Implementation of Different Steganography Techniques in Image Domain

Enrollment Number: 101306

Name: Chyanika Jindal

Project Supervisor: Brig. (Retd.) Prof. Dr. S.P. Ghrera

May 2014

submitted in partial fulfillment of the degree

Bachelor of Technology

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING AND INFORMATION TECHNOLOGY

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
WAKANAGHAT

CERTIFICATE

This is to certify that the work entitled **"Implementation of Different Steganography Techniques in Image Domain"** submitted by **Chyanika Jindal (101306),** in partial fulfillment for the award of degree of **Bachelor of Technology** of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision.

This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Guide:

Name of Guide: **Brig. (Retd.) Prof. Dr. S.P. Ghrera**

Designation: **Head of Department**

Date:

ACKNOWLEDGEMENT

This project could not have been at the stage it is right now had it not been for the cooperation of Brig. (Retd.) Prof. Dr. S.P. Ghrera, our project guide, who was always there to tell us how to go about our project in a systematic manner and who always took out time to help us with our technical and non-technical doubts at various stages of the project.

Equally important was the contribution of Dr. Hemraj Saini, our project supervisor, who kept faith in our ability to complete the project well and on time.

Last but not the least; it was the teachers of the Jaypee University of Information Technology, Waknaghat, particularly Mr. Amol Vasudeva, Mr. Amit Srivastava, and Mrs. Anshul Sood who came to our rescue whenever we got stuck in any piece of code or otherwise.

ABSTRACT

Steganography is the art of hiding the fact that communication is taking place, by hiding information in other information. Many different carrier file formats can be used, but digital images are the most popular because of their frequency on the internet. For hiding secret information in images, there exists a large variety of steganography techniques some are more complex than others and all of them have respective strong and weak points. Steganography is the practice of hiding private or sensitive information within something that appears to be nothing out to the usual. If anyone view the object that the information is hidden inside he or she will have no idea that there is any hidden information, therefore the person will not attempt to decrypt the information. This project report intends to give an overview of image steganography, its uses and techniques. The most common use of steganography is to hide a file/message inside another file.

Signature of Student:
Name:
Date:

Signature of Supervisor:
Name:
Date:

TABLE OF CONTENTS

List of Figures

INTRODUCTION

Information hiding can be mainly divided into three processes - cryptography, steganography and watermarks. Cryptography is the process of converting information to an unintelligible form so that only the authorized person with the key can decipher it. As many advances were made in the field of communication it became rather simple to decrypt a cipher text. Hence more sophisticated methods were designed to offer better security than what cryptography could offer. This lead to the discovery of stenography and watermarking. Stenography is the process of hiding information over a cover object such that the hidden information cannot be perceived by the user. Thus even the existence of secret information is not known to the attacker. Watermarking is closely related to stenography, but in watermarking the hidden information is usually related to the cover object. Hence it is mainly used for copyright protection and owner authentication**.**

Steganography is the art and science of hidding messages in such a way that no one, except the sender and the recipient, identify the existence of the message. The advantage of steganography over cryptography is that messages do not attract attention . Plainly visible encrypted messages—no matter how unbreakable—will arouse suspicion. Therefore, whereas cryptography protects the contents of a message, steganography can be said to protect both messages and communicating parties.

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 steganographic transmission because of their large size.

Different kinds of Steganography are:

1. **Text**- Hiding information in text is most important method of steganography. An obvious method was to hide a secret message in every nth letter of every word of a text message.

2. Images- In the domain of digital images many different image file formats exist, most of them for specific applications. For these different image file formats, different steganographic algorithms exist.

3. Audio/Video- To hide information in audio files similar techniques are used as for image files. One different technique unique to audio steganography is masking, which exploits the properties of the human ear to hide information unnoticeably. A faint, but audible, sound becomes inaudible in the presence of another louder audible sound .This property creates a channel in which to hide information.

4. Protocol- The term protocol steganography refers to the technique of embedding information within messages and network control protocols used in network transmission. In the layers of the OSI network model there exist covert channels where steganography can be used.

Image steganography-

Image steganography techniques can be classified into two broad categories: Spatial-domain based steganography and Transform Domain based steganography.

Image definition –

To a computer, an image is a collection of numbers that constitute different light intensities in different areas of the image. This numeric representation forms a grid and the individual points are referred to as pixels.

Image Compression –

When working with larger images of greater bit depth, the images tend to become too large to transmit over a standard Internet connection. In order to display an image in a reasonable amount of time, techniques must be incorporated to reduce the image's file size. These techniques make use of mathematical formulas to analyze and condense image data, resulting in smaller file sizes. This process is called compression.

Spatial and Transform Domain –

Image steganography techniques can be divided into two groups: those in the Image Domain and those in the Transform Domain. Image – also known as spatial – domain techniques embed messages in the intensity of the pixels directly, while for transform – also known as frequency – domain, images are first transformed and then the message is embedded in the image.

Image domain techniques encompass bit-wise methods that apply bit insertion and noise manipulation and are sometimes characterized as "simple systems". The image formats that are most suitable for image domain steganography are lossless and the techniques are typically dependent on the image format.

Steganography in the transform domain involves the manipulation of algorithms and image transforms.

These methods hide messages in more significant areas of the cover image, making it more robust. Many transform domain methods are independent of the image format and the embedded message may survive conversion between lossy and lossless compression.

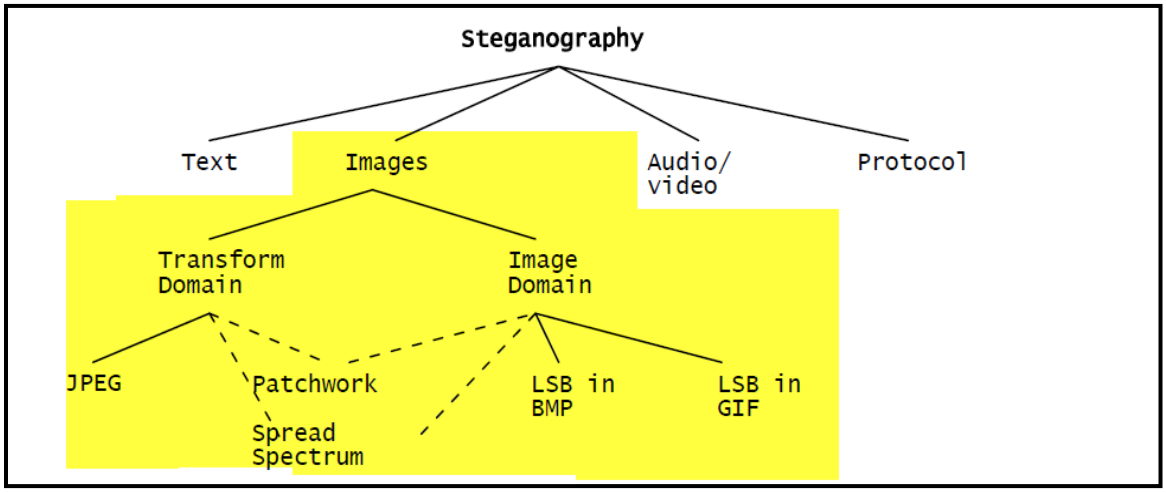


Fig. 1

A. Image Domain-

- LSB Insertion Method
- Grey Level Method
- Hash Based Steganography

B. Transform Domain

- Discrete cosine transform (DCT)
- Integer Wavelet transform(IWT)
- Discrete wavelet transform(DWT)

PROBLEM STATEMENT

Steganography and steganalysis disciplinarians perform spectrum opposite tasks where concealed messages exist in digital media. Steganography is the art and science of writing hidden messages, so only the intended recipient knows the existence of the message. Conversely, steganalysts detect and destroy these hidden messages. In this scenario, a Alice uses a steganography tool to hide a message in a digital cover image. She wishes to pass the image to another Bob.

Bob has the key to the message, so he can extract the message. However, Alice cannot directly hand the image to Bob since they do not share a common network. Therefore, Alice must send the image through internet. This is the scenario used, where the message transportation method is unsecured.

How can we send a message secretly to the destination?

Using steganography, information can be hidden in carriers such as images, audio files, text files, videos and data transmissions.

In this project, we implemented a steganographic system to hide a digital text of a secret message inside the image cover media.

The main objectives of the project are:

- Requirement of this steganography system is that the hidden message carried by stego-media should not be sensible to human beings.
- The other goal of steganography is to avoid drawing suspicion to the existence of a hidden message.
- This approach of information hiding technique has recently become important in a number of application areas.

MOTIVATION

The internet allows for easy dissemination of information over large areas. This is both a blessing and a curse since friends all over the world can view your information but so can everyone else. Encrypting data has been the most popular approach to protecting information but this protection can be broken with enough computational power. An alternate approach to encrypting data would be to hide it by making this information look like something else. This way only friend would realize its true content. In particular, if the important data is hidden inside of an image then everyone but your friends would view it as a picture. At the same time your friends could still retrieve the true information. This technique is often called data hiding or stenography and you will be implementing the technique in the class project. Motivation: Privacy and anonymity is a concern for most people on the internet. Image Steganography allows for two parties to communicate secretly and covertly. It allows for some morally-conscious people to safely whistle blow on internal actions; it allows for copyright protection on digital files using the message as a digital watermark. One of the other main uses for Image Steganography is for the transportation of high-level or top-secret documents between international governments. While Image Steganography has many legitimate uses, it can also be quite nefarious. It can be used by hackers to send viruses and trojans to compromise machines, and also by terrorists and other organizations that rely on covert operations to communicate secretly and safely.

LITERATURE REVIEW

Image Steganography Techniques in Spatial And Trasnform Domain,their Parameters and Analytical Techniques:

APPLICATIONS OF STEGANOGRAPHY:

Steganography, in general, have many applications including copyright protection, Feature Tagging and secret communications etc.

1 .Feature Tagging

Captions, annotations, time stamps and other descriptive elements can be embedded inside an image, such as the name of the individuals in a photo or location in a map. Copying the stego image also copies all of the embedded features and only parties who possess the decoding stego key will be able to extract and view the features.

2. Secret Communication

In many situations, transmitting a cryptographic message draws unwanted attention. The use of cryptographic technology may be restricted or forbidden by law. However, the used steganography does not advertise covert communication and therefore, avoid scrutiny of the sender, message and recipient.

3. Digital Watermark

The objective of a digital watermark is to place an indelible mark on an image. Usually, this means encoding only a handful of bits, sometimes as few as one. This "signature"could be used as a means of tracing the distribution of images for an on-line news service and for photographers who are selling their work for digital publication.

4. Tamper Proofing

The objective of tamper-proofing is to answer the question, "Has this image been modified?" Tamper-proofing techniques are related, but distinct from the other data-hiding technologies.

PARAMETERS OF IMAGE STEGANOGRAPHY TECHNIQUES:

1. Hiding Capacity

Hiding capacity is the size of information that can be hidden relative to the size of the cover. A larger hiding capacity allows the use of a smaller cover for a message of fixed size, and thus decreases the bandwidth required to transmit the stego-image.

2. Perceptual Transparency

The act of hiding the message in the cover necessitates some noise modulation or distortion of the cover image. It is important that the embedding occur without significant degradation or loss of perceptual quality of the cover. In a secret communications application, if an attacker notices some distortion that arouses suspicion of the presence of hidden data in a stego-image, the steganographic encoding has failed even if the attacker is unable to extract the message.

3. Robustness

Robustness refers to the ability of embedded data to remain intact if the stego-image undergoes transformations, such as linear and non-linear filtering, addition of random noise, sharpening or blurring, scaling and rotations, cropping or decimation, lossy compression, and conversion back to digital form.

4. Tamper Resistance

Beyond robustness to destruction, tamper-resistance refers to the difficulty for an attacker to alter or forge a message once it has been embedded in a stego-image, such as a pirate replacing a copyright mark with one claiming legal ownership.

BASIC MODEL OF STEGANOGRAPHY

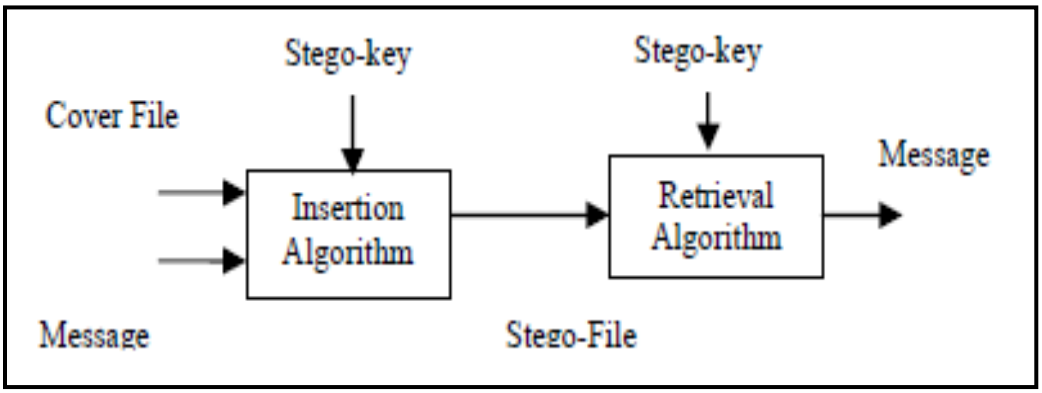


Fig. 2

EXISTING METHODS OF IMAGE STEGANOGRAPHY IN SPATIAL DOMAIN:

1. LSB (Least Significant Bit) Method

The popular and oldest method for hiding the message in a digital image is the LSB method . In LSB method we hide the message in the least significant bits (LSB's) of pixel values of an image. In this method binary equivalent of the secret message is distributed among the LSBs of each pixel.

2. 6^{th}, 7^{th} Bit Method

This algorithm is to hide the message in 6^{th} and 7^{th} bit of pixel value. Here instead of LSB bits 6^{th} and 7^{th} bit of pixel values are used to hide the secret data within an image. This novel proposed method overcomes the all disadvantages of LSB insertion method.

3. GLM (Gray Level Modification) Method

Gray level modification Steganography is a technique to map data (not embed or hide it) by modifying the gray level values of the image pixels. GLM Steganography uses the concept of odd and even numbers to map data within an image. It is a one-to-one mapping between the

binary data and the selected pixels in an image. From a given image a set of pixels are selected based on a mathematical function. The gray level values of those pixels are examined and compared with the bit stream that is to be mapped in the image.

4. Hiding Data in 6^{th}, 7^{th} & 8^{th} Bit of Pixel Values

It is a approach to hide the data 6^{th}, 7^{th} and 8^{th} bit of pixel values. In this method 6^{th}, 7^{th} and 8^{th} bits of the image pixels are used to hide the message. Since this method involves 8^{th} bit for hiding the message, intruder can easily change 8^{th} bit of all image pixels and this may result in the loss of message. To avoid this, time factor has been introduced, i.e. at some time t1, sender sends the cover object with message and at some other time t2 sender sends the cover object without message. Sender and recipient agree on this time factor initially before starting any Communication.

5. Parity Checker Method

This method, gives the concept of odd and even parity. According to this method, 0 can be inserted at a pixel location if that pixel has odd parity i.e. the number of 1's in the binary value of the pixel should be odd. Similarly, 1 can be inserted at a pixel location if that pixel has even parity i.e. the number of 1's in the binary value of pixel should be even. If the corresponding parity does not exist at a pixel location either for 0 or 1, then we make corresponding parity at that pixel location (odd parity for 0 and even parity for 1) by adding or subtracting 1 to the pixel location such that the change in the image quality should not be visible to the human visual system (HVS).

6. PVD (Pixel Value Differencing Method)

The pixel value differencing (PVD) method can successfully provide both high embedding capacity and outstanding imperceptibility for the stego-image. The pixel value differencing (PVD) method segments the cover image into non overlapping blocks containing two connecting pixels and modifies the pixel difference in each block (pair) for data embedding. A larger difference in the original pixel values allows a greater modification.

7. Hash based steganography:

In hash based steganography we use hash function to get random location in the pixels of image using two parameters i.e. Hash key and the number of chunks where we can hide the data. The message is divided into n number of chunks. The designed system has ability to hide text in an image without losing the quality of the image up to much extent. This system specifically works for efficient and secure data hiding in images to make possible large-sized data image stenography and transmission over internet.

EXISTING METHODS OF IMAGE STEGANOGRAPHY IN TRANSFORM DOMAIN:

1. Discrete cosine transform (DCT) based technique:

Discrete cosine transform (DCT): It is a process which converts a sequence of data points in the spatial domain to a sum of sine and cosine waveforms with different amplitudes in the frequency domain. The DCT is a linear transform, which maps an n-dimensional vector to set of n coefficients. A linear combination of n known basis vectors weighted with the n coefficients will result in the original vector. The known basis vectors of transforms from this class are "sinusoidal", which means that they can be represented by sinus shaped waves or, in other words, they are strongly localized in the frequency spectrum. Therefore one speaks about transformation to the frequency domain. The most popular member of this class is the Discrete Fourier Transformation (DFT).The difference between DCT and DFT is that DFT applies to complex numbers, while DCT uses just real numbers

2. Integer Wavelet Transform(IWT)

Integer Wavelet Transform presents a novel approach of building a secure data hiding technique of Steganography using integer wavelet transform along with cropping. This scheme embeds data in integer wavelet transform coefficients by using a cropping function in an 8x8 block on the cover image. The optimal pixel adjustment process is applied after embedding the message. We employed frequency domain to increase the robustness of our steganography method. Integer

wavelet transform avoid the floating point precision problems of the wavelet filter. We use cropping function and Optimal Pixel Adjustment Process to obtain an optimal mapping function to reduce the difference error between the cover and the stego-image and to increase the hiding capacity with low distortions respectively.

3. Discrete Wavelet Transform(DWT)

DWT is a hierarchical sub-band system. Wavelet transform decomposes an image into a set of band limited components which can be reassembled to reconstruct the original image without error. Since the bandwidth of the resulting coefficient sets is smaller than that of the original image, the coefficient sets can be down sampled without loss of information. Reconstruction of the original signal is accomplished by up sampling, filtering and summing the individual sub bands. For 2-D images, applying DWT corresponds to processing the image by 2-D filters in each dimension. The filters divide the input image into four non-overlapping multi-resolution coefficient sets, a lower resolution approximation image (LL) as well as horizontal (HL), vertical (LH) and diagonal (HH) detail components. The sub-band LL represents the coarse-scale DWT coefficients while the coefficient sets LH, HL and HH represent the fine-scale of DWT coefficients. To obtain the next scale of wavelet coefficients, the sub-bands are further processed until some final scale N is reached.

A Novel Steganographic Method for Gray-Level Images

The technique we propose is Grey Level Modification (GLM) Steganography technique. The images that we use are 18 rawstring images. The grey level modification algorithm is designed with the purpose of information hiding rather than image processing. The information that has to be hidden is to be in binary data format.

Grey level modification steganography is a technique to map data (not embed or hide it) by modifying the grey level values of the image pixels. GLM steganography uses the concept of odd and even numbers to map data within an image. It is a one-to-one mapping between the binary data (i.e. a bit stream with 1s and 0s) and the selected pixels in an image. From a given image a

set of pixels are selected based on a mathematical function. The grey level values of those pixels are examined and compared with the bit stream that is to be mapped in the image. Initially, the grey level values of the selected pixels (which are odd) are made even by changing the grey level by one unit. Once all the selected pixels have an even grey level it is compared with the bit stream which has to be mapped. The first bit from the bit stream is compared with the first selected pixel. If the first bit is even (i.e. 0), then the first pixel is not modified as all the selected pixels have an even grey level value. But if the bit is odd (i.e. 1), then the grey level value of the pixel is decremented by one unit to make its value odd, which then would represent an odd bit mapping. This is carried out for all bits in the bit stream and each and every bit is mapped by modifying the grey level values accordingly, gives a diagrammatic view of the technique.

Advantages of GLM:

- Chances of insertion of data are optimal.
- Easy to implement.

Disadvantages of GLM

- Vulnerable to Steganalysis.
- Not immune to noise and compression.

Algorithm for Embedding Process of GLM

i. Select pixels according to an arbitrary function g (x,y).

ii. Modify the gray level values of the selected pixels to make them even by adding one. These even gray levels will represent 0 in a bit stream.

iii. To represent 1, modify the appropriate pixel by decrementing its gray level value by 1.

iv. Thus, we can represent both 1s and 0s using pixels, which satisfy the condition of being odd or even.

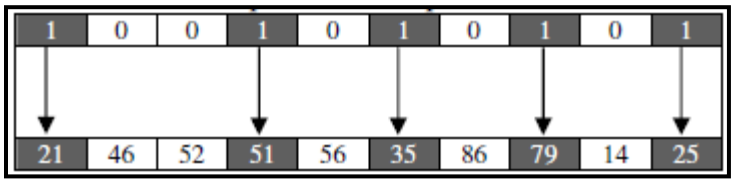


Fig. 3

Algorithm for Retrieval of Data

i. Now there is a modified image and the secret function (which is used to find pixel location). Using this function, those pixels that have been used to embed data are identified.

ii. The receiver knows the algorithm logic that an even value of gray level represents 0 while an odd value represents 1. Knowing this, the receiver can acquire the hidden data.

iii. The receiver first picks up the selected pixels and map them to the respective binary data.

iv. Odd number is mapped to one while even number is mapped to zero.

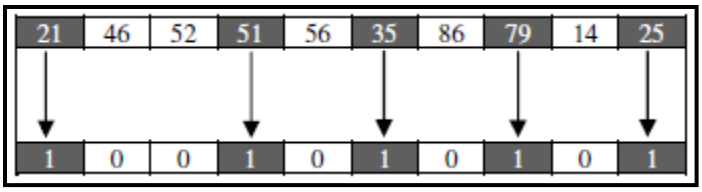


Fig. 4

A New Approach for LSB Based Image Steganography using Secret Key

This paper introduces a best approach for Least Significant Bit (LSB) based on image steganography that enhances the existing LSB substitution techniques to improve the security level of hidden iriformation. It is a new approach to substitute LSB of RGB true color image. The new security conception hides secret information within the LSB of image where a secret key encrypts the hidden iriformation to protect it from unauthorized users. In general, in LSB

methods, hidden iriformation is stored into a specific position of LSB of image. For this reason, knowing the retrieval methods, anyone can extract the hidden iriformation. In our paper, hidden iriformation is stored into different position of LSB of image depending on the secret key. As a result, it is difficult to extract the hidden iriformation knowing the retrieval methods. To hide hidden information we have to take a cover image. This cover image is divided into three matrices (Red, Green and Blue). The secret key is converted into an array of bit stream. Secret key and Red matrix are used only for decision making to replace hidden information into either Green matrix or Blue matrix. Each bit of secret key is XOR with each LSB of Red matrix. The resulting XOR value decides that the 1 bit of hidden information will be placed with either LSB of Green matrix or Blue matrix. The same process will be continued until the hidden information is finished.

Advantages of LSB:

- 100 % chances of insertion.
- Easy to implement.

Disadvantages of LSB:

- One of the major disadvantage associated with LSB method is that intruder can change the least significant bit of all the image pixels. In this way hidden message will be destroyed by changing the image quality, a little bit, i.e. in the range of +1 or -1 at each pixel position.
- Not immune to noise and compression technique.
- One of the basic techniques so more vulnerable to Steganalysis.

Encoding Algorithm:

- Select the upper left pixel of the image
- Get the red R, green G and blue B of the pixel and get

one bit K of secret key

- Take XOR(R,K)

- If LSB is 0, replace LSB of B by value of message bit
- If LSB is 1, replace LSB of G by value of message bit

- Repeat the procedure for every message bit taking consecutive pixels

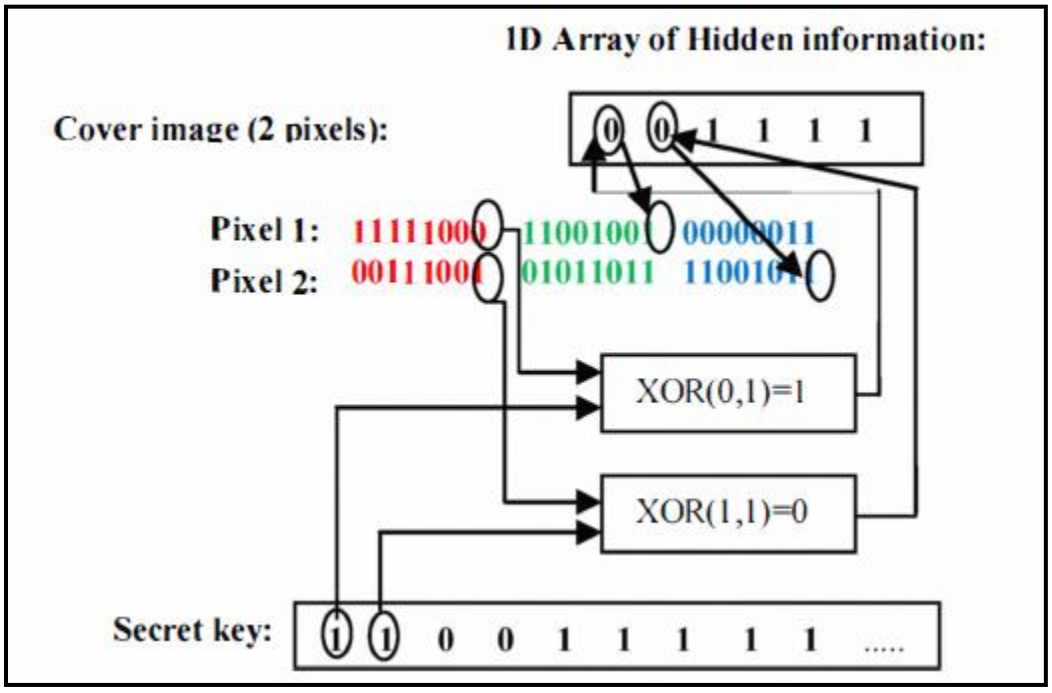


Fig. 5

Decoding Algorithm:

- Select the upper left pixel of the stego image
- Get the red R, green G and blue B of the pixel and get

one bit K of secret key

- Take XOR(R,K)
  - If LSB is 0, concatenate the output with LSB of B
  - If LSB is 1, concatenate the output with LSB of G
- Repeat the procedure for no. of message bit taking consecutive pixels

A Hash-Based Approach for Image Steganography

In this, paper we propose a novel hash-based approach for color image steganography. As, the available approaches for color image steganography are using chaos-based and symmetric-key based cryptographic algorithms are not efficient and good for bulky data. However, the hash based algorithms based approaches are considerably better in terms of providing better speed but

these approaches are vulnerable in terms of providing security due to inherent flaws caused by used checksum approach. The key reason of vulnerability is that the used algorithms in such approaches such as MD5 and SHA-2 have flaws. In our approach, we purpose the use of perfect hashfunction algorithm to provide a secure and fast approach for colour image steganography. We also present a prototype tool in this paper that is implementation of the presented approach and is also a proof of concept. Another contribution of the approach is that the presented approach can be used for coding data in any type of colour images such as bmp, jpeg, g and tifT as other available approaches are me format specific. The results of the initial experiments are very encouraging and support not only the used approach but also uphold the potential of the presented approach in general.

Advantages:

- This system specifically works for efficient and secure data hiding in images to make possible large-sized data encryption and transmission over internet.

Disadvantage:

- If hash function used is not perfect hash function, then we need to have collision resolution, in order to avoid loss of message.

Encoding Algorithm:

a) Input a text (.txt) file containing textual data and input (Jpg, .gii: .bmp or .tiff) image.
b) Read text file, tokenize the text and make chunks of the text of 3 characters each and store each chunk of data in an array-list (l_c).Total count of data chunks are represented as n.
c) Generate a random number that is used as a hash key and hash-key is represented using h.
d) The hash-function H uses the hashkey (h) and total number of chunks (n) to generate a pattern i.e. sequence of numbers (hash-values) those are position of the pixels where data will be stored.
e) The generated pattern (containing sequence of numbers) is stored in an array-list (l_p).
f) First chunk from l_c and l_p are read. String stored in Ie is read and tokenized. The ASCII value of 1^{st} token of chunk $l_c[i]$ is replaced with the red-byte of the $l_c[i]$, similarly, ASCII

value of 2nd token of chunk $l_c[i]$ is replaced with the green-byte of the $l_p[i]$, and the ASCII value of 3rd token of chunk $l_p[i]$ is replaced with the blue-byte of the $l_p[i]$.

g) The output is the image containing coded data and a hash-key (h) that is used to decode data.

Decoding Algorithm:

a) Input the (.jpg, .gii: .bmp or .tiff) image that contains that coded information and the hash key(h) that was actually used to code data.

b) The input hash-key (h) is used with the hash function (H) to generate sequence of numbers as an array-list (l_p) and these numbers are actually position of the pixels where data was be stored. Here the hash-function (H) generates specifically same pattern of random numbers for a hash key (H) those were generated at the time of coding.

c) Each value from the generated patterns represent index of a pixel where the data is saved. Values of read, green and blue byte are read at I_p are read. As each byte contains an ACSII value of a character, the read ASCII value is converted to a character and each character is written to a text file in sequence it is read from the image.

d) The output is a text file that contains the decoded data from image.

Discrete Wavelet Transform Tchnique for Image Steganography

- DWT is simple and fast transformation approach that translates an image from spatial domain to frequency domain.The DWT technique divides the image into sub-bands-high frequency and low frequency .DWT is used for digital images. Many DWTs are available. Depending on the application appropriate one should be used. The simplest one is haar transform. To hide text message integer wavelet transform can be used. When DWT is applied to an image it is decomposed into 4 sub bands: LL, HL, LH and HH. LL part contains the most significant features. So if the information is hidden in LL part the stego image can withstand compression

or other manipulations. But sometimes distortion may be produced in the stego image and then other sub bands. The decomposition of image by 2 levels of 2D - DWT is shown in Figure 1

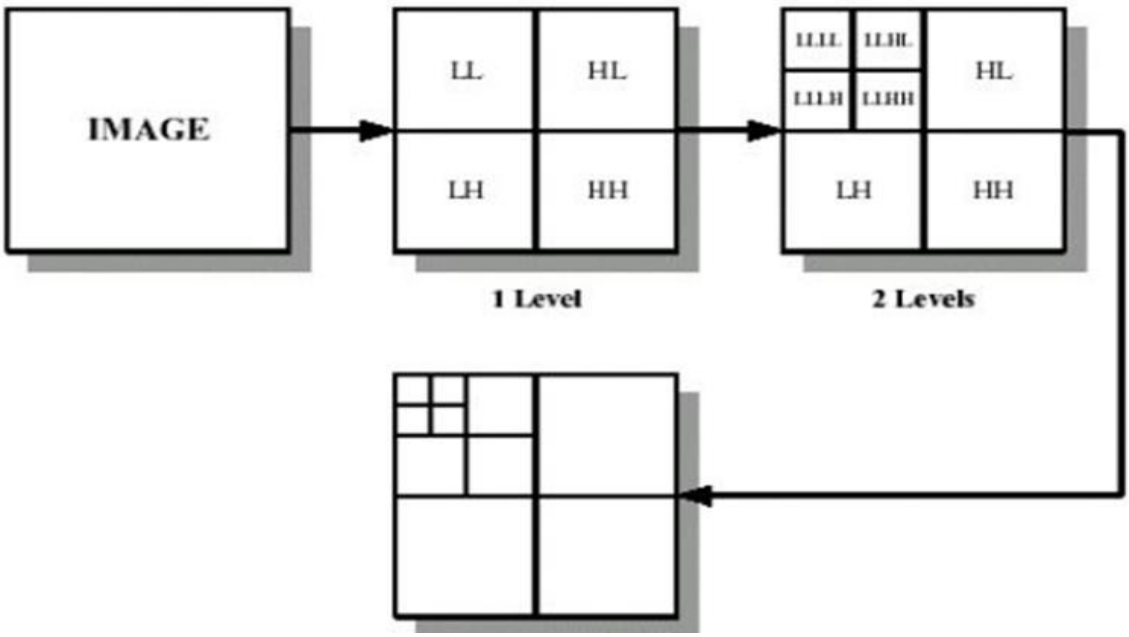


Fig. 6

Advantages:

- Higher flexibility: Wavelet function can be freely chosen
- Transformation of the whole image→ introduces inherent scaling

Disadvantage:

- Lower quality than JPEG at low compression rates
- Longer compression time
- The cost of computing DWT is higher

Encoding Algorithm:

- Read cover image and store its pixel values in an array.
- Read secret key and convert it in binary.
- Perform the DWT on the cover image (pixel array) and convert the image into four sub bands.
- Embed the secret key by changing the LSB of the each sub band
- Generate the stego image.

Decoding Algorithm:

- Read stego image and store its pixel values in an array.
- Calculate the IDWT.
- Retrieve the message

PROBLEM SOLUTION AND WORK DONE

BLOCK DIAGRAM

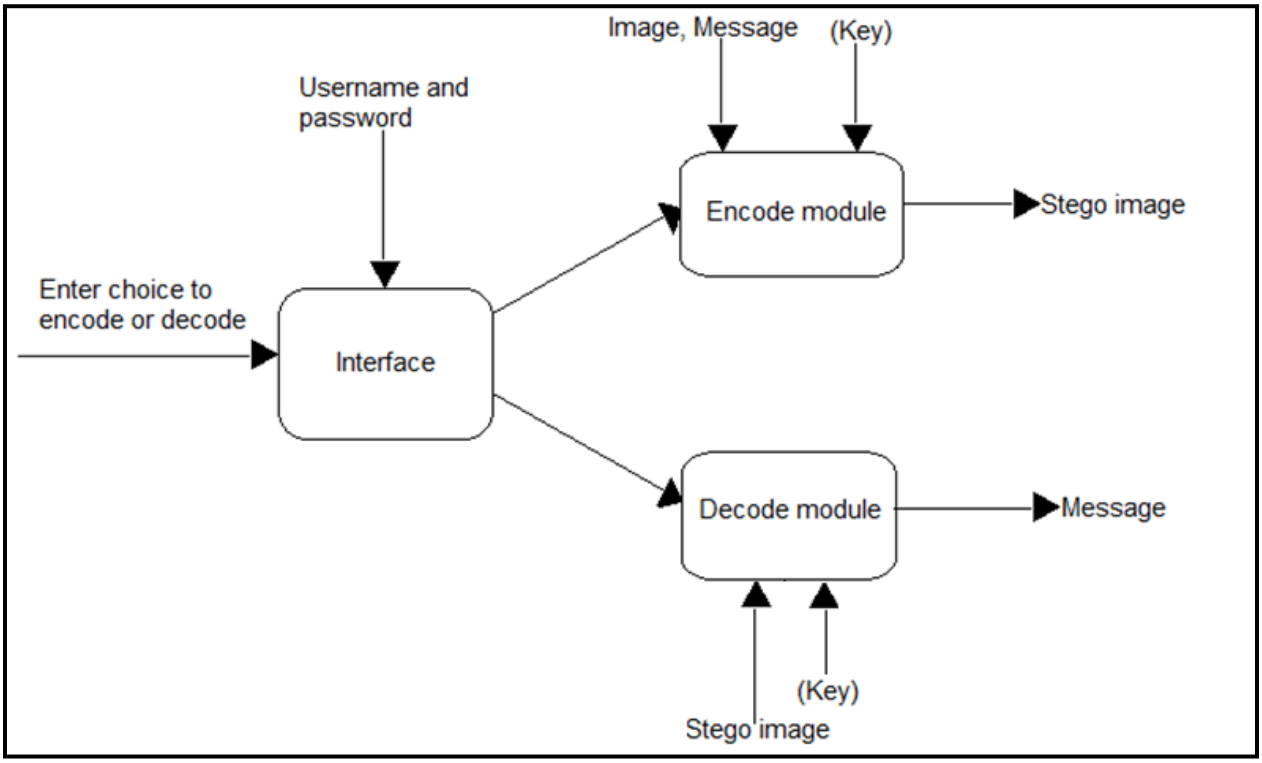


Fig. 7

DFD-LEVEL 0

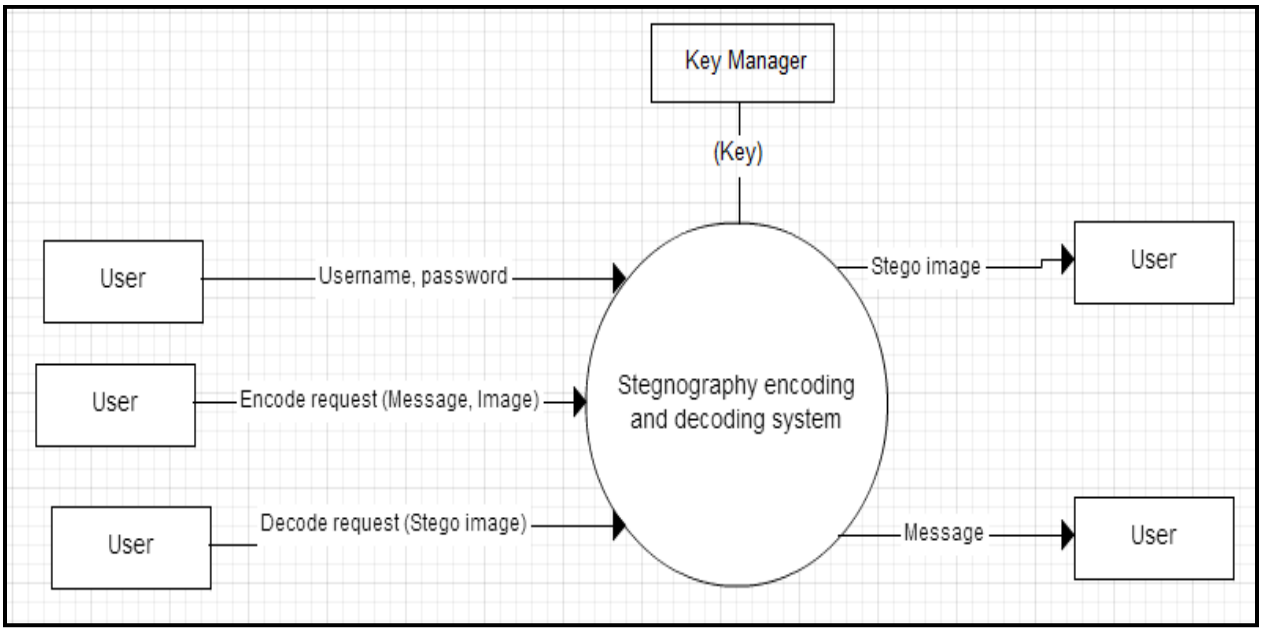


Fig. 8

DFD-LEVEL 1

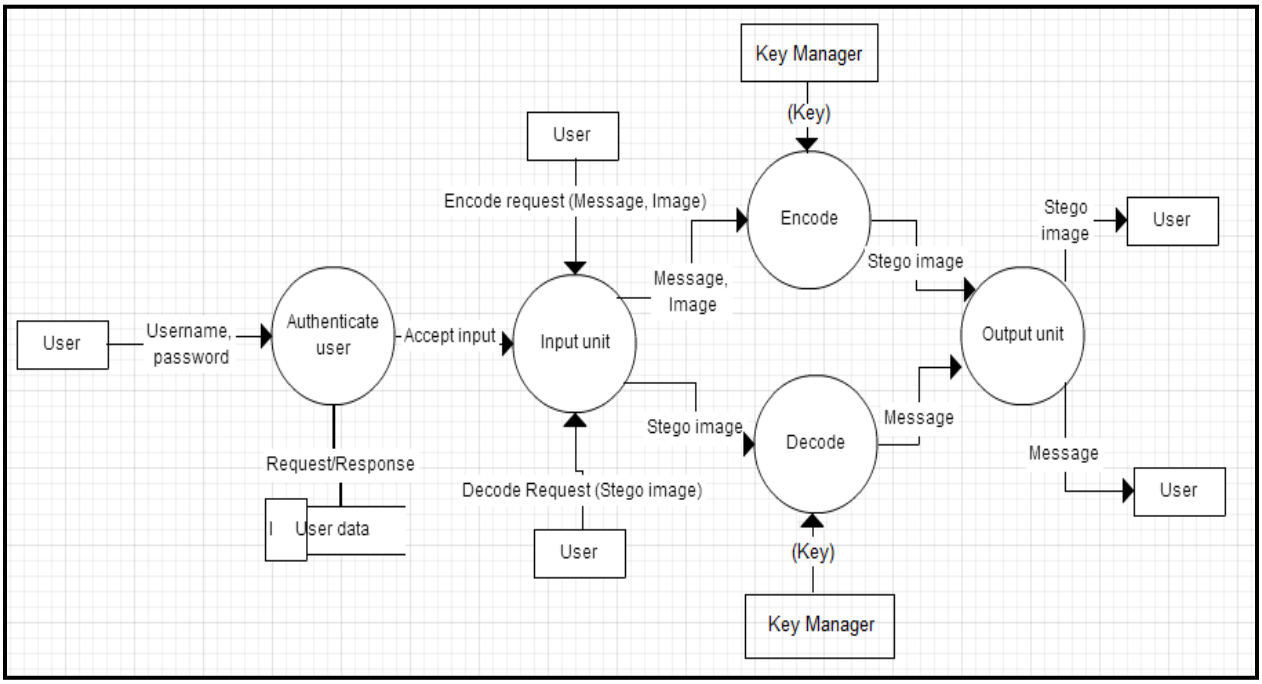


Fig. 9

JAVA CODES:

GreyLevelEn.java (Grey Level Modification – Encoding And Decoding)

```
class GreyLevelEn extends Technique
{
        BufferedImage image1,image2;

        GreyLevelEn(BufferedImage b)
        {
                image1=b;
                 image2=new
BufferedImage(image1.getWidth(),image1.getHeight(),BufferedImage.TYPE_INT_ARGB);
        }
        public void encode(String s)
{
                 int len=s.length(),w=0;
                 int width = image1.getWidth();
                 int height = image1.getHeight();
                 int[][] res = new int[height][width];
                 try
                 {   res=GetRGB(image1);}
        catch(IOException e){}

                 for(int i=0;i<height;i++)
                         for(int j=0;j<width;j++)
                                 image2.setRGB(j,i,res[i][j]);
                 for(int i=0;i<len;i++)
                 {
```

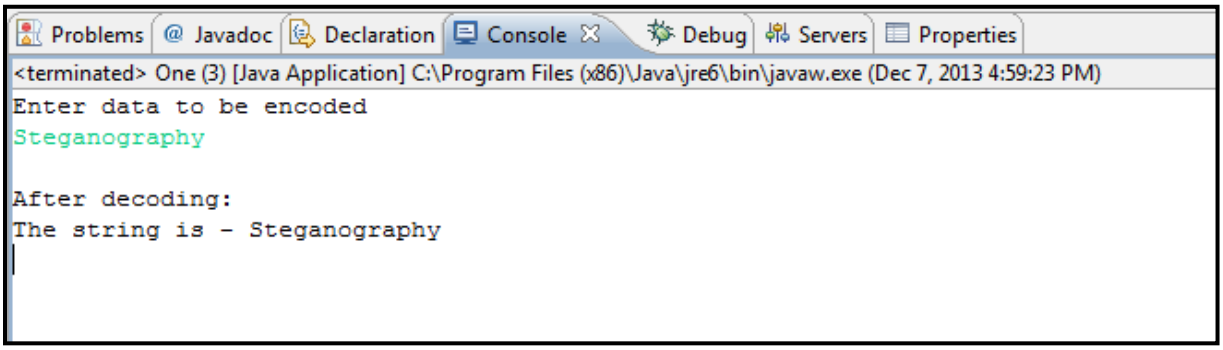
```
                w=((2*i)+5)%width;
if(res[i][w]%2!=0)
res[i][w]+=1;
                if(s.charAt(i)=='1')
res[i][w]-=1;
                image2.setRGB(w,i,res[i][w]);                                    }
}
public String decode(int slen)
{
        int w;
        int width = image1.getWidth();
        int height = image1.getHeight();
        String s=new String();
        int res[][]=new int[height][width];
        try
        {   res=GetRGB(image1);}
catch(IOException e){}

        for(int i=0;i<slen;i++)
        {
                w=((2*i)+5)%width;
if(res[i][w]%2!=0)
                        s=s+'1';
                else
                        s=s+'0';
        }
        return s;
}
public BufferedImage getImage()
{
        return image2;
```

```
    }

    public void setImage(BufferedImage i)
    {
        image1=i;
    }
}
```

OUTPUT:



LSBen.java(LSB –Encoding And Decoding)

```
class LSBen extends Technique
{
    BufferedImage image1,image2;
```

```
        BitAlpha ba=new BitAlpha();
        String key=new String();
        int index=2;
        PRNG pr=new PRNG();

        LSBen(BufferedImage b)
        {
                image1=b;
                image2=new
BufferedImage(image1.getWidth(),image1.getHeight(),BufferedImage.TYPE_INT_ARGB);
        }

        public void encode(String s)
{
                int len=s.length();
                int width = image1.getWidth();
                int height = image1.getHeight();
                int[][] res = new int[height][width];
                int[][][]result=new int[height][width][4];
                int pixel;
                int a;
                byte[] nb=new byte[1000];
                nb=pr.keygen();
                try
                {   res=GetRGB(image1);}
        catch(IOException e){}
                try
                {   result=GetRGBVal(image1);
                }
             catch(IOException e){}
```

```
for(int i=0;i<height;i++)
        for(int j=0;j<width;j++)
                image2.setRGB(j,i,res[i][j]);
int k=0;
int m=0;

for(int i=0;i<height;i++)
{
        for(int j=0;j<width;j++)
        {
                key=intToBinary(nb[m]);
                int x=result[i][j][1];
                int y=0;
                try{
                y=Integer.parseInt(key.substring(index,index+1));}
                catch(StringIndexOutOfBoundsException e)
                {}
                if((x^y)==0)//making all chosen values even
                {
                        a=(res[i][j])&0xFF;
                        if(a%2==0 && s.charAt(k)=='1')
                                a=res[i][j]+1;
                        else if(a%2==1 && s.charAt(k)=='0')
                                a=res[i][j]-1;
                        pixel = (result[i][j][0]<<24) + (result[i][j][1]<<16) +
(result[i][j][2]<<8) + a;
                }
                else
                {
                        a=(res[i][j]>>8)&0xFF;
                        if(a%2==0 && s.charAt(k)=='1')
```

```
                                        a=a+1;
                                    else if(a%2==1 && s.charAt(k)=='0')
                                        a=a-1;

                                    pixel = (result[i][j][0]<<24) + (result[i][j][1]<<16) +
(a<<8) + (result[i][j][3]);
                                }
                                image2.setRGB(j,i,pixel);
        k++;
                                m++;
                                if(m==1000)
                                    m=0;
                                if(k>=len)
                                    break;
                        }
                        if(m==1000)
                            m=0;
                        if(k>=len)
                            break;
                            }
                try
                {
                    ImageIO.write(image2,"png",new File("Output.png"));
                }
                catch(IOException e){
                    System.out.println("Unable to write file!!");
                }
        }
        public String decode(int slen)
        {
            int width = image1.getWidth();
```

```
int height = image1.getHeight();
String s=new String();

int result[][][]=new int[height][width][4];
byte[] nb=new byte[2000];
nb=pr.keygen();
try
{   result=GetRGBVal(image1);
}
catch(IOException e){}
int k=0;
int m=0;
for(int i=0;i<height;i++)
{
        for(int j=0;j<width;j++)
        {
                key=Integer.toBinaryString(nb[m]);
                int x=result[i][j][1];
                int y=0;
                try
                {
                y=Integer.parseInt(key.substring(index,index+1));
                }
                catch(StringIndexOutOfBoundsException e){}
                if((x^y)==0)
                {
                        if(result[i][j][3]%2==0)
                                s=s+'0';
                        else
                                s=s+'1';
                }
```

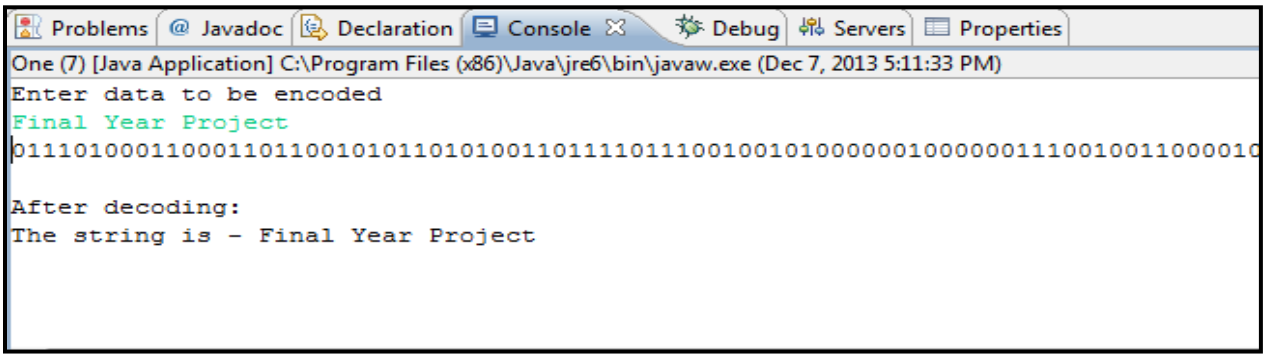
```
                    else
                    {
                        if(result[i][j][2]%2==0)
                            s=s+'0';
                        else  s=s+'1';
                    }
                    m++;
                    if(m>=1000)
                        m=0;
                    k++;
                    if(k>=slen)
                        break;
                    m++;
                    if(m>=1000)
                        m=0;
            }
                if(m>=1000)
                    m=0;
                if(k>=slen)
                    break;
                }
            return s;
    }
    public BufferedImage getImage()
    {
            return image2;
    }
    public void setImage(BufferedImage i)
    {
            image1=i;
    }
```

```
public String intToBinary(int i)
{
	int j;
	if(i<0)
		j=-1*i;
	else
		j=i;
	String k=Integer.toBinaryString(j);
	int x=k.length();
	if(x==1)
		k="0000000"+k;
	else if(x==2)
		k="000000"+k;
	else if(x==3)
		k="00000"+k;
	else if(x==4)
		k="0000"+k;
	else if(x==5)
		k="000"+k;
	else if(x==6)
		k="00"+k;
	else if(x==7)
		k="0"+k;
	if(i<0)
	{
		x=k.length();
		for(int y=0;y<x;y++)
		{
			if(y>0)
				{
					if(k.charAt(y)=='0')
```

```
                        k=k.substring(0,y)+'1'+k.substring(y+1,x);
                else
                        k=k.substring(0,y)+'0'+k.substring(y+1,x);
        }
        else
        {
                if(k.charAt(y)=='0')
                        k='1'+k.substring(y+1,x);
                else
                        k='0'+k.substring(y+1,x);
        }
}
char carry='0';
if(k.charAt(x-1)=='0')
        k=k.substring(0,x-1)+'1';
else
        {
        k=k.substring(0,x-1)+'0';
        carry='1';
        }
for(int y=x-2;y>=0;y--)
{
        if(k.charAt(y)=='0' && carry=='1' )
        {
                k=k.substring(0,y)+'1'+k.substring(y+1,x);
                carry='0';
        }
        else if(k.charAt(y)=='1' && carry=='0')
        {
                k=k.substring(0,y)+'1'+k.substring(y+1,x);
                carry='0';
```

```
                    }
                    else if(k.charAt(y)=='0' && carry=='0')
                    {
                        k=k.substring(0,y)+'0'+k.substring(y+1,x);
                        carry='0';
                    }
                    else
                    {
                        k=k.substring(0,y)+'0'+k.substring(y+1,x);
                        carry='1';
                    }
                }
            }
            return k;
        }
}
```

OUTPUT:



Hashen.java (Hash-Based Encoding And Decoding)

```
class Hashen extends Technique
{
	BufferedImage image1,image2;
	BitAlpha ba=new BitAlpha();
	String key=new String("1010111001");
	int index;
	Hashen(BufferedImage b)
	{
		image1=b;
		image2=new
BufferedImage(image1.getWidth(),image1.getHeight(),BufferedImage.TYPE_INT_ARGB);
	}

	public void encode(String s)
	{
		int len=s.length();
		int width = image1.getWidth();
```

```
int height = image1.getHeight();
int[][] res = new int[height][width];   //to store pixel values
int[][][]result=new int[height][width][3];
int pixel;
int a,i;
try
{   res=GetRGB(image1);}
catch(IOException e){}
try
{   result=GetRGBVal(image1);
}
catch(IOException e){}

for( i=0;i<height;i++)
        for(int j=0;j<width;j++)
                image2.setRGB(j,i,res[i][j]);

int h1[]=new  int[100];
h1=hashh.pr(2);
for(i=0;i<100;i++)
{
        if(h1[i]<0)
                h1[i]=-1*h1[i];
}
int h2[]=new  int[100];
h2=hashh.pr(1);
for(i=0;i<100;i++)
{
        if(h2[i]<0)
```

```
                    h2[i]=-1*h2[i];
            }
            int k=0;
            index=0;
for( i=0;i<100;i++)
{
            for (int j=0;j<100;j++)
            {
                    int c=h1[i];
                    int d=h2[j];
                    int p=Integer.parseInt(s.substring(index,index+1));
                    int q=Integer.parseInt(s.substring(index+1,index+2));
                    int r=Integer.parseInt(s.substring(index+2,index+3));

                    int s1=result[c][d][1];
                    if(p==1 && s1%2==0)
                                s1=s1+1;
                        else if(s1%2==1 && p==0)
                                s1=s1+1;
                    int t=result[c][d][2];
                    if(q==1 && t%2==0)
                                t=t+1;
                        else if(t%2==1 && q==0)
                                t=t+1;
                    int u=result[c][d][3];
                    if(r==1 && u%2==0)
                                u=u+1;
                        else if(u%2==1 && r==0)
                                u=u+1;
```

```
                        pixel = (result[c][d][0]<<24) + (s1<<16) + (t<<8) + u;
                image2.setRGB(j,i,pixel);
                index=+3;
                if(index>=len)
                        break;
            }
                        if(index>=len)
                break;
        }
        try
        {
            ImageIO.write(image2,"png",new File("Output.png"));
        }
        catch(IOException e){
            System.out.println("Unable to write file!!");
        }
}
        public String decode(int slen)
        {
            int width = image1.getWidth();
            int height = image1.getHeight();
            String s=new String();
            String z=new String();
            int res[][]=new int[height][width];
            int k=0;
            int result[][][]=new int[height][width][4];
            try
            {   res=GetRGB(image1);
            }
```

```
        catch(IOException e){}
        try
        {   result=GetRGBVal(image1);
}
        catch(IOException e){}
        int h1[]=new  int[100];
        h1=hashh.pr(2);
        for(int i=0;i<100;i++)
        {
                if(h1[i]<0)
                        h1[i]=-1*h1[i];
                                }
        int h2[]=new  int[100];
        h2=hashh.pr(1);
        for(int i=0;i<100;i++)
        {
                if(h2[i]<0)
                        h2[i]=-1*h2[i];
                                }
        for(int i=0;i<100;i++)
        {
                for(int j=0;j<100;j++)
                {
                        int c=h1[i];
                        int d=h2[j];
                        if(result[c][d][1]%2==0)
                                s=s+'0';
                        else
                                s=s+'1';
```

```
                                        if(result[c][d][2]%2==0)
                                                s=s+'0';
                                        else
                                                s=s+'1';
                                        if(result[c][d][3]%2==0)
                                                s=s+'0';
                                        else
                                                s=s+'1';
                                                        k=+3;
                                if(k>=slen)
                                        break;
                        }
                                if(k>=slen)
                                        break;
                        }
                                        return s;
        }
                public BufferedImage getImage()
        {
                return image2;
        }
        public void setImage(BufferedImage i)
        {
                image1=i;
        }
}
```

OUTPUT:

```
Problems  Javadoc  Declaration  Console  Debug  Servers  Properties
One (8) [Java Application] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (Dec 8, 2013 1:21:52 PM)
Enter data to be encoded
stego
01101111011001110110010101110100011100 11


After decoding:
The string is - tzdW3
```

Techniquedwt.java (Discrete Wavelet Transfor – The harr function that divide the image)

```
abstract class Techniquedwt
{
        protected BufferedImage image1,image2;
        protected int[][] GetRGB(BufferedImage image) throws IOException
        {
                int w = image.getWidth();
                int h = image.getHeight();
                int[][] pixel = new int[h][w];
                int[][] mat=  new int[w][h];
                                for (int row = 0; row < h; row++)
                {
```

```
                for (int col = 0; col < w; col++)
                {
                        pixel[row][col] = image.getRGB(col, row);
                }
     int   nc2 = h/2;
for (int i=0; i<w; i++) {
 for (int j=0; j<h; j+=2) {
  int  j1 = j+1;
 int    j2 = j/2;
   mat[i][j2] = pixel[i][j] + pixel[i][j1];
   mat[i][nc2+j2] = pixel[i][j] - pixel[i][j1];
 }
      }
          for (int i=0; i<w; i++)
   for (int j=0; j<h; j++)
    pixel[i][j] = mat[i][j];
 int nr2 = w/2;
 for (int i=0; i<w; i+=2) {
   for (int j=0; j<h; j++) {
    int i1 = i+1;
   int  i2 = i/2;
    mat[i2][j] = pixel[i][j] + pixel[i1][j];
    mat[nr2+i2][j] = pixel[i][j] - pixel[i1][j];
   }
 }
 for (int i=0; i<w; i++)
     for (int j=0; j<h; j++)
             pixel[i][j] = mat[i][j];
   }
   return pixel;
                }
```

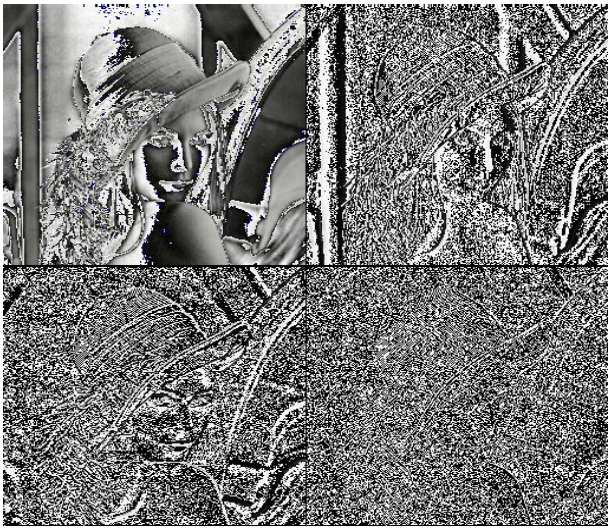
```
protected int[][][] GetRGBVal(BufferedImage image) throws IOException
{
        int w = image.getWidth();
        int h = image.getHeight();
        int[][] result1 = new int[h][w];
        int[][] mat=  new int[w][h];
        for (int row = 0; row < h; row++)
        {
            for (int col = 0; col < w; col++)
            {
                result1[row][col] = image.getRGB(col, row);
            }
                        int   nc2 = h/2;
          for (int i=0; i<w; i++) {
           for (int j=0; j<h; j+=2) {
            int  j1 = j+1;
          int    j2 = j/2;
            mat[i][j2] = result1[i][j] + result1[i][j1];
            mat[i][nc2+j2] = result1[i][j] - result1[i][j1];
           }
                    }

              for (int i=0; i<w; i++)
            for (int j=0; j<h; j++)
              result1[i][j] = mat[i][j];
           int nr2 = w/2;
           for (int i=0; i<w; i+=2) {
            for (int j=0; j<h; j++) {
              int i1 = i+1;
            int  i2 = i/2;
              mat[i2][j] = result1[i][j] + result1[i1][j];
```

```
                    mat[nr2+i2][j] = result1[i][j] - result1[i1][j];
                  }
                }
                for (int i=0; i<w; i++)
                    for (int j=0; j<h; j++)
                            result1[i][j] = mat[i][j];

                  }
            int width = image.getWidth();
            int height = image.getHeight();
            int[][][] result2 = new int[height][width][4];
                            for (int row = 0; row < height; row++)
            {
                  for (int col = 0; col < width; col++)
                  {
                          result2[row][col][0] = (result1[row][col] >> 24) & 0xFF;
                            result2[row][col][1]= (result1[row][col] >> 16) & 0xFF;
                            result2[row][col][3] = result1[row][col] & 0xFF;
                            result2[row][col][2]=(result1[row][col] >> 8) & 0xFF;
                  }
            }
            return result2;
}
}
```

OUTPUT:

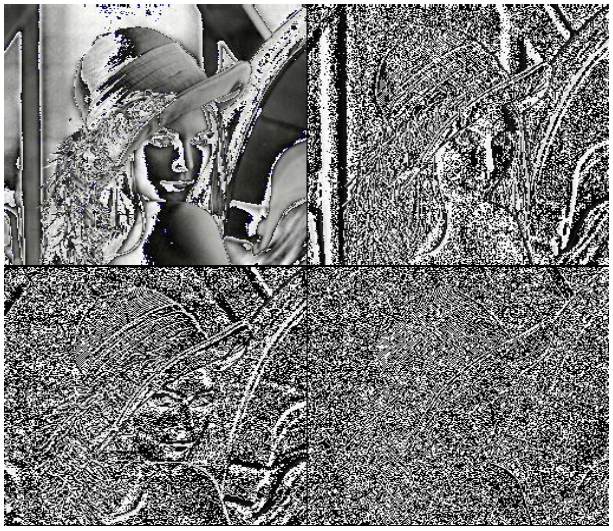


The inverse Function

```
for (int row = 0; row < height; row++)
                {
                        for (int col = 0; col < width; col++)
                        {
                                pixel1[row][col] = image2.getRGB(col, row);
                        }
                }
    int i,j;
                    int  nr2 = width/2;
              for (i=0; i<nr2; i++) {
                for (j=0; j<height; j++) {
                  int i2 = i*2;
                  result1[i2][j]   = (pixel1[i][j] + pixel1[nr2+i][j])/2;
                  result1[i2+1][j] = (pixel1[i][j] - pixel1[nr2+i][j])/2;
                }
              }
                        for (i=0; i<width; i++)
                for (j=0; j<height; j++)
                  pixel1[i][j] = result1[i][j];
```

```
int  nc2 = height/2;
 for (i=0; i<width; i++) {
  for (j=0; j<nc2; j++) {
    int j2 = j*2;
    result1[i][j2]   = (pixel1[i][j] + pixel1[i][nc2+j])/2;
    result1[i][j2+1] = (pixel1[i][j] - pixel1[i][nc2+j])/2;
  }
 }
          for (i=0; i<width; i++)
  for (j=0; j<height; j++)
    pixel1[i][j] = result1[i][j];
              int x,y;
                  for ( x=0;x<width-1;x++)
 {
    for( y=0;y<height-1;y++)
    {
       image.setRGB(x,y,pixel1[x][y]);
    }
 }
              try
    {
              ImageIO.write(image,"png",new File("Output.png"));
    }
    catch(IOException e){
              System.out.println("Unable to write file!!");
    }
  }
```

OUTPUT:



GUI (Graphical User Interface)

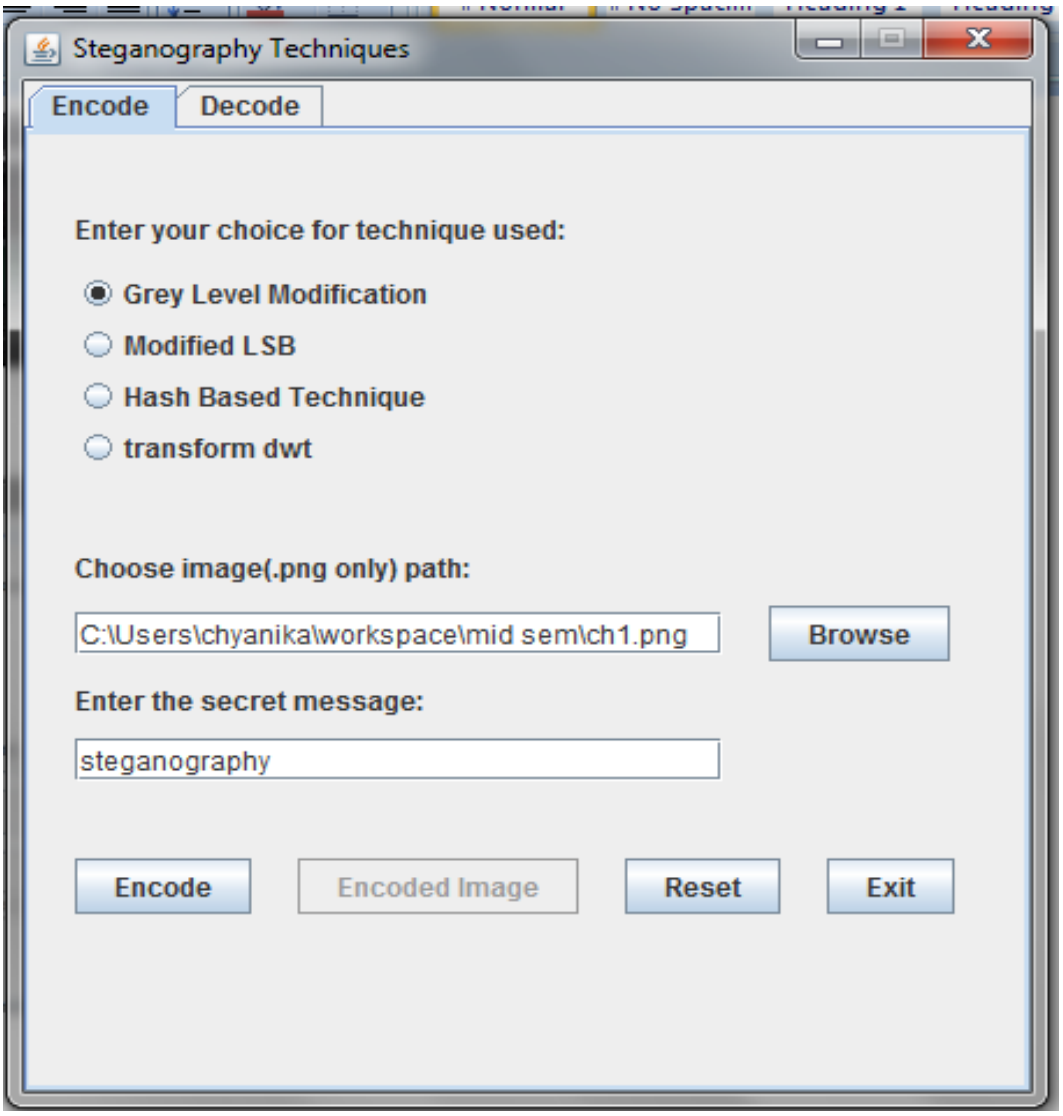


Fig. 10

Conclusion And Future Scope

This project mainly concentrated on embedding the data into an image. A steganographic application which embed the data into the image. Normally after embedding the data into the image ,the image lose its resolution ,in this approach the image remained unchanged.

In the present world ,the data transfer using internet is rapidly growing because it is easier and faster to transfer the data to the destination. Many individuals and business people use to transfer business documents ,important information using internet .Security is important issue while transferring the data using internet because any unauthorized individual can hack the data and make useless or obtain information un-intended to him.

The future work on this project is to improve the hash function so that the length of the message can be increased that has to be encrypted .Also project can be extended to a level such that it can be used for different types of image format. Also various other techniques such as Discrete Cosine Transform (DCT)and Integer Wavelet Transform(IWT) can be implemented and the effects of various attacks can be estimated.

REFERENCES

Article

[1] S. M. Masud Karim et al.,"A New Approach for LSB Based Image Steganography using Secret Key", Proceedings of 14th International Conference on Computer and Information Technology, Dhaka, Bangladesh, 2011

[2] Rubata Riasat1 et al. "A Hash-Based Approach for Colour Image Steganography", IEEE,978-1-61284-941-6/11 ,2011

[3] M.B. Ould MEDENI and El Mamoun SOUIDI, " A Novel Steganographic Method for Gray-Level Images With four-pixel Differencing and LSB Substitution" Laboratoire de Math´ematiques, Informatique et Applications, Morocco

[4] R.Z. Wang, C.F. Lin, J.C. Lin, Image hiding by optimal LSB substitution and genetic algorithm, Pattern Recognit. 34 (3) (2001) 671-683.

[5] I.C. Lin, Y.B. Lin, C.M. Wang, Hiding data in spatial domain images with distortion tolerance, Comput. Stand. Inter. 31 (2) (2009) 458-464.

[6] Kenny Hunt: A Java Framework for Experimentation with Steganography,ACM SIGCSE Bulletin Proceedings Volume 37 Issue 1, 2005. pp.282–286.

[7] Kamaldeep, "Image Steganography Techniques in Spatial Domain, their Parameters and Analytical Techniques: A Review Article", UIET, M.D.University, Rohtak,Volume 2 Issue 5,2013, ISSN: 2278-7844

[8] Y. Mao and M. Wu, "A joint signal processing and cryptographic approach to multimedia encryption," IEEE Transactions on Image Processing, 15(7X2006)2061-2075.

[9] M. Zeghid, M Machhout, L Khriji, A Baganne, and R Tourki, "A modified AES based algorithm for image encryption," International Journal of Computer Science and Engineering, 1(IX2006) 70-75.

[10] Ahmad and AS. Das, "Hardware implementation analysis of SHA -256 and SHA -512 algorithms on FPGAs," Computers & Electrical Engineering, 31(6X2005)345-360.

[11] David Talbot, Thorsten Brants, "Randomized Language Models via Perfect Hash Functions", in proceedings of ACLHLT 2008, Ohio, USA, June 2008, pp:505-513

[12] M. Hossain, S.A. Haque, F. Sharmin, "Variable Rate Steganography in Gray Scale Digital Images Using Neighborhood Pixel Iriformation", Proceedings of 2009

[13] Marvel, L.M., Boncelet Jr., C.G. & Retter, C., “Spread Spectrum Steganography”,IEEE Transactions on image processing, 8:08, 1999

[14] CSE, Sri Vishnu Engg. ,EEE Dept, KSR College of Technology, Tiruchengode, Tamilnadu, "The Modified Haar Wavelet Transformation in Image Compression "2007

[15] Mei Jiansheng1, Li Sukang1 and Tan Xiaomei2 1 Nanchang Power Supply Company,Nanchang, China "A Digital Watermarking Algorithm Based On DCT and DWT" Proceedings of 2009

Book

[1] Thomas R Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein Introduction to Algorithms, Second Edition MIT Press and McGraw-Hill, 2001. Section 11.5: Perfect hashing, pp. 245-249.

Online

[1] T. Morkel, J.H.P. Eloff, M.S. Olivier, "An overview/ image steganography", http://mo.co.za/openistegoverview.pdf

[2] Sellars, Duncan, "Introduction to Steganography", http://www.cs.uct.ac.za/courses/ CS400WINIS/papers99/dsellars/stego.html