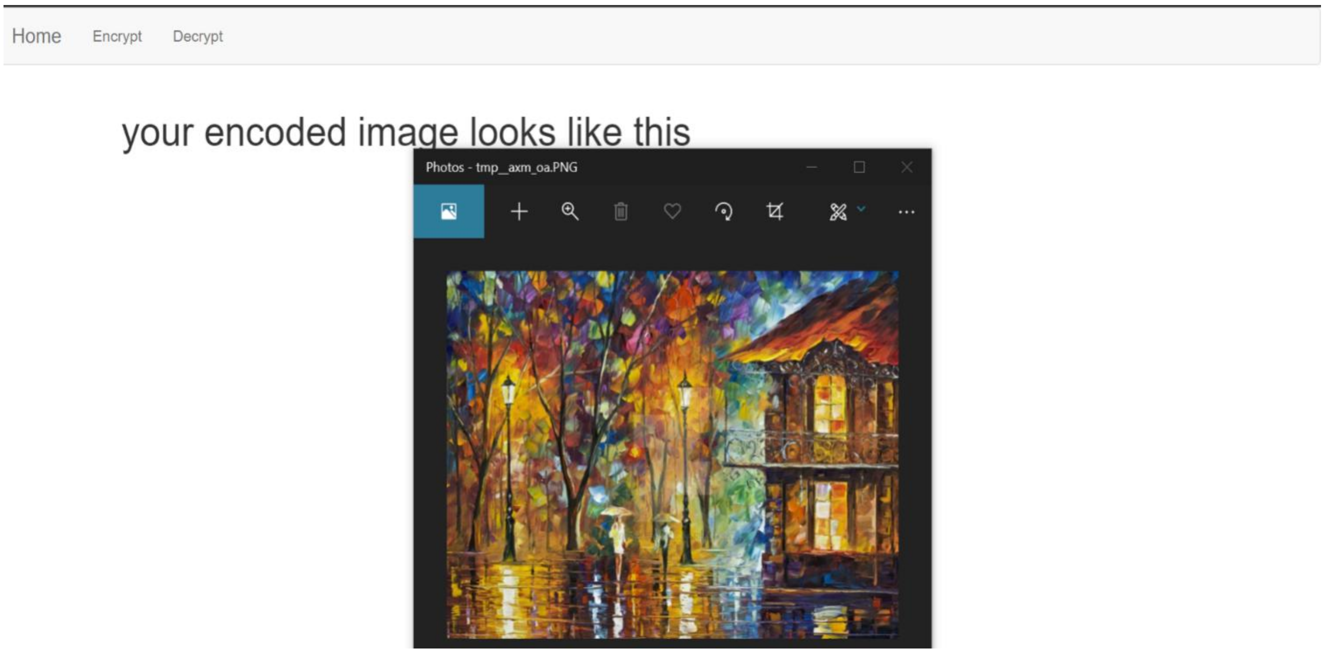
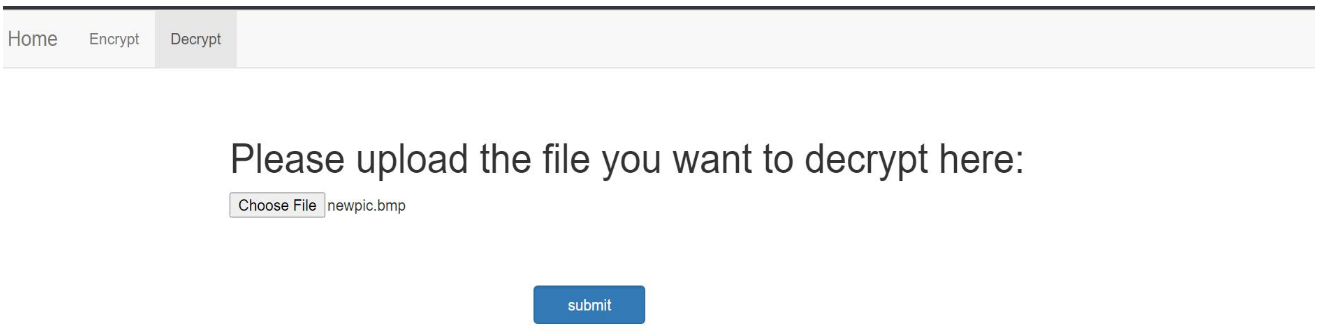


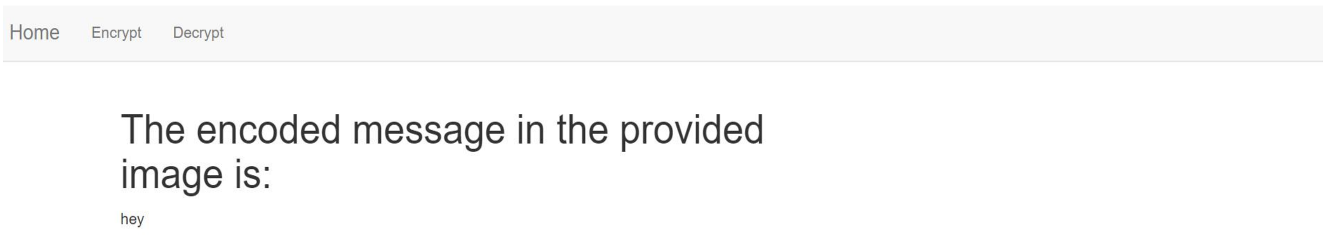
Figure 4.4 Encoded image after successful encryption



Please upload the file you want to decrypt here:

newpic.bmp

Figure 4.5 Decryption page asking for cover image to decode the message



The encoded message in the provided image is:

hey

Figure 4.6 Page displaying the decoded message

The program is successfully running on all the file extensions such as .jpg , .png , .jpeg etc.

CONCLUSION AND FUTURE SCOPE

Steganography, though is still a fairly new idea. There are constant advancements in the computer field, suggesting advancements in the field of steganography as well. It is likely that there will soon be more efficient and more advanced techniques for Steganalysis. A hopeful advancement is the improved sensitivity to small messages. Knowing how difficult it is to detect the presence of a fairly large text file within an image, imagine how difficult it is to detect even one or two sentences embedded in an image! It is like finding a microscopic needle in the ultimate haystack. What is scary is that such a small file of only one or two sentences may be all that is needed to commence a terrorist attack. In the future, it is hoped that the technique of Steganalysis will advance such that it will become much easier to detect even small messages within an image. In this work it explores only a small part of the science of steganography. As a new discipline, there is a great deal more research and development to do. The following section describe areas for research which were offshoots of, or tangential to, our main objectives.

1. Detecting Steganography in Image Files: Can steganography be detected in images files? This is difficult question. It may be possible to detect a simple Steganographic technique by simple analysing the low order bits of the image bytes. If the Steganographic algorithm is more complex, however, and spreads the embedded data over the image is random way or encrypts the data before embedding, it may be nearly impossible to detect.

2. Steganography on the World Wide Web: The world wide web(www) makes extensive use of inline images. There are literally millions of images on various web pages worldwide. It may be possible to develop STEGANOGRAPHY Dept. of Information Technology an application to serve as a web browser to retrieve data embedded in web page images. This stego-web could operate on top of the existing WWW and be a means of covertly disseminating information.

3. Steganography in printed media: If the data is embedded in an image, the image printed, then scanned and stored in a file can the embedded data be recovered? This would require a special form of a steganography to which could allow for in accuracies in the printing and scanning equipment.

REFERENCES

[1] Vipul Sharma and Sunny Kumar, “A New to Hide Text in Images Using Steganography,” *International Journal of Advance Research in Computer Science and Software Engineering* , vol. 3, Issue 4, April 2013.

[2] Jasleen Kour and Deepankar Verma, “Steganography Techniques- A Review Paper,” *International Journal Of Emerging Research in Management and Technology* , vol. 3, Issue 5 May 2014.

[3] T. Morkel, J.H.P Eloff, M.S. Olivier, “An Overview Of Image Steganography,” *Proceedings of the Fifth Annual Information Security South Africa Conference*, Sandton, South Africa, June/July 2005.

[4] "Image Steganography in Cryptography - GeeksforGeeks", *GeeksforGeeks*. [Online]. Available: https://www.geeksforgeeks.org/image-steganography-in-cryptography/

[5] "Welcome to Flask — Flask Documentation (1.1.x)", *Flask.palletsprojects.com*, 2010. [Online]. Available: https://flask.palletsprojects.com/en/1.1.x/

[6] R. Roy, "Hiding data in an image : Image Steganography using Python", *Medium*, 2020. [Online]. Available: https://towardsdatascience.com/hiding-data-in-an-image-image-steganography-using-python-e491b68b1372. [Accessed: 03- Jun- 2020]

[7] "Steganography", *En.wikipedia.org*, 2001. [Online]. Available: https://en.wikipedia.org/wiki/Steganography

APPENDICES

1. Function For Encoding

```
def encodeMessage(image, message,name):
    # img.show()
   imgcpy = image.copy()

   if len(message) == 0:
      raise Exception("Data is empty")
   message += " /// "
   mess8bit = To8bitBin(message)
   '''
   i = int(input("please enter the x coordinate of the starting pixel: "))
   j = int(input("please enter the y coordinate of the starting pixel: "))
   if i == "":
      i = 0
   if j == "":
      j = 0
   '''
   i = 0
   j = 0
   getPixEncode(mess8bit, imgcpy, name, j, i)
   print("Encryption Complete!")
   img = Image.open("new" + name[:-3] + "bmp")
   # img.show()

def To8bitBin(mess):
   binmess = []
   for i in mess:
      binmess.append(format(ord(i), '08b'))
   return binmess

'''
Encodes the picture with the message
Algorithm:
   1: each pixel it switches which value it manipulates
      first the R value
      second the G value
      third the B value
      repeat
   2: Uses two least significant bits to encode the message two bytes at a time
'''
```

2. Function for Replacing LSB bits Of Pixels With Bits of Text

```
def getPixEncode(mess8bit, img, name, i=0, j=0):
    pixelmap = img.load()
    rgb = 0
    for k in mess8bit:
        l = 0
        while l < 8:
            pixel = pixelmap[i, j]
            # print("pixelmap1:",pixelmap[i,j])
            # print("pixel:",pixel)
            p = str(format(int(pixel[rgb]), '08b'))    #08 formats the number to eight digits
zero-padded on the left and 'b' converts to binary
            # print("p:",p)
            newpix = p[0:6] + k[l:l + 2]
            # print("newpix1:", newpix)
            l += 2
            newpix = int(newpix, 2)
            # print("newpix2:",newpix)
            if rgb == 0:   #insert to the red part
                pixelmap[i, j] = (newpix, int(pixel[1]), int(pixel[2]))
            elif rgb == 1:              #insert to the green part
                pixelmap[i, j] = (int(pixel[0]), newpix, int(pixel[2]))
            else:                 #insert to the blue part
                pixelmap[i, j] = (int(pixel[0]), int(pixel[1]), newpix)
            # print("pixelmap2:",pixelmap[i,j])
           #looping conditions for the pixel array's iterator (rgb)
            if rgb == 2:
                rgb = 0
            else:
                rgb += 1
            j += 1
            if j == img.size[1]:
                j = 0
                i += 1

    img.save("new" + name[:-3] + "bmp")

    return render_template('encresult.html',te=img.show())
```

3. Function To decode

```
def decodeMessage(img):
    i=0
    j=0

    # img = Image.open(name)
    # img.show()
    st = getPixDecode(img, i, j)
    return(st[0:len(st) - 5])
```

4. Function For Retrieving Original bits Of Texts From Pixels

```
def getPixDecode(img, i=0, j=0):
    pixelmap = img.load()
    rgb = 0
    f = 1
    temp = ""
    st = ""
    while i < img.size[0]:
        # print("i: ", i)
        while j < img.size[1]:
            pixel = pixelmap[i, j]
            # print("pixel:",pixel)
            p = str(format(int(pixel[rgb]), '08b'))
            # print("p:",p)
            rgb += 1
            temp += p[6:8]
            # print("temp:",temp)
            if rgb == 3:
                rgb = 0
            if f == 4:
                num = int(temp, 2)
                # print("num:",num)
                ch = chr(num)
                # print("ch:",ch)
                st = st + ch
                # print("st: ",st[-1:])
                f = 0
                temp = ""
                num = check(st)
                if num == 1:
                    return st
            f += 1
            j += 1
        j = 0
        i += 1
    return "***NO HIDDEN MESSAGE PRESENT***     "
```

5. Linking Python Script With Flask

```
#flask link part
app = Flask(__name__)
nav=Nav(app)
Bootstrap(app)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
nav.register_element('my_navbar',Navbar(
   View('Home', 'index'),
    View('Encrypt','encryptselect'),
    View('Decrypt','decuploadpage')
    ));
```