IMAGE STEGANOGRAPHY USING LEAST SIGNIFICANT BIT

MINOR PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degree of **M.Sc. Computer Science**

Submitted by

KULPREET KOUR (1893904)

MONI TRIPATHI(1893908)

Under the Supervision of

Prof. Sunil Kumar Muttoo

Department of Computer Science, Faculty of Mathematical Sciences, University of Delhi

ACKNOWLEDGEMENT

It was an intellectually enriching experience to undertake this minor project as a part of our MSc. Computer science 3rd semester curriculum at Department of Computer Science, University of Delhi.

We sincerely express our gratitude towards our project supervisor **Mr**. **Sunil Kumar Muttoo**, Associate Professor, Department of Computer Science, University of Delhi, who guided us at every step in the conceptual undertaking and the implementation of the project work.

We also express our thanks to the Head of Department of Computer Science, **Prof**. **Neelima Gupta**, for providing us the facilities to carry out the project work.

We also acknowledge the cooperation received from the entire laboratory and office staffs.

DECLARATION

We hereby declare that the project work entitled Image Steganography using least significant bit being submitted in partial fulfilment of the requirements for the award of the degree of M.Sc. (Computer Science), is a record of original and bona-fide work carried out by the undersigned in the Department of Computer Science, Faculty of Mathematical Sciences, University of Delhi, New Delhi, India. The work presented in this Project has not been submitted to any other Institute or University for the award of any degree or diploma.

Moni Tripathi	**Kulpreet kour**
(1893908)	**(1893904)**

TABLE OF CONTENTS

ABSTRACT

Steganography is the art and science of hiding the secret data in to the cover file in such a way that no one apart from sender and intended recipient, suspects the secret message. Here image steganography technique is used, in which secret data is embedded in the image file which acts like cover file. Traditional steganographic techniques uses sequential LSB data embedding method, in which secret data is going to embed in the least significant bit position of cover image pixels in sequential manner, this method is easily vulnerable to the attacks and attacker can easily get know to the data embedded position because data is embedded in sequential pixel position in least significant bits.

INTRODUCTION

Steganography is an art of hiding information inside information.

The main objective of Steganography is mainly concerned with the protection of contents of the hidden information. Images are ideal for information hiding because of the large amount of redundant space is created in the storing of images.

Steganography is the technique of hiding secret data within an ordinary, non-secret, file or message in order to avoid detection; the secret data is then extracted at its destination. The use of steganography can be combined with encryption as an extra step for hiding or protecting data.

Definition: It is an art and science of hiding information by embedding it in some other data

Steganography literally means “covered writing”

Goals : To hide a secret message within an object.

STEGANOGRAPHY VS CRYPTOGRAPHY

- The meaning of the steganography is “covered or hidden writing” while cryptography signifies “secret writing”.

- Steganography is an attempt to achieve secure and undetectable communication. On the other hand, cryptography intends to make the message readable for only the target recipient but not by others through obtaining a disguised form of message.

- In steganography, the main structure of the message is not changed whereas cryptography imposes a change on the secret message before transferring it over the network.

- The cryptography is prevalently used unlike steganography, which is not so familiar.

- The degree of the security of the secret data is measured by the key length which makes the algorithm strong and unbreakable. Conversely, there is no such thing in steganography.

- Steganography provides only confidentiality and authentication. On the contrary, the principles of security provided by the cryptography are confidentiality, integrity, authentication, and non-repudiation.

- Spacial domain, transform domain embedding and model-based are some of the algorithms used in steganography. In contrast, the cryptography uses techniques named as transpositional, substitution, stream and block ciphers.

- The steganography can be employed on any medium such as text, audio, video and image while cryptography is implemented only on the text file.

- The reverse engineering employed to decode the message in cryptography is known as cryptanalysis. As against, the technique used to detect the presence of the steganography is known as steganalysis.

TYPES OF STEGANOGRAPHY

There are four type of steganography

- ❖ Text steganography
- ❖ Image steganography
- ❖ Audio steganography
- ❖ Video Steganography

Text steganography: Text steganography can be achieved by altering the text formatting, or by altering certain characteristics of textual elements (e.g.characters).

Text Steganography Techniques:

a) Selective hiding: This hides the characters in the first (or any specific location) characters of the words. Concatenating those characters help extracting the text. But this technique requires huge amount of plain text.

b) HTML web pages: This may hide text using the fact that attributes of HTML tags are case insensitive. Those characters can then be used to retrieve the original text.

c) Hiding using Whitespace: Fewer numbers of whitespaces may specify a 0 and more number of whitespaces between words may determine a 1.

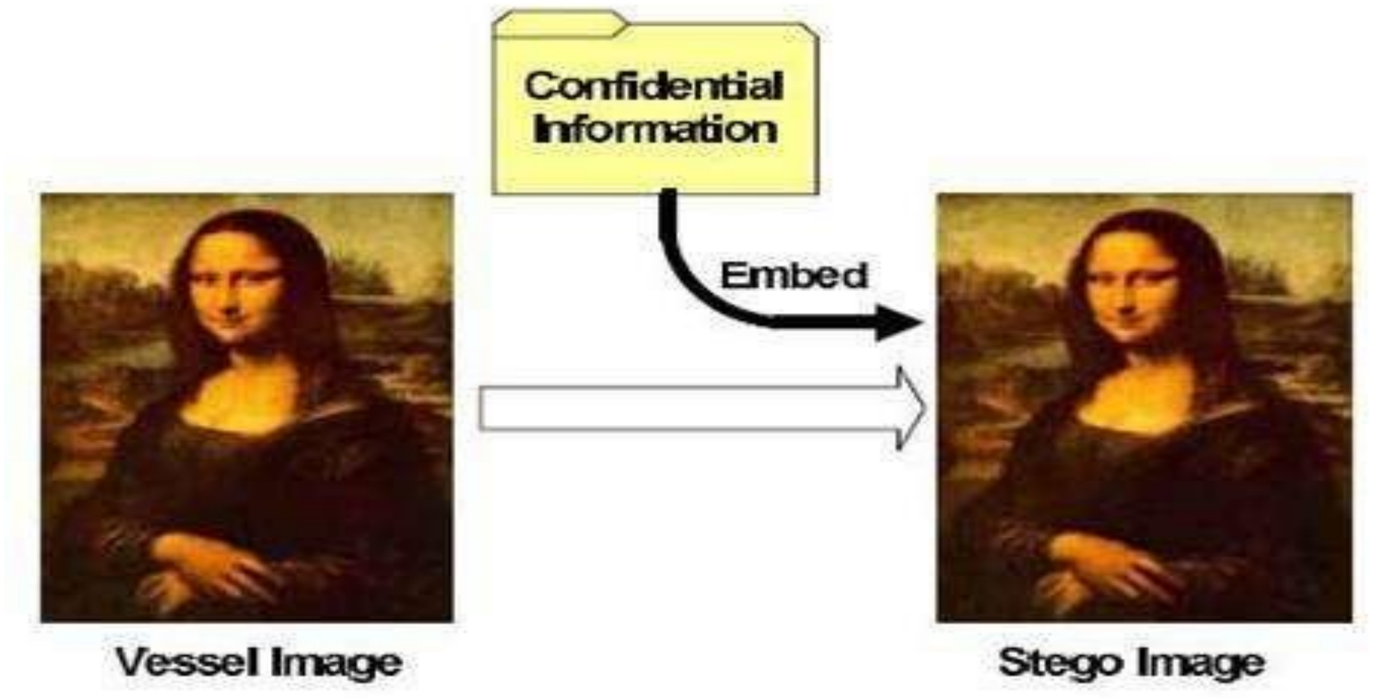
d) Semantic Hiding: Uses synonyms to hide the message

Image steganography: Images are used as the message carriers. Images are the most popular cover objects used for steganography.

Audio steganography: In audio steganography, secret message is embedded into digitized audio signal which result slight altering of binary sequence of the corresponding audio file.

Video Steganography: Video Steganography is a technique to hide any kind of files or information into digital video format. Video (combination of pictures) is used as carrier for hidden information.

IMAGE STEGANOGRAPHY



Images are used as the message carriers.

Images are the most popular cover objects used for steganography.

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.

Image Steganography requires following elements to carry out the work:

Cover medium: It is an image that holds secret message.

The Secret message: it is message to be transmitted. It can be plain or encrypted text, images or any other data.

The Stego-key: it is key used to hide the message (May or may not be used).

How is it done?

An image is represented as an N*M (in case of greyscale images) or N*M*3 (in case of colour 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. The recipient of the image must be aware of the same algorithm in order to known which pixels he or she must select to extract the message.

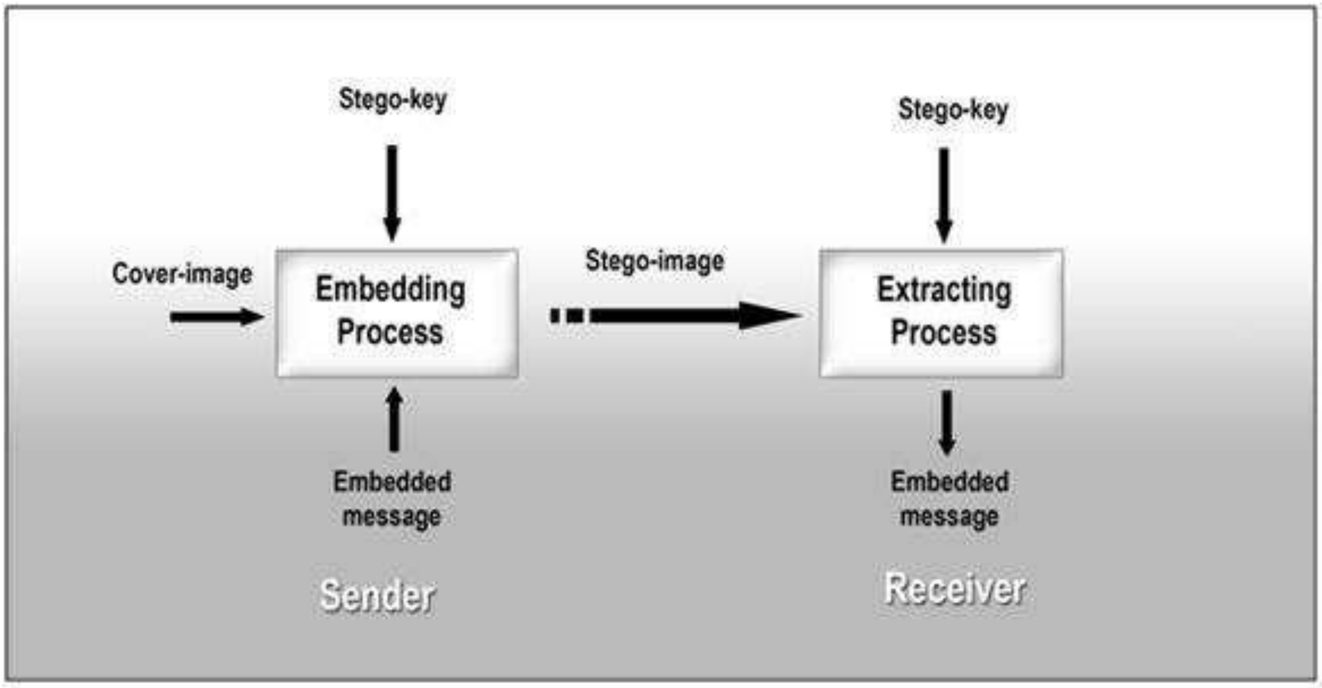


IMAGE STEGANOGRAPHY TECHNIQUES

- ❖ By using LSB(Least Significant Bit algorithm)
- ❖ Masking and Filtering
- ❖ Algorithms and Transformation

Image Steganography using LSB algorithm

- The most common and popular method of modern day steganography is to make use of LSB of picture’s pixel information
- This technique works best when the file is longer than the message file and if image is grayscale.
- When applying LSB techniques to each byte of a 24 bit image, three bits can be encoded into each pixel.

If the LSB of the pixel value of cover image C(I ,j) is equal to the message bit SM of secret massage to be embedded, C(i ,j) remain unchanged; if not, set the LSB of C(i, j) to SM.

Message embedding Procedure is given below:

- S(i ,j) = C(i ,j) - 1, if LSB(C(i ,j)) = 1 and SM = 0
- S(i ,j) = C(i ,j) + 1, if LSB(C(i ,j)) = 0 and SM = 1
- S(i ,j) = C(i ,j), if LSB(C(i ,j)) = SM

Where LSB(C(i, j)) stands for the LSB of cover image C(i ,j) and “SM” is the next message bit to be embedded. S(i ,j) is the stego image

Implementation

Tools

Language used: python

Platform used : Anaconda

IDE :Jupyter notebook

Packages used:

- Numpy
- Math
- PIL
- Cv2
- Matplotlib
- Skimage

Phases

This project is divided into 2 phases:

1. Data Embedding
2. Data Extraction

Data Embedding Algorithm

Step 1: Extract the pixels of the cover image.

Step 2: Extract the characters of the text .

Step 3: Extract the characters from the Stego key.

Step 4: Choose first pixel and place it in first component of pixel.

Step 5: Place some terminating symbol to indicate end of the key. 0 has been used as a terminating symbol in this algorithm.

Step 6: Insert characters of text in each component of next pixels by replacing it.

Step 7: Repeat step 6 till all the characters has been embedded.

<u>Data Extraction Algorithm</u>

Step 1: Extract the pixels of the stego image.

Step 2: Now, start from first pixel and extract stego key characters from first component of the pixels. Follow Step3 up to terminating symbol, otherwise follow step 4.

Step 4: If this extracted key matches with the key entered by the receiver, then follow Step 5, otherwise terminate the program

Step 5 till up to terminating symbol, otherwise follow step6

Step 6: Extract secret message.

Example:

We can use images to hide things if we replace the last bit of every color's byte with a bit from the message.

Message A-01000001

Image with 3 pixels

Pixel 1 : 11111000 1100100100000011

Pixel 2 : 11111000 1100100000000011

Pixel 3 : 11111000 11001001 00000011

Now we hide our message in the image.

Message: 01000001

Pixel 1 : 11111000 11001001 00000010

Pixel 2 : 11111000 11001000 00000010

Pixel 3: 11111000 1100100100000011

IMPLEMENTATION

STEGANOGRAPHY FOR COLORED IMAGE

- We take first colored image: flowers.jpg
  # On red channel
  Peak signal noise ratio of colored image :55.6852
  We concatenated two colored image

original image

stego image

- We take second colored image :color.jpg
  #On red channel
  Peak signal noise ratio of colored image:56.045
  We concatenated two colored image

original image stego image

- We take third colored image:frog.jpg
  # On red channel
  Peak signal noise ratio of colored image:56.956
  We concatenated two colored image

original image stego image

- We take fourth colored image:eye.jpg
  # On red channel
  Peak signal noise ratio of colored image:42.986
  We concatenated two colored image

original image stego image

- We take colored image : flowers.jpg
  # On red channel/green channel/bluechannel

```
# libraries for importing image
from PIL import Image, ImageFont, ImageDraw
import textwrap

def decode_image(file_location="abc3.png"):
 """Decodes the hidden message in an image
```

```
    file_location: the location of the image file to decode. By
default is the provided encoded image in the images folder
    """
    encoded_image = Image.open(file_location)
    red_channel =  encoded_image.split()[0]
    x_size = encoded_image.size[0]
    y_size = encoded_image.size[1]

    decoded_image = Image.new("RGB",
encoded_image.size)
    pixels = decoded_image.load()

    for i in range(x_size):
        for j in range(y_size):
            if bin(red_channel.getpixel((i, j)))[-1] == '0':
                pixels[i, j] = (255, 255, 255)  #update here also
                print(pixels[i, j])
            else:
                pixels[i, j] = (0,0,0)

                green_channel = encoded_image.split()[1]
    x_size = encoded_image.size[0]
    y_size = encoded_image.size[1]

    decoded_image = Image.new("RGB",
encoded_image.size)
    pixels = decoded_image.load()
    for i in range(x_size):
        for j in range(y_size):
            if bin(green_channel.getpixel((i, j)))[-1] == '0':
                pixels[i, j] = (255, 255, 255)  #update here also
                print(pixels[i, j])
            else:
                pixels[i, j] = (0,0,0)
                blue_channel = encoded_image.split()[2]


    x_size = encoded_image.size[0]
    y_size = encoded_image.size[1]

    decoded_image = Image.new("RGB",
encoded_image.size)
    pixels = decoded_image.load()

    for i in range(x_size):
        for j in range(y_size):
            if bin(blue_channel.getpixel((i, j)))[-1] == '0':
                pixels[i, j] = (255, 255, 255)  #update here also
                print(pixels[i, j])
            else:
                pixels[i, j] = (0,0,0)
    decoded_image.save("abc.png")

def  write_text(text_to_write, image_size):
    """
    Writes text to an RGB image. Automatically line wraps
    text_to_write: the text to write to the image
    """
    image_text = Image.new("RGB", image_size)
    font = ImageFont.load_default().font
    drawer = ImageDraw.Draw(image_text)

    #Text wrapping. Change parameters for different text
formatting
    margin = offset = 10
    for line in textwrap.wrap(text_to_write, width=60):
        drawer.text((margin,offset), line, font=font)
        offset += 10
    return image_text


def  encode_image(text_to_encode,
template_image="flowers.jpg"):
    """Encodes a text message into an image
    text_to_encode: the text to encode into the template
image
    template_image: the image to use for encoding. An
image is provided by default.
```

```
    """
    template_image = Image.open(template_image)
    red_template = template_image.split()[0]
    green_template = template_image.split()[1]
    blue_template = template_image.split()[2]
    #the size of the image
    x_size = template_image.size[0]
    y_size = template_image.size[1]

    #text draw
    #to hide the text in the image by calling write_text
function
    image_text = write_text(text_to_encode,
template_image.size)
    print(image_text)
    # convert the hidden text image into grayscale image
    bw_encode = image_text.convert('1')
    print(bw_encode)

    #encode text into image
    encoded_image = Image.new("RGB", (x_size, y_size))
    pixels = encoded_image.load()

    for i in range(x_size):
        for j in range(y_size):
            red_template_pix = bin(red_template.getpixel((i,j)))
            #print(red_template_pix)
            old_pix = red_template.getpixel((i,j))
            #print(old_pix)
            tencode_pix = bin(bw_encode.getpixel((i,j)))

            if tencode_pix[-1] == '1':
                red_template_pix = red_template_pix[:-1] + '1'
            else:
                red_template_pix = red_template_pix[:-1] + '0'
            pixels[i, j] = (int(red_template_pix, 2),
green_template.getpixel((i,j)), blue_template.getpixel((i,j)))

    for i in range(x_size):
        for j in range(y_size):
            green_template_pix =
bin(green_template.getpixel((i,j)))
            #print(green_template_pix)
            old_pix = green_template.getpixel((i,j))
            #print(old_pix)
            tencode_pix = bin(bw_encode.getpixel((i,j)))

            if tencode_pix[-1] == '1':
                green_template_pix = green_template_pix[:-1] +
'1'
            else:
                green_template_pix = green_template_pix[:-1] +
'0'
```

```
            pixels[i, j] =
(red_template.getpixel((i,j)),int(green_template_pix,
2),blue_template.getpixel((i,j)))

    for i in range(x_size):
        for j in range(y_size):
            blue_template_pix =
bin(blue_template.getpixel((i,j)))
            #print(blue_template_pix)
            old_pix = blue_template.getpixel((i,j))
            #print(old_pix)
            tencode_pix = bin(bw_encode.getpixel((i,j)))

            if tencode_pix[-1] == '1':
                blue_template_pix = blue_template_pix[:-1] + '1'
            else:
                blue_template_pix = blue_template_pix[:-1] + '0'
            pixels[i, j] =
(red_template.getpixel((i,j)),green_template.getpixel((i,j)),
int( blue_template_pix,2))

    encoded_image.save("abc3.png")

if __name__ == '__main__':
    #decode_image()
    encode_image("hello world ")
    decode_image()
```

```
Peak signal noise ratio of colored image:39.263
```

```
"""
Signal-to-noise ratio numbers are all about the strength of
the desired signal compared to the unwanted noise.
The larger the number, the more the desired signal “stands
out”
in comparison to the noise, which means a clearer
transmission of better technical quality.
A negative number means the noise is stronger than the
desired signal, which may spell trouble,
such as a cell phone conversation that’s too garbled to
understand.
"""
import numpy
import math
import cv2
original1 = cv2.imread("flowers.jpg")
contrast1 = cv2.imread("abc3.png",1)#stego image
def psnr(img1, img2):
    mse = numpy.mean( (img1 - img2) ** 2 )
    #If you have a replica of your signal (image) that is noise
free,
    #you can calculate the correlation coefficient which is
directly related to SNR.
```

```
    #In this context there is no "maximum SNR" but will be
the SNR for your entire image,
    #meaning the power of your desired signal relative to
everything else (distortions).
    if mse == 0:
        return 100
    PIXEL_MAX = 255.0
    return 20 * math.log10(PIXEL_MAX / math.sqrt(mse))

d=psnr(original1,contrast1)
print(d)# unit is in power(watts)
```

We concatenated two colored image

original image

stego image

STEGANOGRAPHY FOR GREY SCALE IMAGE

- We take first grey scale image:apple.jpg
  # On red channel /green channel/blue channel
  Peak signal noise ratio of grey scale image:55.879
  We concatenated two grey scale image

original image

stego image

- We take second grey scale image:dog.jpg
  # On red channel /green channel/bluechannel
  Peak signal noise ratio of grey scale image:56.1144
  We concatenated two grey scale image

original image stego image

- We take third grey scale image:land.jpg
  #On red channel /green channel/blue channel
  Peak signal noise ratio of grey scale image:55.9874
  We concatenated two grey scale image

original image stego image

- We take fourth grey scale image: tower.jpg
  # On red channel /green channel/blue channel.
  Peak signal noise ratio of grey scale image:55.876
  We concatenated two grey scale image

original image

stego image

CONCLUSION AND FUTURE SCOPE

To hide confidential information steganography can be effectively used. The objective of any Steganographic method is to hide maximum secret information which is immune to external attacks and also should not convey the fact that the cover medium is carry secret information. This thesis has used LSB substitution technique for time domain and different transforms for frequency domain.

The random key made use of while LSB technique is employed has proved to be better than simple LSB substitution. When the personal is key is made use of in the techniques have not altered the resolution of the image much and appears to be negligible as been seen with the obtained results. Hence the hidden data getting damaged buy the third person is almost impossible. The algorithm can be implemented in both grayscale and color image since it has made use of 8 bit and 24 bit images of size for both cover and secret image.

In Spatial domain methods one can get high payload capacity. The edge detection techniques which are used in the current methods do not recognizes the shades of the edge region, which can also be considered to embed the extra bits. The number of edge pixels can be increased by identifying the edges and shades of edges.

The Transform domain methods are highly robust. They embed the message bits in the regions which are highly insensitive to compression, filtration or transformation.

But their payload capacity is low. Also the visual qualities of the Stego-image are poor.

Spatial domain is better compared to transform domain.

In Spatial Domain techniques simple LSB substitution methods are less secure and payload capacity is less. To increase security Randomization techniques are used.

These methods spread the message randomly into the cover image but payload capacity is still less. Edge detection method increases the payload capacity and it improves

randomness and hence security. As discussed the current edge detection methods do not recognizes the shades of the edge region which are also capable of embedding more bits

with less distortion. The shades of the edges are detected by Multiple Edge Detection method. In Multiple Edge Detection method the edge detection method is applied for 2-3times which increase the number of edge pixels. As the number of edge pixels is increased more data can be hidden in to the cover image. To add randomness to the embedding procedure which increases security, Variable Embedding can be employed in which a suitable embedding ratio is chosen and according to that the message bits are embedded into the image pixels. The increase in the payload decrease PSNR value, which indicates image degradation amount. To improve PSNR Minimum Error Replacement[MER] method is used. In this method the next higher bit than the last embedded bit is toggle to decrease the pixel error. Stego and crypto way shows new way of embedding the data, especially in Multiresolution analysis, there are different ways of getting Multiresolution; this thesis has made use of Multiresolution analysis on wavelets and Curvelets. The experimental analysis has proved that Curvelets is the best Multiresolution transformation available.

This work has been implemented with the library which has an accuracy of 15 fractional digits. The results obtained have a good PSNR value, along with the crypto style of embedding. Steganography is to create secrete communication, in addition to this crypto way of embedding gives us higher end of security. Even if the person gets both stego and cover image he needs key to retrieve the data, without the key one can’t recover the data. Thus additional security is incorporated to the normal Steganography technique. Image steganography is successfully obtained for different embedding techniques in both wavelet and curvelet domain. Depending on the demand of the application one can choose any of the techniques developed. The simulations are performed in MatLab7.7.0(R2008b) and PSNR are calculated. It is found that Curvelet is a better approach than wavelet transformation. In this work data embedding is done by considering block of size 85*85.

In the frequency domain technique use of skin tone algorithm has given very good results. The proposed method uses skin tone detection for finding skin portion of image and within this skin portion secret data embedding is done using DWT domain. Four cases of embedding are considered. It is observed that hiding of secret data in only the cropped skin portion enhances the security.

And according to result and discussion proposed scheme provides good image quality in all four cases. In Transform domain methods the pay load capacity is low and the future scope is to enhance Stego image quality and payload capacity. The payload capacity can still be increased by embedding more bits into edge pixels and less to non edge pixels. Future Enhancement can be done by embedding data pixel by pixel, thus increase in the payload can be attained.

Future Enhancement

In this work it explores only a small part of the science of steganography. As a new displine, 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 analyzing 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. How widespread is the Use of Steganography?

If a technique or set of techniques could be devised to detect steganography, it would be interesting to conduct a survey of images available on the internet to determine if steganography is used, by whom and for what purposes. Steganographic applications are available on the Internet, but it is not known if they are being used.

3. 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 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.

4. 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.

5. Anti-steganography measures

As was seen in this thesis, JPEG garbles any unencoded steganographically embedded data. Also, palettization(mapping a large number of colors in an image to a smaller subset of colors) of an image will it unsuitable for steganography. It is likely, as with JPEG, that some means may be employed to prevent loss of steganographically embedded data when its wrapper file is processed. The question remains open as to what is the most effective anti Steganographic tools or set of tools

REFRENCES

- https://techdifferences.com/difference-between-steganography-and-cryptography.html

- Book title: Information hiding techniques for steganagraphy and digital watermaking by Stefan Katzenbeisser and Fabien A.P. Petitcolas

- https://en.wikipedia.org/wiki/Lossy_compression

- https://www.geeksforgeeks.org/image-steganography-in-cryptography/

- https://shodhganga.inflibnet.ac.in/bitstream/10603/92404/12/15.chapter%206.pdf

- https://www.slideshare.net/SreelekshmiSree1/image-steganography-using-lsb

- https://en.wikipedia.org/wiki/Lossless_compression

- https://pythonprogramming.net/loading-images-python-opencv-tutorials/