IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Implementation of Adaptive Steganalysis Of LSB Replacement in Colour Images

A.S.Renin [*1], Dr.P.Ramamoorthy[2]

[*1,2]Department of ECE, SNS College of Technology, Coimbatore, India

as.renin1802@yahoo.in

Abstract

This paper deals with the detection of hidden bits in the Least Significant Bit (LSB) plane of a natural colour image. The mean Level and the covariance matrix of the image, considered as a quantized Gaussian random matrix, are unknown. An adaptive statistical test is preferred such that its probability distribution is always independent of the unknown image parameters, while ensuring a high probability of hidden bits detection. The pixel of the original image and the cover image will be changed and detected by the detection algorithm. This test is based on the likelihood ratio test except that the unknown parameters are replaced by estimates based on a local linear regression model. It is shown that this test maximizes the probability of detection as the image size becomes arbitrarily large and the quantization step vanishes. This provides an asymptotic upper-bound for the detection of hidden bits based on the LSB replacement mechanism. Numerical results on real natural images show the relevance of the method and the sharpness of the asymptotic expression for the probability of detection.

Keywords: Adaptive detection, information hiding, natural image, nuisance parameters, statistical hypotheses testing.

Introduction

In simple words, Steganography can be defined as the art and science of invisible communication. This is accomplished through hiding information in other information, thus hiding the existence of the communicated information.

Though the concept of steganography and cryptography are the same, but still steganography differs from cryptography. Cryptography focuses on keeping the contents of a message secret, steganography focuses on keeping the existence of a message secret. Steganography and cryptography are both ways to protect information from unwanted parties but neither technology alone is perfect and can be compromised. Once the presence of hidden information is revealed or even suspected, the purpose of steganography is partly defeated. The strength of steganography can thus be amplified by combining it with cryptography.

Almost all digital file formats can be used for steganography, but the formats that are more suitable are those with a high degree of redundancy. Redundancy can be defined as the bits of an object that provide accuracy far greater than necessary for the object's use and display. The redundant bits of an object are those bits that can be altered without the alteration being detected easily. Image and audio files especially comply with this requirement, while research has also uncovered other file formats that can be used for information hiding.

Given the proliferation of digital images, especially on the Internet, and given the large amount of redundant bits present in the digital representation of an image, images are the most popular cover objects for steganography. Ill the domain of digital images many different image file format exit, most of them for specific applications. For these different image file formats, different steganographic algorithms exist.

The most studied algorithm is undoubtedly the simple yet popular technique of hiding in the Least Significant Bit (LSB) of the cover image either in the pixel or transform domain, or its variants. There can be no doubt that replacement of LSBs in digital images is a poor choice for steganography but it remains popular in free steganography software. Moreover, this is the mechanism which inspires the majority of existing hiding methods. Broadly, the literature contains three main classes of detectors for LSB replacement. Numerical experiments on real images and comparisons with existing detection algorithms confirm the statistical performances of the test.

The most studied algorithm is undoubtedly the simple yet popular technique of hiding in the Least Significant Bit (LSB) of the cover image either

in the pixel or transform domain, or its variants. There can be no doubt that replacement of LSBs in digital images is a poor choice for steganography but it remains popular in free steganography software. Moreover, this is the mechanism which inspires the majority of existing hiding methods. Broadly, the literature contains three main classes of detectors for LSB replacement. Numerical experiments on real images and comparisons with existing detection algorithms confirm the statistical performances of the test.

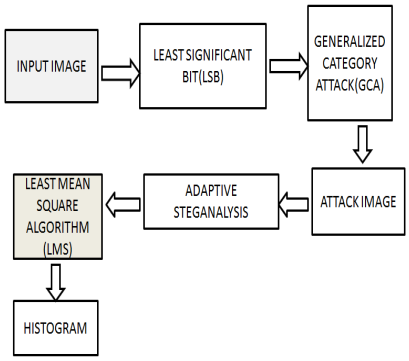


Fig 1:Block Diagram LSB Replacement

In this section we deal with data encoding in still digital images. In essence, image steganography is about exploiting the limited powers of the human visual system (HVS). Within reason, any plain text, cipher text, other images, or anything that can be embedded in a bit stream can be hidden in an image. Image steganography has come quite far in recent years with the development of fast, powerful graphical computers, and steganographic software is now readily available over the Internet for everyday users.

Least Significant Bit Insertion Technique

LSB (Least Significant Bit) substitution is the process of adjusting the least significant bit pixels of the carrier image. It is a simple approach for embedding message into the image. The Least Significant Bit insertion varies according to number of bits in an image. For an 8 bit image, the least significant bit i.e., the 8^{th} bit of each byte of the image is changed to the bit of secret message. For 24 bit image, the colours of each component like RGB (red, green and blue) are changed. LSB is effective in using BMP images since the compression in BMP is lossless. But for hiding the secret message inside an image of BMP file using LSB algorithm it requires a large image which is used as a cover. LSB substitution is also possible for GIF formats, but the problem with the GIF image is whenever the least significant bit is changed the whole colour palette will be changed. The problem can be avoided by only using the gray scale GIF images since the gray scale image contains 256 shades and the changes will be done gradually so that it will be very hard to detect. For JPEG, the direct substitution of steganographic techniques is not possible since it will use lossy compression. So it uses LSB substitution for embedding the data into images.

One of the most common techniques used in steganography today is called least significant bit (LSB) insertion. This method is exactly what it sounds like; the least significant bits of the cover-image are altered so that they form the embedded information. The following example shows how the letter A can be hidden in the first eight bytes of three pixels in a 24-bit image.

Pixels: (00100111 11101001 11001000)
(00100111 11001000 11101001)
(11001000 00100111 11101001)
A: 01000001
Result: (00100110 11101001 11001000)
(00100110 11001000 11101000)
(11001000 00100111 11101001)

The three underlined bits are the only three bits that were actually altered. LSB insertion requires on average that only half the bits in an image be changed. Since the 8-bit letter A only requires eight bytes to hide it in, the ninth byte of the three pixels can be used to begin hiding the next character of the hidden message. A slight variation of this technique allows for embedding the message in two or more of the least significant bits per byte. This increases the hidden information capacity of the cover-object, but the cover-object is degraded more, and therefore it is more detectable. Other variations on this technique include ensuring that statistical chain the image do not occur. Some intelligent software also checks for areas that are made up of one solid color. Changes in these pixels are then avoided because slight changes would cause noticeable variations in the area.

While LSB insertion is easy to implement, it is also easily attacked. Slight modifications in the color palette and simple image manipulations will destroy the entire hidden message. In a “typical” natural scene, the number of even gray values is not the same as the number of odd values. If you embed a 0-1 message string into the least significant bits (LSBs) of an image, then (since it is uniformly distributed), there will be approximately the same number of even and odd values. Statistical “attack” to detect this anomaly, using chi-square statistic.

Transform techniques embed the message by modulating coefficients in a transform domain, such as the Discrete Cosine Transform (DCT) used in JPEG compression, Discrete Fourier Transform, or

Wavelet Transform. These methods hide messages in significant areas of the cover-image, which make them more robust to attack. Transformations can be applied over the entire image, to block throughout the image, or other variants. Data hiding techniques operate in two existing domains, spatial domain and frequency domain.

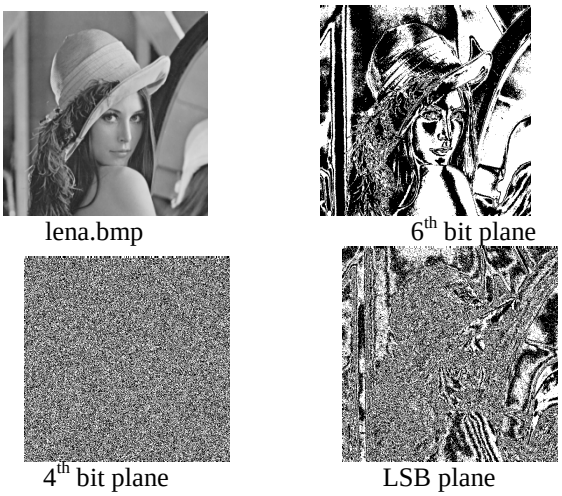


Figure 2. LSB Insertion

The LSB steganography technique is the most common form of technique in spatial domain. This technique makes use of the fact that the least significant bits in an image could be thought of as random noise and changes to them would not have any effect on the image. In Transform domain, embedding is done by altering DCT (Discrete Cosine Transform) coefficients. One of the constraints in transform domain is that many of the coefficients are zero and altering them changes the compression rate of the image. That is why the information carrying capacity of an image is much lesser in transform domain than in spatial domain. Two popular transform domain steganography algorithm are F5 and Out Guess. F5 has two important features. First, it permutes the DCT coefficients before embedding aiming to distribute the induced changes uniformly over the image. Second, it employs matrix embedding to minimize the amount of change made to DCT coefficients. Outguess is also a two step algorithm. It first identifies redundant DCT coefficients which have minimal effect on the cover image, and then depending on the information obtained in initial steps, chooses bits in which it would embed the message. Both F5 and OutGuess have been successfully attacked. Two other tools available on the internet for LSB steganography in transform domain are JSteg and JPHide. While JSteg modifies all the DCT coefficients, JPHide modifies some predecided set of coefficients and hence is more difficult to detect. More sophisticated techniques try to model image statistics and try to minimize changes to them. For example, the method in, the transformed image coefficients are broken down into two parts and replaces the perceptually insignificant component with the coded message signal. There is some work on exploring what is a good cover medium. For example, good cover media are selected given some knowledge about the steganalysis tool.

Steganalysis is the science of detecting steganography. Most methods only aim to detect whether a medium contains hidden data and do not seek to recover the hidden message as well as that is a very hard problem in a general setting. Steganalysis methods can be classified into two categories

1.Specific to a particular steganographic algorithm.

2.Universal steganalysis

Obviously, success rate with former kind of methods is much higher. Provos et al [8] present a tool Steg Detect which is targeted at JSteg, JPHide and OutGuess. They show good detection results and high processing speeds and further use this tool to crawl the internet and find steganographic images.

Most steganalysis techniques in the second category look at how the embedding modifies the statistics of the medium but many of them do not take into account that medium are images which have certain characteristic statistics. However, does try to exploit natural image statistics to some extent. None of the methods make any assumption about the kind of data hidden inside the medium. Universal steganalysis techniques essentially design a classifier based on a training set of cover-objects and stego-objects obtained from a variety of embedding algorithms. Note that none of these techniques allow for recovering the hidden images automatically. In this paper, we are aiming to develop tools which can automatically recover hidden images in LSB steganography scheme.

Kharrazi et al [6] do a comprehensive study of available steganalyzers and study their performance with varying image parameters like size, JPEG compression factors, compression artifacts etc. We are more interested in correlating some image statistics of cover or hidden images with the ability to detect steganography.

Image Encoding Techniques

Information can be hidden many different ways in images. Straight message insertion can be done, which will simply encode every bit of information in the image. More complex encoding can be done to embed the message only in ``noisy'' areas of the image, that will attract less attention. The

message may also be scattered randomly throughout the cover image.

The most common approaches to information hiding in images are:

- Least significant bit (LSB) insertion
- Masking and filtering techniques
- Algorithms and transformations
- Each of these can be applied to various images, with varying degrees of success. Each of them suffers to varying degrees from operations performed on images, such as cropping, or resolution decrementing, or decreases in the color depth.

DCT Encoding

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 . In images there are two types of compression. lossy and lossless compression . Compression plays a very important role in choosing which steganographic algorithm to use. Lossy compression techniques result in smaller image file sizes, but it increases the possibility that the embedded message may be partly lost due to the fact that excess image data will be removed. Lossless compression.though, keeps the original digital image intact without the chance of lost, although is does not compress the image to such a small file size. To compress an image into JPEG format, the RGB colour representation is first converted to a YUV representation. In this representation the Y component corresponds to the luminance (or brightness) and the U and V components stand for chrominance (or color). According to research the human eye is more sensitive to changes in the brightness (luminance) of a pixel than to changes in its color. This fact is exploited by the JPEG compression by down sampling the color data to reduce the size of the file. The color components (U and V) are halved in horizontal and vertical directions, thus decreasing the file size by a factor of 2.

The next step is the actual transformation of the image. For JPEG ,the Discrete Cosine Transform (DCT) is used, but similar transforms are for example the Discrete Fourier Transform (DIT). These mathematical transforms convert the pixels in such a way as to give the effect of "spreading" the location of the pixel values over part of the image. The DCT transforms a signal from an image representation into a frequency representation, by grouping the pixels into 8 x 8 pixel blocks 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 next step is the quantization phase of the compression. Here another biological property of the human eye is exploited: The human eye is fairly good at spotting small differences in brightness over a relatively large area, but not so good as to distinguish between different strengths in high frequency brightness. This means that the strength of higher frequencies can be diminished, without changing the appearance of the image. JPEG does this by dividing all the values in a block by a quantization coefficient. The results are rounded to integer values and the coefficients are encoded using Huffman coding to further reduce the size.

Originally it was thought that steganography would not be possible to use with JPEG images, since they use lossy compression which results in parts of the image data being altered. One of the major characteristics of steganography is the fact that information is hidden in the redundant bits of an object and since redundant bits are left out when using JPEG it was feared that the hidden message would be destroyed. Even if one could somehow keep the message intact it would be difficult to embed the message without the changes being noticeable because of the harsh compression applied. However, properties of the compression algorithm have been exploited in order to develop a steganographic algorithm for JPEGs.

One of these properties of JPEG is exploited to make the changes to the image invisible to the human eye. During the DCT transformation phase of the compression algorithm, rounding errors occur in the coefficient data that are not noticeable. Although this property is what classifies the algorithm as being lossy, this property can also be used to hide messages.

It is neither feasible nor possible to embed information in an image that uses lossy compression, since the compression would destroy all information in the process. Thus it is important to recognize that the JPEG compression algorithm is actually divided into lossy and lossless stages . The DCT and the quantization phase form part of the lossy stage, while the Huffman encoding used to further compress the data is lossless. Steganography can take place between these two stages. Using the same principles of LSB insertion the message can be embedded into the least significant bits of the coefficients before applying the Huffman encoding. By embedding the information at this stage, in the transform domain, it

is extremely difficult to detect, since it is not in the visual domain.

Due to the nature of the compression algorithm, JPEG is excellent at compressing full-color (24-bit) photographs, or compressing grayscale photos that include many different shades of gray. The JPEG algorithm does not work well with web graphics, line art, scanned text, or other images with sharp transitions at the edges of objects. The reason this is so will become clear in the following sections. JPEG also features an adjustable compression ratio that lets a user determine the quality and size of the final image. Images may be highly compressed with lesser quality, or they may forego high compression, and instead be almost indistinguishable from the original.

JPEG compression and decompression consist of 4 distinct and independent phases. First, the image is divided into 8 x 8 pixel blocks. Next, a discrete cosine transform is applied to each block to convert the information from the spatial domain to the frequency domain. After that, the frequency information is quantized to remove unnecessary information. Finally, standard compression techniques compress the final bit stream. This report will analyze the compression of a grayscale image, and will then extend the analysis to decompression and to color images.

Phase One: Divide the Image

Attempting to compress an entire image would not yield optimal results. Therefore, JPEG divides the image into matrices of 8 x 8 pixel blocks. This allows the algorithm to take advantage of the fact that similar colors tend to appear together in small parts of an image. Blocks begin at the upper left part of the image, and are created going towards the lower right. If the image dimensions are not multiples of 8, extra pixels are added to the bottom and right part of the image to pad it to the next multiple of 8 so that we create only full blocks. The dummy values are easily removed during decompression. From this point on, each block of 64 pixels is processed separately from the others, except during a small part of the final compression step.

Phase one may optionally include a change in colorspace. Normally, 8 bits are used to represent one pixel. Each byte in a grayscale image may have the value of 0 (fully black) through 255 (fully white). Color images have 3 bytes per pixel, one for each component of red, green, and blue (RGB color). However, some operations are less complex if you convert these RGB values to a different color representation. Normally, JPEG will convert RGB colorspace to YCbCr colorspace. In YCbCr, Y is the luminance, which represents the intensity of the color. Cb and Cr are chrominance values, and they actually describe the color itself. YCbCr tends to compress more tightly than RGB, and any colorspace conversion can be done in linear time. The colorspace conversion may be done before we break the image into blocks; it is up to the implementation of the algorithm.

Finally, the algorithm subtracts 128 from each byte in the 64-byte block. This changes the scale of the byte values from 0…255 to –128…127. Thus, the average value over a large set of pixels will tend towards zero.

The following images show an example image, and that image divided into an 8 x 8 matrix of pixel blocks. The images are shown at double their original sizes, since blocks are only 8 pixels wide, which is extremely difficult to see. The image is 200 pixels by 220 pixels, which means that the image will be separated into 700 blocks, with some padding added to the bottom of the image. Also, remember that the division of an image is only a logical division, but in figure 1 lines are used to add clarity.

Before: 

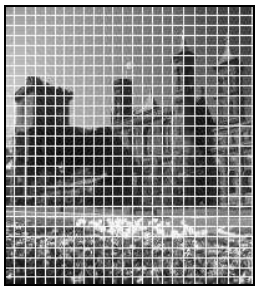
After: 

Figure 3: Example of Image Division

Phase Two: Conversion to the Frequency Domain

At this point, it is possible to skip directly to the quantization step. However, we can greatly assist that stage by converting the pixel information from the spatial domain to the frequency domain. The conversion will make it easier for the quantization process to know which parts of the image are least important, and it will de-emphasize those areas in order to save space.

Currently, each value in the block represents the intensity of one pixel (remember, our example is a grayscale image). After converting the block to the frequency domain, each value will be the amplitude of a unique cosine function. The cosine functions each have different frequencies. We can represent

the block by multiplying the functions with their corresponding amplitudes, then adding the results together. However, we keep the functions separate during JPEG compression so that we may remove the information that makes the smallest contribution to the image.

There are many algorithms that convert spatial information to the frequency domain. The most obvious of which is the Fast Fourier Transform (FFT). However, due to the fact that image information does not contain any imaginary components, there is an algorithm that is even faster than an FFT. The Discrete Cosine Transform (DCT) is derived from the FFT, however it requires fewer multiplications than the FFT since it works only with real numbers. Also, the DCT produces fewer significant coefficients in its result, which leads to greater compression. Finally, the DCT is made to work on one-dimensional data. Image data is given in blocks of two-dimensions, but we may add another summing term to the DCT to make the equation two-dimensional. In other words, applying the one-dimensional DCT once in the x direction and once in the y direction will effectively give a two-dimensional discrete cosine transform.

The 2D discrete cosine transform equation is given in figure 2, where C(x) = 1/√2 if x is 0, and C(x) = 1 for all other cases. Also, $f(x, y)$ is the 8-bit image value at coordinates (x, y), and $F(u, v)$ is the new entry in the frequency matrix.

$$F(u,v) = \frac{1}{4} \cdot C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\frac{(2x+1)\cdot u\pi}{16}\cos\frac{(2y+1)\cdot v\pi}{16}\right]$$

We begin examining this formula by realizing that only constants come before the brackets. Next, we realize that only 16 different cosine terms will be needed for each different pair of (u, v) values, so we may compute these ahead of time and then multiply the correct pair of cosine terms to the spatial-domain value for that pixel. There will be 64 additions in the two summations, one per pixel. Finally, we multiply the sum by the 3 constants to get the final value in the frequency matrix. This continues for all (u, v) pairs in the frequency matrix. Since u and v may be any value from 0…7, the frequency domain matrix is just as large as the spatial domain matrix.

The frequency domain matrix contains values from -1024…1023. The upper-left entry, also known as the DC value, is the average of the entire block, and is the lowest frequency cosine coefficient. As you move right the coefficients represent cosine functions in the vertical direction that increase in frequency. Likewise, as you move down, the coefficients belong to increasing frequency cosine functions in the horizontal direction. The highest frequency values occur at the lower-right part of the matrix. The higher frequency values also have a natural tendency to be significantly smaller than the low frequency coefficients since they contribute much less to the image. Typically the entire lower-right half of the matrix is factored out after quantization. This essentially removes half of the data per block, which is one reason why JPEG is so efficient at compression.

Phase Three: Quantization

Having the data in the frequency domain allows the algorithm to discard the least significant parts of the image. The JPEG algorithm does this by dividing each cosine coefficient in the data matrix by some predetermined constant, and then rounding up or down to the closest integer value. The constant values that are used in the division may be arbitrary, although research has determined some very good typical values. However, since the algorithm may use any values it wishes, and since this is the step that introduces the most loss in the image, it is a good place to allow users to specify their desires for quality versus size.

The algorithm uses the specified final image quality level to determine the constant values that are used to divide the frequencies. A constant of 1 signifies no loss. On the other hand, a constant of 255 is the maximum amount of loss for that coefficient. The constants are calculated according to the user's wishes and the heuristic values that are known to result in the best quality final images. The constants are then entered into another 8 x 8 matrix, called the quantization matrix. Each entry in the quantization matrix corresponds to exactly one entry in the frequency matrix. Correspondence is determined simply by coordinates, the entry at (3, 5) in the quantization matrix corresponds to entry (3, 5) in the frequency matrix.

The equation used to calculate the quantized frequency matrix is fairly simple. The algorithm takes a value from the frequency matrix (F) and divides it by its corresponding value in the quantization matrix (Q). This gives the final value for the location in the quantized frequency matrix ($F_{quantize}$). Figure 3 shows the quantization equation that is used for each block in the image.

$$F_{Quantize}(u,v) = \left(\frac{F(u,v)}{Q(u,v)}\right) + 0.5$$

Noise

Impulse noise is caused by malfunctioning pixels in camera sensors, faulty memory locations in

hardware, or transmission in a noisy channel. See [5] for instance. Two common types of impulse noise are the salt-and-pepper noise and the random-valued noise.

Results

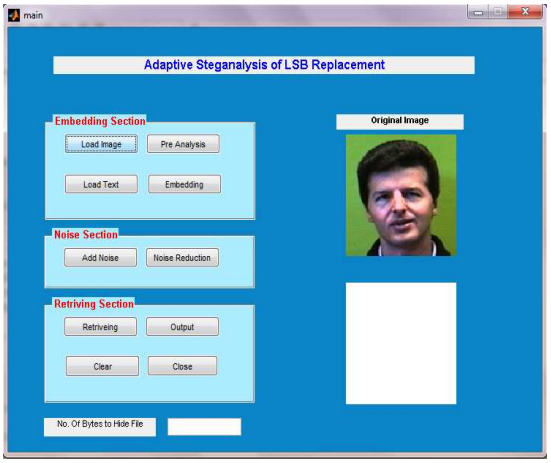


Figure 3:Result for Loading the Image

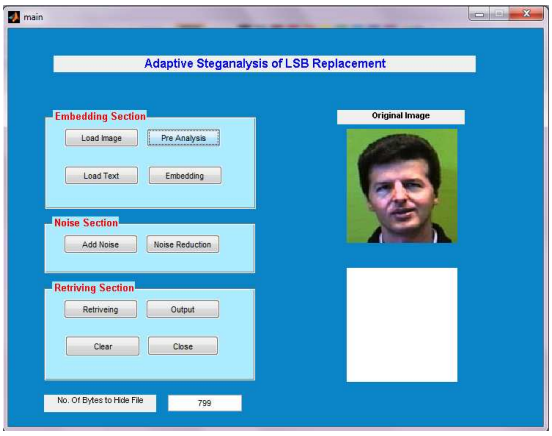


Figure 4: Result for Preanalysis

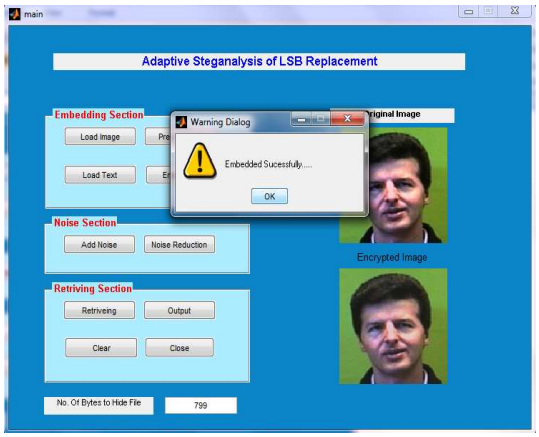


Figure 5: Result for Encoding

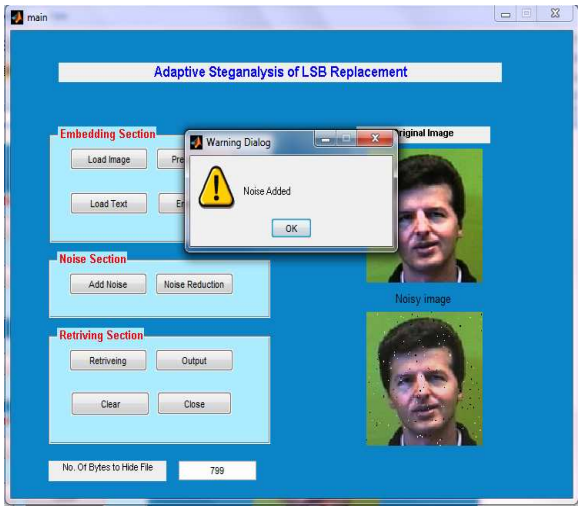


Figure 6: Noise Intrusion

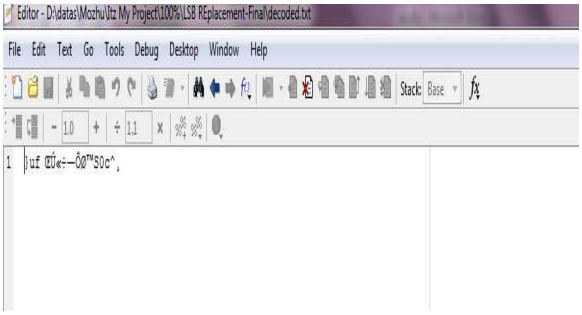


Figure 7:Output for Decoing when noise is intruded

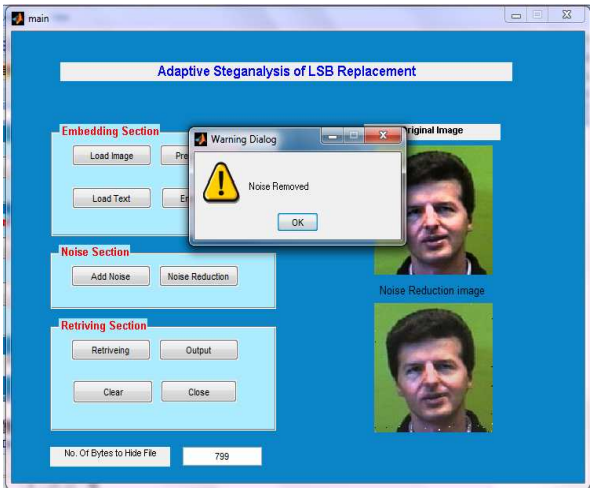


Figure 8: Removal of noise

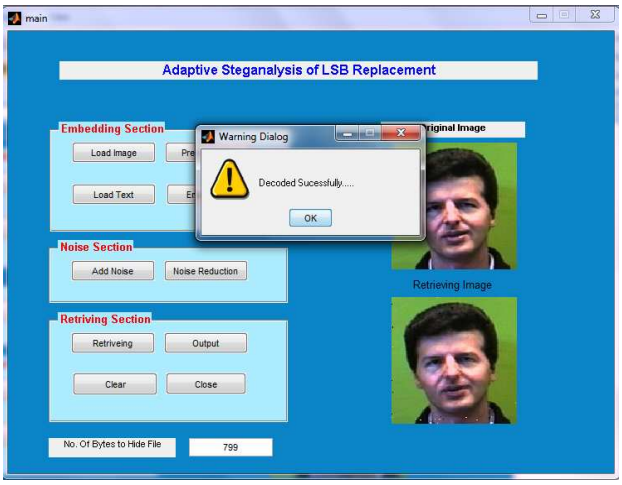


Figure 9: Decoding Process

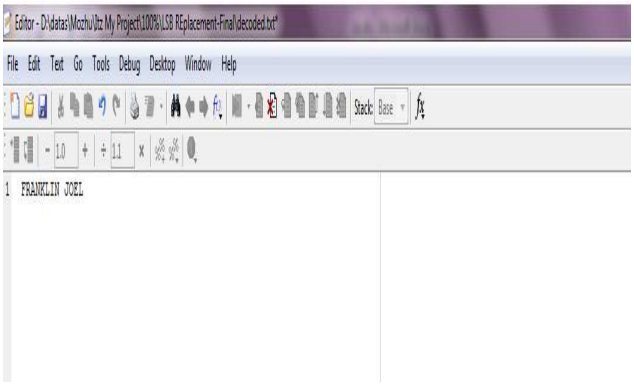


Figure 3: Output of Decoding(Hidden information is a text)
FRANKLIN JOEL

Conclusion

Steganography can be used for hidden communication. The limits of steganography are explained. The image steganographic system using LSB approach to provide a means of secure communication is discussed. In this approach, the message bits are embedded into the least significant bits of cover image pixels. As presented, LSB Embedding has the advantage that it is simple to implement. This is especially true in the 24-bit bitmap case. It also allows for a relatively high payload, carrying one bit of the secret image per byte of pixel data. In addition, it is also seemingly undetectable by the average human if done right. However, the assumption has been that the stego-image is indistinguishable from the original cover image by the human eye. There have been many statistical techniques developed to determine if an image has been subjected to LSB Embedding.

Acknowledgement

I Extend my heartfull thanks to my College as well as to my department .

References

[1] H. Sencar, M. Ramkumar, and A. Akansu, Data Hiding Fundamentals and Applications: Content Security in Digital Multimedia. Elsevier: Academic, 2004.

[2] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker, Digital Watermarking and Steganography. San Francisco, CA: Morgan Kaufmann,2007.

[3] J. Fridrich, Steganography in Digital Media—Principles, Algorithms, and Applications. New York: Cambridge Univ. Press, 2009.

[4] R. Böhme, Advanced Statistical Steganalysis. New York: Springer,2010.

[5] N. Provos and P. Honeyman, "Hide and seek: An introduction to steganography," IEEE Secur. Priv. J., vol. 1, no. 3, pp. 32–44, 2003.

[6] X.-Y. Luo, D.-S. Wang, P. Wang, and F.-L. Liu, "A review on blind detection for image steganography," Signal Process., vol. 88, no. 9, pp.2138–2157, Sep. 2008.

[7] A. Nissar and A. Mir, "Classification of steganalysis techniques: A study," Digit. Signal Process., vol. 20, no. 6, pp. 1758–1770, 2010.

[8] R. Cogranne, C. Zitzmann, L. Fillatre, F. Retraint, I. Nikiforov, and P.Cornu, "Statistical decision by using quantized observation," in Proc.Int. Symp. Inf. Theory, St. Petersburg, Russian, 2011, pp. 1135–1139.

[9] R. Gray and D. Neuhoff, "Quantization," IEEE Trans. Inf. Theory, vol.44, pp. 2325–2384, 1998.