INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN 2278 – 0211 (Online)

Colour Image Steganography Using LZW Compression and Fisher-Yates Shuffle Algorithm

Tejeshwar G.
M Tech (Software Engineering), Department of Information Science and Engineering
PES Institute of Technology, Bangalore, India

Abstract:
Steganography is the art and science that involves communicating secret data in an appropriate multimedia carrier, e.g., image, audio and video files, to conceal the very existence of the embedded data. This report presents a novel technique for colour image Steganography based on LZW compression and Fisher – Yates shuffle algorithm. Two 24-bit colour images are used as cover image and secret image respectively. Fisher-Yates shuffle algorithm is implemented to shuffle the pixels of secret image and LZW encoding is performed over the shuffled secret image/message before embedding and each LZW encoded bit of secret image/message is embedded inside the cover image by altering the last two least significant bit (LSB) of each of the pixel's intensities of cover image. LZW algorithm is organized around a translation table, referred to as dictionary that maps input characters into fixed-length codes. The LZW dictionary has a prefix property in that for every string in dictionary, its prefix string is also in the dictionary. A two-level compression performs compression of bit strings into 2-bit string. Two-level decompression performs decompression of 2-bit string into full length bit strings. Furthermore, satisfactory security is maintained since the secret image cannot be extracted without LZW dictionary, Fisher – Yates shuffle algorithm and two-level compression/decompression modules.

Keywords: *Image Steganography, LZW Compression, Fisher – Yates Shuffle Algorithm, LSB, PSNR*

1. Introduction

Steganography is the art of hiding information in ways that prevent the detection of hidden messages. Steganography, derived from Greek, literally means "covered writing" (Greek words "stegos" meaning "cover" and "gratia" meaning "writing"). It comes under the assumption that if the feature is visible, the point of attack is evident, thus the goal here is always to conceal the very existence of the embedded data. Therefore Steganography gets a role on the stage of information security [1] [2].
Steganography is the science that involves communicating secret data in an appropriate multimedia carrier, e.g., image, audio and video files.

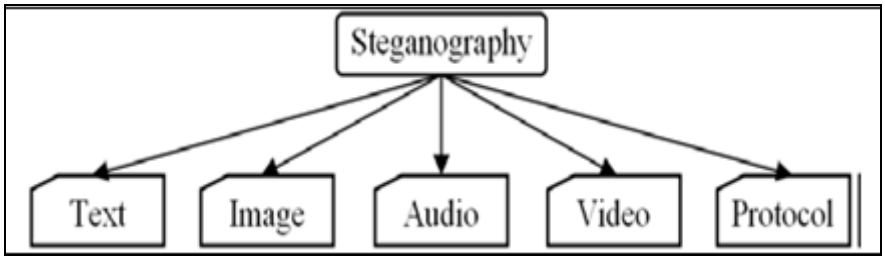

Figure 1: Steganography Communicate Secret Data in Multiple Carrier

The media with or without hidden information are called stego media and cover media, respectively. Steganography can meet both legal and illegal interests, e.g., civilians may use it for protecting privacy while terrorists may use it for spreading terroristic information [3].
Steganography and cryptography are cousins in the spy craft family. Cryptography scrambles a message so it cannot be understood. Steganography hides the message so it cannot be seen. A message in cipher text, for instance, might arouse suspicion on the part of the recipient while an "invisible" message created with Steganographic methods will not. [2]

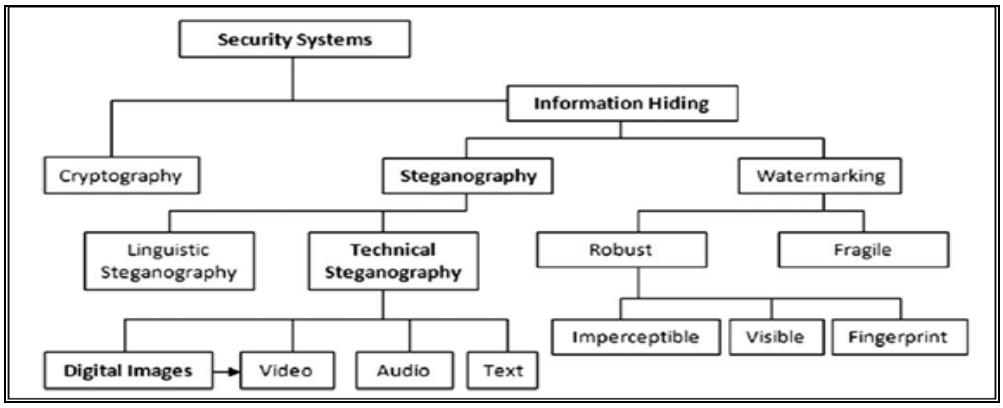

Figure 2:.The Different Disciplines Of Information Hiding
The Arrow Indicates an Extension and Bold Face Indicates the Focus of This Study

Another technology that is closely related to Steganography is watermarking. Watermarking is a protecting technique which protects (claims) the owner's property right for digital media (i.e. Images, music, video and software) by some hidden watermark. Therefore, the goal of Steganography is the secret messages while the goal of watermarking is the cover object itself.
The advantages of least-significant-bit (LSB) Steganographic data embedding are that it is simple to understand, easy to implement, and it produces stego-image that is almost similar to cover image and its visual infidelity cannot be judged by naked eyes.
A good technique of image Steganography aims at three aspects:

- First one is capacity (the maximum data that can be stored inside the cover image).
- Second one is the imperceptibility (the visual quality of stego-image after data hiding).
- Last one is robustness.

The main objective of using image Steganography is to communicate securely in such a way that the true message is not visible to the observer. A novel technique is used for image Steganography based on LZW compression algorithm and Fisher-Yates shuffle algorithm. LZW algorithm is mainly used to encode and compress lossless (bitmap, png format) images so as to make them fit inside the cover image. Fisher – Yates algorithm shuffles the secret image pixel's location so as to make the secret image look scrambled, an added feature of security. This novel technique is applicable for 24-bit depth colour images.

2. Related Works
Many different researchers employed different techniques for the purpose of secured secret image embedding. Following are the few related works carried out by various research groups:

2.1. Steganography by Embedding Message inside an Image
Steganography can be accomplished by simply feeding into a windows OS command window, e.g., the following code:
C:\> Copy Cover_Image.Jpg /B + Message.Txt /B Stego Image.Jpg
However, this simple technique would not resist any kind of editing to the stego-image nor any attacks by steganalysis experts [1].

2.2. Steganography in Spatial Domain
In spatial domain methods, we modify the secret data and the cover medium in the spatial domain, which involves encoding at the level of the LSBs. A simple way of Steganography is based on modifying the least significant bit layer of images, known as the LSB technique. The LSB technique directly embeds the secret data within the pixels of the cover image. In some cases LSB of pixels visited in random or in certain areas of image and sometimes increment or decrement the pixel value. You can hide data in the least and second least significant bits and still the human eye would not be able to discern it [1].

2.3. Steganography in Frequency Domain
The scheme of the frequency domain is to embed the secret data within the cover image that has been transformed such as DCT (discrete cosine transformation). The DCT transforms a cover image from an image representation into a frequency representation, by grouping the pixels into non overlapping blocks of 8 × 8 pixels and transforming the pixel blocks into 64 DCT coefficients each. A modification of a single DCT coefficient will affect all 64 image pixels in that block. The DCT coefficients of the transformed cover image will be quantized, and then modified according to the secret data [1].
Capacity, security and robustness, are the three main aspects affecting Steganography and its usefulness. Capacity refers to the amount of data bits that can be hidden in the cover medium. Security relates to the ability of an eavesdropper to figure the hidden information easily. Robustness is concerned about the resist possibility of modifying or destroying the unseen data.
Hiding the secret image/message in the spatial domain can easily be extracted by unauthorized user and in the frequency domain the quality of the extracted secret image deteriorates. Here, a spatial domain Steganographic technique based on huffman encoding for

hiding a large amount of data with high security, good invisibility and no loss of secret message. The objective here was to develop a procedure which will provide a better security to the secret image without compromising on the quality of the stego image. This algorithm has three main parts. First, it embeds the huffman encoded bit stream of the secret image into the cover image. Second, it embeds the size of the encoded bit stream into the cover image. Third, it also embeds the huffman table corresponding to the secret image into the cover image.

But in normal use, the size of input symbols is limited by the size of the tradition table needed for compression. That is, a table is needed that lists each input symbol and its corresponding code. If a symbol is a one eight-bit byte, then a table of 256 entries is sufficient. Such a table is economical in storage costs but limits the degree of compression achieved.

Second problem with huffman encoding is the complexity of the decompression process. The length of each code to be interpreted for decompression is not known until the first few bits are interpreted. The basic method for interpreting each code is to interpret each bit in sequence and choose a translation sub table according to whether the bit is a one or zero. Decompression with variable-length symbols gives a cost/performance disadvantage whenever the data volume is high. Third problem with huffman encoding is that we need to know the frequency distribution for the ensemble of possible input symbols.

3. Modified LZW Compression

The LZW algorithm is organized around a translation table, referred to here as a string table that maps strings of input characters into fixed-length codes. The use of 12-bit codes is common. The LZW string table has a prefix property in that for every string in the table its prefix string is also in the table. That is, if string ω*k*, composed of some string ω and some single character *k*, is in the table, then ω is in the table. *K* is called the extension character on the prefix string ω. The string table in this explanation is initialized to contain all single-character strings.

The LZW string table contains strings that have been encountered previously in the message being compressed. It consists of a running sample of strings in the message, so the available strings reflect the statistics of the message.

In LZW, the input string is examined character-serially in one pass, and the longest recognized input string is parsed off each time. A recognized string is one that exists in the string table. The strings added to the string table are determined by this parsing: each parsed input string extended by its next input character forms a new string added to the string table. Each such added string is assigned a unique identifier, namely its code value. Here, the code values are represented by English alphabets, such as, a, b, c… etc. For each added string, the next alphabet is used as its code value.

4. Modified LZW Decompression

The LZW decompression logically uses the same string table as the compression and similarly constructs it as the message is translated. Each received code value is translated by way of the string table into a prefix string and extension character. The extension character is pulled off and the prefix string decomposed into its prefix and extension. This operation is recursive until the prefix string is a single character, which completes decompression of that code. This terminal character called here the final character encountered by the compressor when the string was parsed out.

An update to the string table is made for each code received (except the first one). When a code has been translated, its final character is used as the extension character, combined with the prior string, to add a new string to the string table. This new string is assigned a unique code value, represented by English alphabets, which is the same code that the compression is assigned to that string. In this way, the decompression incrementally reconstructs the same string table that the compression used.

5. Modified Fisher – Yates Shuffle Algorithm

The Fisher–Yates shuffle (named after Ronald Fisher and Frank Yates), also known as the Knuth shuffle (after Donald Knuth), is an algorithm for generating a random permutation of a finite set—in plain terms, for randomly shuffling the set. The Fisher–Yates shuffle is unbiased, so that every permutation is equally likely.

The idea behind this algorithm is – random choice. The algorithm in its each step chooses a number n from interval $< 1, m - k>$, where k is a number of already processed elements (number of already performed iterations) and m is the length of the input list. Then the algorithm swaps the element at index n (indexed starting at 1) with an element at index m-k. The algorithm terminates after n-1 iterations (i.e. after possibly swapping all the input elements). In modified version, before the algorithm begins swapping the element at index n and index m – k, a mapping dictionary is created which holds the initial location of colours originally present in secret image. This mapping is shared only between sender and receiver(s) which, later, help in unscrambling the secret image.

6. Proposed System

Hiding the secret image/message in the spatial domain can easily be extracted by unauthorized user and in the frequency domain the quality of the extracted secret image deteriorates. Here, a spatial domain Steganographic technique based on LZW compression for hiding a large amount of data with high security, good invisibility and no loss of secret message. The schematic block diagram of the whole process is given in figure3 and figure 4.

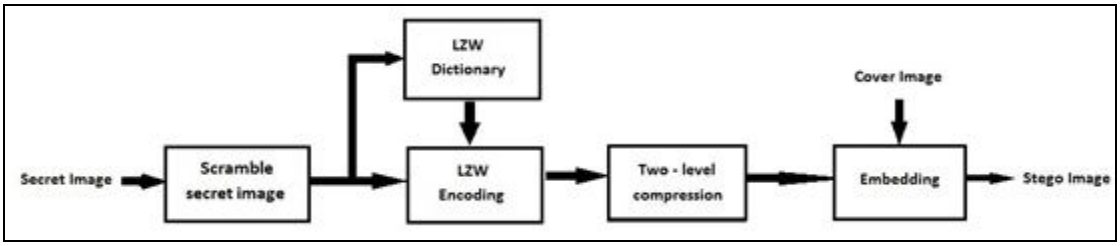

Figure3: Insertion of Secret Image into Cover Image

The above figure 3 shows the process of insertion of secret image into the cover image using LZW algorithm. The secret image is scrambled using Fisher – Yates shuffle algorithm [9][10]. Here, the colour value of each pixel of secret image is stored in a 1 dimension array and Fisher – Yates shuffle algorithm is performed. A scrambled secret image is then recreated. The scrambled secret image is read pixel by pixel and for each of these extracted pixels, LZW dictionary is created. After creation of LZW dictionary, these extracted pixels undergo LZW encoding process with the help of LZW dictionary. The process then undergoes compression, using LZW compression algorithm, wherein the encoded data are compressed by two-levels, thereby returns an encoded, compressed secret data. Then, embedding process takes place wherein this encoded, compressed data is embedded inside the cover image using least significant bit (LSB) process to obtain stego image.

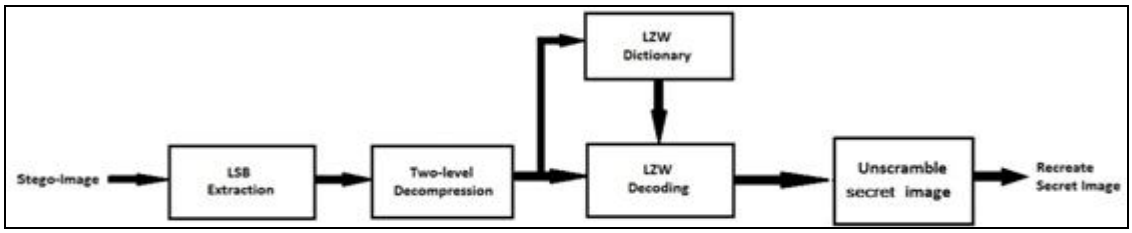

Figure 4: Extraction of Secret Image from Stego Image

The above figure 4 describes the process of extracting secret image from stego image. The stego image is read pixel by pixel for initiating the extraction process. LSB extraction method is used for extracting the pixels of stego image and preparing it for decompression. These extracted bits obtained after LSB extraction then undergoes two-level decompression. The LZW dictionary is created which is then used to undergo two-level decompression process, using LZW decompression algorithm. The process then undergoes LZW decoding process, with the help of LZW dictionary, to recreate the secret image (secret message) in scrambled form. This scrambled secret image is then unscrambled using the mapping dictionary shared between sender and receiver(s).

6.1. Procedure of Two-Level Compression Using LZW Compression

Every pixel value extracted from lossless images for compression, such as bitmap, has three colour channels – red (r), green (g), blue (b). Each of these colour channels are represented by 8-bits. Therefore, every colour value obtained from a pixel has a total of 24-bit binary data. This 24-bit binary data undergoes modified LZW compression to obtain a sequence of English alphabets as codes values. At this stage, first level of compression is completed, in which, a 24-bit binary data is reduced to approximately 12 – 13 characters of English alphabets. The length of this sequence of English characters is taken and is divided by 2. This results in two substrings – each of approximate length 5 - 6 characters. For each substring, its length is calculated and converted to binary form thereby reducing it to 3 characters (3 – bit data in binary form). Same procedure is applied to second substring as well. After completing this procedure, we obtain two 3 – bit binary data. These two 3-bit binary data are concatenated into a single 6 – bit binary data. Then, this 6-bit binary data is divided into three parts, by length, to obtain three 2-bit binary data. At this stage, second level compression is achieved. Each 2-bit binary data represents coded values for r, g and b channel. These 2-bit binary data replaces last two bits of r, g, and b channel of each pixel of cover image using LSB technique.

Let us say the 24-bit binary data is **11111011 11111101 11111010** taken as input for LZW compression. These results in sequence of English alphabets coded values, **b c c a d g b f h f a**. This is your first level compression. The string is then divided into two substrings - **b c c a d** and **g b f h f a**. The length of first substring is 5 and length of second substring is 6. Converting it to binary values results in **101** and **110**. These two binary strings are concatenated to form **101110**. This is then divided into three parts – **10**, **11** and **10**which represents the compressed binary data for R, G and B channels of one pixel of secret image respectively. These three binary data replaces the last two bits of R, G and B channel of one pixel of cover image, using LSB technique.

6.2. Proposed Algorithm for Insertion of Secret Image and Extraction of Secret Image

6.2.1. Insertion Algorithm

Input: A cover image of size M ×N and a secret image of size P × Q.
Output: A stego-image of size M × N.

- Step 1: Read the cover image and secret image. Perform Fisher – Yates Shuffle algorithm on secret image to obtain scrambled secret image.
- Step 2: Read each pixel of scrambled secret image, extract 8-bit binary data from R, G and B channels. Concatenate these three 8-bit binary data together to form a 24-bit binary data.
- Step 3: Send this 24-bit binary data as input for LZW compression to obtain encoded, compressed data string.
- Step 4: Divide this data string into two substrings, by length. Calculate the length of each substring and convert it to binary form of data string.
- Step 5: Concatenate together to obtain 6-bit data string in binary form. Divide this data string into three parts, by length.
- Step 6: Replace these three 2-bit binary data with last two bits of cover image using LSB technique. This results in creation of stego image.
- Step 7: Write the new stego image into disk.
- Step 8: End.

6.2.2. Extraction Algorithm

Input: A M × N stego-image.

Output: A P × Q secret image.

- Step 1: Extract the last 2-bits of each pixel of stego image. Concatenate these 2-bit data together to form 6-bit data.
- Step 2: Divide the length of 6-bit data into two parts to form two substrings.
- Step 3: Obtain first substring LZW compression code. Use this to perform LZW decompression to obtain its binary data. Repeat this step for second substring.
- Step 4: Concatenate these two binary data to obtain 24-bit binary data.
- Step 5: Divide this 24-bit binary data into three parts, each representing 8-bit binary data for R, G and B channel.
- Step 6: Recreate the image from these data for all pixels. The image obtained at the end is a scrambled secret image.
- Step 7: To unscramble the secret image, use the mapping dictionary obtained from Fisher – Yates Shuffling algorithm to get back original secret image.
- Step 8: End.

7. Results

The measurement of the quality between cover image C and stego image S of size M × N is done by using the PSNR (peak signal to noise ratio) value. PSNR is the ratio between maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation [14]. Here, signal in this case is original image and the noise is the error introduced by compression. PSNR ratio is given as follows:

$PSNR = 10 \times \text{Log}_{10} (MAX_I^2 / MSE)$

Where, max_i^2 is the maximum possible pixel value of the image. When pixels are represented using 8-bits per sample, this is 255.

MSE (mean squared error) for a noise-free m × n monochrome image C and its noisy approximation **s** is defined as:

$$MSE = \frac{1}{M.N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [C(i,j) - S(i,j)]^2$$

Where, c (i, j) and s (i, j) are intensity values of pixel at position (i, j) of cove image and stego image respectively. Larger the PSNR indicates higher the image quality i.e. there's little difference between cover image and stego image. PSNR is measured in db. For colour images with three RGB values per pixel, image is converted into a different colour space (YCbCr, HSL) [14]. Because the human eye is most sensitive to luma information, PSNR for colour image is computed by converting the image to a colour space that separates the intensity (luma) channel, such as YCbCr. The y (luma) in YCbCr represents a weighted average of r, g and b. G is given most weight, because the human eye perceives it most easily. Therefore, we compute PSNR only on luma channel [15].

The Matrix Equation For Conversion Of RGB To YCbCr Is Given As Follows [16]:

$$[\text{YCbCr}] = [\text{RGB}] \begin{bmatrix} 0.299 & -0.168935 & 0.499813 \\ 0.587 & -0.331665 & -0.418531 \\ 0.114 & 0.50059 & -0.081282 \end{bmatrix}$$

In figure 5, original cover images are lenna, baboon, airplane, house, sailboat on lake. Secret image is peppers. All images are 24-bit bitmap format. Cover image size is 1024 × 1024. Secret image size is 256 × 265.

Figure 5: Cover Images Are From (A) – (E). Secret Image Is (F)

Table 1 shows the PSNR comparison between four cover images and corresponding four stego images and secret image.

Cover Image	**PSNR (In Db)**	
	Cover Image And Stego Image	**Original Secret Image And Extracted Secret Image**
Lenna	+60.58	Infinity Value Indicates Both Images Are Identical.
Baboon	+60.44	Infinity Value Indicates Both Images Are Identical.
Airplane	+60.33	Infinity Value Indicates Both Images Are Identical.
House	+60.53	Infinity Value Indicates Both Images Are Identical.
Sailboat On Lake	+60.42	Infinity Value Indicates Both Images Are Identical.

Table 1: PSNR Comparison between Original Images and Corresponding Stego Images

Table 2 Shows Comparison between Huffman Encoding Method and Proposed Method

Cover Image	**PSNR (In Db) Between Cover Image And Stego Image**	
	Steganography Based On Huffman Encoding(Grayscale Images)	**Steganography Based On LZW Compression(Colour Images)**
Lenna	+57.43	+60.58
Baboon	+57.46	+60.44
Airplane	+57.46	.+60.33
Sailboat On Lake	+57.46	+60.42

Table 2: PSNR Comparison between Huffman Encoding Method and Proposed Method

Below are some of snapshots taken after performing the proposed method?

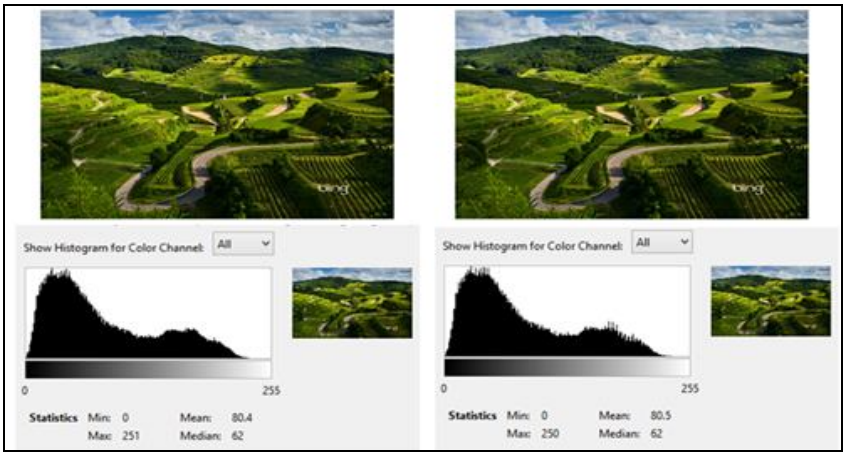

Figure 6: Comparison between Cover Image (L) and Stego Image (R) Using Histograms

Figure 7: Secret Image (R) Retrieved From Stego Image (L)

Figure 8: Original Secret Image (R) Shuffled To Obtain Scrambled Image (L) Using Fisher – Yates Shuffle Algorithm

8. Conclusion

This paper shows a novel technique in performing image Steganography on colour images using LZW compression algorithm and Fisher – Yates shuffle algorithm. LZW compression algorithm, which is usually used to compress text data, has been used successfully in a novel approach in encoding and compressing the pixels of secret images and embedding them inside cover images. Also, Fisher – Yates shuffle algorithm has been used successfully in a novel way of scrambling (shuffling) the colours of pixels of secret images, to obtain scrambled secret image before embedding inside cover image. This project has also shown compressing the binary data of R, G, B channels of secret image into 2-bits binary data as a result of encoding and compression, thereby, allowing to embed two or more secret images inside the cover image. These techniques on image Steganography have been implemented successfully on lossless colour images such as bitmap, png without transparency, gif without transparency.

9. References

1. Rig das, themrichontuithung, “a novel Steganography method for image based on huffman encoding”, IEEE conference 2012.
2. Nadeem akthar, pragatijohri, shahbaaz khan, “enhancing the security and quality of LSB based image Steganography”, 2013 5th international conference on computational intelligence and communication networks
3. Image Steganography based on block-oct and huffman encoding". International journal of computer science and information technology, volume 2, number 3, June 2010.
4. Abbas cheddad, joan condell, kevin curran, paul mckevitt. "digital image Steganography: survey and analysis of current methods".elsevier journal on signal processing 90 (2010) 727-752
5. Alkhraishabes, "4 least significant bits information hiding implementation and analysis", ICGST int. Conf. On graphics, vision and image processing (gvip-05), Cairo, Egypt, 2005.
6. A. Nag, s. Biswas, d. Sarkar, p. P. Sarkar, "a novel technique for image Steganography based on block-oct and huffman encoding". International journal of computer science and information technology, volume 2, number 3, June 2010
7. Neil f. Jhonson, sushiljajodia. "Exploring Steganography: seeing the unseen".IEEE paper of February 1998.
8. Abbas cheddad, joan condell, kevin curran, paul mckevitt. “Digital image Steganography: survey and analysis of current methods”, school of computing and intelligent systems, faculty of computing and engineering, university of ulster at magee, londonderry, bt48 7jl, northern Ireland, UK.
9. http://En.Wikipedia.Org/Wiki/Fisher-Yates_Shuffle
10. http://En.Algoritmy.Net/Article/43676/Fisher-Yates-Shuffle
11. http://Bost.Ocks.Org/Mike/Shuffle/
12. http://Msdn.Microsoft.Com/En-Us/Library/67ef8sbd%28v=Vs.71%29.Aspx
13. http://Msdn.Microsoft.Com/En-Us/Library/System%28v=Vs.110%29.Aspx
14. http://En.Wikipedia.Org/Wiki/Peak_Signal-To-Noise_Ratio
15. http://Www.Mathworks.In/Help/Vision/Ref/Psnr.Html
16. http://Msdn.Microsoft.Com/En-Us/Library/Ff635643.Aspx