Image Steganography tool using Adaptive Encoding Approach to maximize Image hiding capacity

Manoj Kumar Meena, Shiv Kumar, Neetesh Gupta

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 Steganographic techniques some are more complex than others and all of them have respective strong and weak points. The Least Significant Bit (LSB) insertion method is the most common and easiest method for embedding messages in an image. This paper work intends to give an overview of image Steganography, its uses and techniques along with an attempt to implement a basic image Steganographic model by using an adaptive encoding algorithm which attempts to maximize the embedding capacity of each pixel through LSB insertion method. The Implementation is named StegSan which also uses file compression and data encoding mechanisms to make the image more secure and less detectable.**

***Index Terms*— Steganography, LSB Insertion method, adaptive encoding, StegSan approach.**

I. Introduction

Nowadays data hiding is a challenging issue. Steganography is considered one of the branches in data hiding and it is the art of hiding secret data within the cover image [1] [2]. The simplest steganogaphic method is carried out by embedding secret data in the least significant bit(s) (LSB) of the cover image. Two main issues are considered when using the algorithm. The first issue is imperceptibility which is the visibility of the secret data to the human eyes. The second issue is capacity, i.e., how much amount of secret data that can be embedded in the cover image [3]. In this paper the main focus is to improvement made term of imperceptibility in comparison with adaptive algorithm. The proposed method utilizes to calculate intensity parameter value of pixel from nearest target pixel. The StegSan technique is discussed in section V. The proposed algorithm is explained step by step in section V.A. in section VII, the experimental results are presented. Finally in section 8, the desired conclusion is made based on the results achieved.

Manuscript received April, 2011.

Manoj K Meena is M.Tech student of Technocrats Institute of Technology,
Bhopal, India, e-mail: manu1.love@gamil.com
Prof. Shiv Kumar, Asst. Professor, Department of Information Technology, Technocrats Institute of Technology, Bhopal (M.P.), India (e-mail: shivksahu@rediffmail.com).
Prof. Neetesh Gupta, Asst. Professor & Head, Department of Information Technology, Technocrats Institute of Technology, Bhopal (M.P.), India (e-mail: gupta.neetesh81@gmail.com).

II. Image Steganography

Image compression techniques are extensively used in Steganography. Among the two types of image compressions, lossy compression and loss less compression, loss less compression formats offer more promises. Typical examples of loss less compression formats are CompuServe's GIF (Graphics Interchange Format) and Microsoft's BMP (Bitmap) [3]. We have used an 8-bit image size for implementation of our Steganography. In using an 8-bit image as the cover-image, many Steganography experts recommend using images featuring 256 shades of gray as the palette[12]. Grey-scale images are preferred because the shades change very gradually between palette entries. This increases the image's ability to hide information. Once a suitable cover image has been selected, an image encoding technique needs to be chosen. Improvement in Steganographic techniques make it possible to apply the Detecting LSB Steganography in Color and Gray- Scale Images which were confined to gray scale images in the initial stages

A. Image Steganography Model

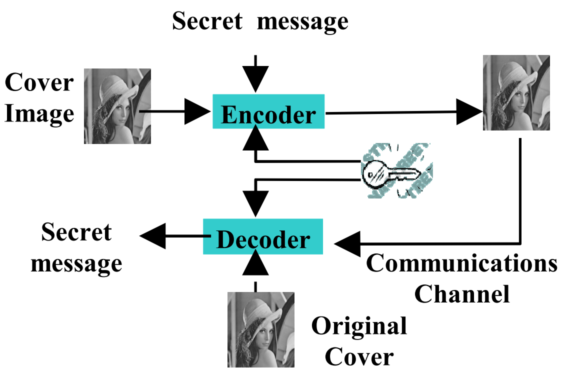

Fig: 1. Model of Steganography

The Basic model of Steganography is as shown in the above figure. The cover image acts as a carrier for the secret data. The secret data is embedded into the cover image by means of a Steganographic algorithm. The result obtained is the Stego-image. This Stego-image is transferred from the sender's end to the receiver's end over the communication channel. At the receiver's end, the same Steganographic algorithm works to extract the original secret data from the cover image. The sender and receiver must agree upon a common key to embed and extract the secret data from the cover image.

III. STEGANOGRAPHY TECHNIQES IN IMAGE DOMAIN

The Most common Steganographic Technique in Image domain is Least Significant Bit Insertion method. This method is the easiest one and in fact most challenging. It has been explained below.

A. *Least Significant Bit (LSB) Insertion*

Least significant bit (LSB) insertion is a common, simple approach to embedding information in a cover image. Unfortunately, it is vulnerable to even a slight image manipulation. Converting an image from a format like GIF or BMP, which reconstructs the original message exactly (*lossless compression*) to a JPEG, which does not (*lossy compression*), and then back could destroy the information hidden in the LSBs[10].

24-bit Images

When using a 24-bit image, a bit of each of the red, green and blue color components can be used, since they are each represented by a byte. In other words, one can store 3 bits in each pixel. An 800 × 600 pixel image, can thus store a total amount of 1,440,000 bits or 180,000 bytes of embedded data. For example a grid for 3 pixels of a 24-bit image can be as follows:

```
(00101101     00011100     11011100)
(10100110     11000100     00001100)
(11010010     10101101     01100011)
```

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

(0010110**1**   0001110**1**   1101110**0**)
(1010011**0**   1100010**1**   0000110**0**)
(1101001**0**   1010110**0**   01100011)

Although the number was embedded into the first 8 bytes of the grid, only the 3 underlined bits needed to be changed according to the embedded message. On average, only half of the bits in an image will need to be modified to hide a secret message using the maximum cover size. Since there are 256 possible intensities of each primary color, changing the LSB of a pixel results in small changes in the intensity of the colors. These changes cannot be perceived by the human eye - thus the message is successfully hidden. With a well-chosen image, one can even hide the message in the least as well as second to least significant bit and still not see the difference. In the above example, consecutive bytes of the image data – from the first byte to the end of the message – are used to embed the information. This approach is very easy to detect. A slightly more secure system is for the sender and receiver to share a secret key that specifies only certain pixels to be changed. Should an adversary suspect that LSB Steganography has been used, he has no way of knowing which pixels to target without the secret key. In its simplest form, LSB makes use of BMP images, since they use lossless compression. Unfortunately to be able to hide a secret message inside a BMP file, one would require a very large cover image. Nowadays, BMP images of 800 × 600 pixels are not often used on the Internet and might arouse suspicion. For this reason, LSB Steganography has also been developed for use with other image file formats.

8-bit Images

8-bit images are not as forgiving to LSB manipulation because of color limitations. Steganography software authors have devised several approaches— some more successful than others—to hide information in 8-bit images. First, the cover image must be more carefully selected so that the Stego-image will not broadcast the existence of an embedded message. When information is inserted into the LSBs of the raster data, the pointers to the color entries in the palette are changed. In an abbreviated example, a simple four-color palette of white, red, blue, and green has corresponding palette position entries of 0 (**00**), 1 (**01**), 2 (**10**), and 3 (**11**), respectively. The raster values of four adjacent pixels of white, white, blue, and blue are **00 00 10 10**. Hiding the binary value **1010** for the number 10 changes the raster data to **01 00 11 10**, which is red, white, green, blue. These gross changes in the image are visible and clearly highlight the weakness of using 8-bit images. On the other hand, there is little visible difference noticed between adjacent gray values.

IV. PROBLEM ANALYSIS

Although a lot of the work has been done in the field of Image Steganography, it has emerged as a challenging field in data security due to the availability of scope of improvement in its various techniques and algorithms. Still Steganography has received little attention in computing. Primarily this is the factor which has motivated us a lot. The other motivating factors are as follows.

A. The rapid revolution in digital multimedia and the ease of generating identical and unauthorized digital data.

B. Detecting counterfeiter, unauthorized presentation, embed key, embed author ID.

C. 9/11 Al Quaida Attacks.

D. Need to establish reliable methods for copyright protection and authentication

V. PROPOSED TECHNIQUE: STEGSAN

A. Adaptive Encoding

The main feature of **StegSan** is its adaptive encoding algorithm which optimizes the use of hiding space in a particular **cover image**. In bitmaps, Steganography is usually performed by replacing the **least-significant bit** (LSB) of the color values of each pixel with the data bits of the file or message to be hidden. Since LSBs represent only a small portion of the actual color values, such changes in an image are often invisible to the human eye.

Consider, for example, a 640x480 true-color bitmap. True color images are composed of red, green, and blue color channels. In each pixel, the intensity or color value for each channel is represented by an 8-bit number[4]. Using the LSBs of the three color values of each pixel would generate a hiding space of 1,15,200 bytes (or 921,600 bits), 1/8 of the total image size.

To maintain image fidelity, StegSan assigns different hiding capacities for each pixel (instead of a fixed size) based on the texture formed by surrounding pixels. The idea behind this is that areas of an image with more complex textures are less sensitive to modifications as compared to areas with smooth and solid textures. StegSan therefore uses optimum hiding capacities in every cover image, thereby making the Steganographic process less detectable and more secure.

To Achieve Adaptive Encoding three basic methods are being used.

1) Calculate insertion bits among pixel
2) Minimum error replacement.
3) Error diffusion.

1.calculate insertion bits among pixel

To illustrate Calculate insertion bits among surrounding pixel, consider the set of grayscale pixels in Figure2, wherein pixel *P* represents the pixel being analyzed. The first step is to get the *gray value variation* among the pixels on the top and middle-left of the pixel being analyzed. In Figure2, these are pixels *A*, *B*, *C*, and *D*. The gray value variation is defined as the difference between the maximum and the minimum gray values among the four pixels. The formula for the gray level variation *V* can be written as:

$$V = \max\{A, B, C, D\} - \min\{A, B, C, D\}$$

The number of bits that can be modified in pixel *P* is the minimum number of bits needed to store the binary value of *V* minus 1. This can be expressed mathematically as:

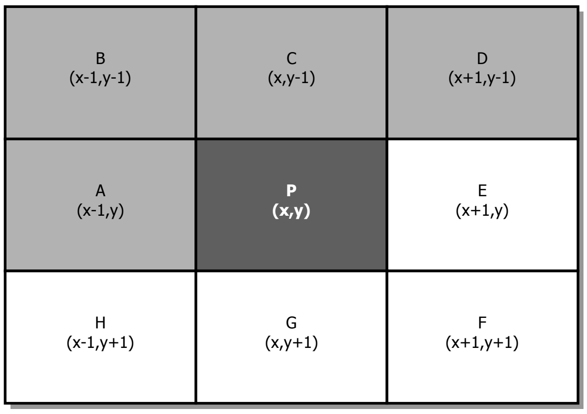

Fig: 2. Set of grayscale pixels

$$K = \lfloor \log_2 V \rfloor$$

It is important to note that with this technique, embedding can only be performed in a top-to-bottom left-to-right orientation. Only the top and middle left adjacent pixels are considered in the evaluation because the embedding function has already passed through these pixels. The other four pixels on the bottom and middle right of pixel *P* are still to undergo the embedding process, which means their values may still change. Calculate insertion bits among surrounding pixel function calculate a pixel's hiding capacity based on the *range* of gray values of surrounding pixels.

However, since StegSan is designed for true-color images, the above concept would have to be implemented differently. First of all, Calculate insertion bits among surrounding pixel will have to be performed on each of the three color channels independently. As a result, one color component of a pixel may increase while another decreases. In such cases, the increase in total *embedding error* may result to visible distortions. To compensate for this, the original formula for *gray value variation* is replaced with a formula for *color intensity variation* based on the "texture" formed by pairs of adjacent pixels on the top and middle-left sides of the pixel being evaluated. This new formula may be expressed in the form:

$$V = \text{round}\,[(|C-A|+|A-B|+|B-C|+|C-D|) / 4]$$

This modified Calculate insertion bits among surrounding pixel function provides a more sensitive and more accurate estimate of a pixel's tolerance to modifications.

2. *Minimum-Error Replacement*

In order to minimize the changes made to a pixel as a result of embedding, *minimum-error replacement* (MER)[9] is also incorporated in StegSan. The idea behind MER is to adjust the bit that is immediately succeeding the modified LSB's of a particular color value in such a way that the change caused by the embedding operation is minimal.

For example, if a binary number 1000 (decimal number 8) is changed to 1111 (decimal number 15) because its three LSB's were replaced with embedded data, the difference from the original number is 7. This difference in the original value of a color component is called the ***embedding error***. By adjusting the fourth bit from a value of 1 to a value of 0, the binary number now becomes 0111 (decimal number 7) and the embedding error is reduced to 1 while at the same time preserving the value of the three embedded bits.

3. *Error Diffusion*

Since the Calculate insertion bits among surrounding pixel technique analyzes only the top and middle left adjacent pixels, distortions in the contour of certain areas in an image might still be made. For an image to adapt to the changes in color caused by embedding, an *error diffusion* technique must also be employed.

An error diffusion technique called *Improved Gray-Scale Compensation* (IGSC), which makes up for the embedding error in the grayscale value of a pixel by spreading that error evenly across the adjacent pixels is proposed. This is done by adding ¼ of the embedding error to the intensity or grayscale value of the bottom and middle right adjacent pixels. These pixels are depicted in Figure 2 as pixels *E*, *F*, *G*, and *H*.

B. File Security

StegSan uses Computed MD5 hash value of the user given password. Since hash value have the tendency of not been unique it is possible (although very unlikely) for an unauthorized user to use an incorrect password that coincidentally generates the same hash value with that of the correct password.

A. *Implementation Model*

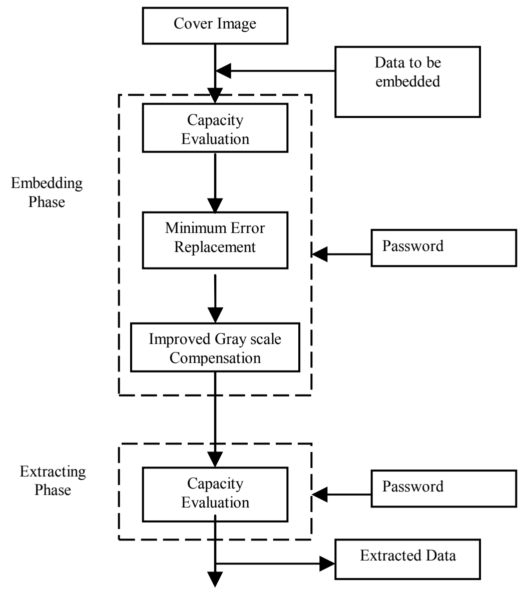

Fig: 3 Block Diagram of StegSan

C. File Wiping

StegSan also features a *file wiping* option which allows users to permanently delete files from the computer. Typically, when a file is deleted, it is simply unregistered from the file system but its actual contents are left intact in the storage media. This makes it possible for an attacker to recover the contents of the deleted file. To ensure data security, the actual contents of the file must be erased. Such an operation is generally called file wiping

VI. RESULT ANALYSIS

Stenographic algorithms have to comply with a few basic requirements. The requirements are: Imperceptibility, Capacity, Robustness against statistical attacks, and Robustness against image manipulation. The following table compare least significant bit (LSB) insertion in BMP and in GIF files, JPEG compression steganography, the adaptive encoding approach and StegSan technique, according to the about requirement.

TABLE 1

COMPARISON OF IMAGE STEGANOGRAPHY ALGORITHM

Comparison/ requirements	Single LSB insertion method	Adaptive Encoding using 4th LSB insertion method
Imperceptibility	High*	High*
Capacity	Medium	High*
Robustness against statistical attacks	Low	High
Robustness against Manipulation	Low	Medium
Data encryption	Low	High

*- Depends on cover image used.

The levels at which the algorithms satisfy the requirements are defined as high, medium and low. A high level means that the algorithm completely satisfies the requirement, while a low level indicates that the algorithm has a weakness in this requirement.

TABLE II

EMBEDDING ALGO.(KB) VS IMAGE SIZE (PIXEL)

These results are taken by StegSan tool. This tool has

Resolution/ Embedding Algo (KBits)	LSB Insertion method	Adaptive Encoding using 4th LSB insertion method
450*450	75.93	106.11
564*423	89.46	111.83
720*600	162.00	245.19
800*600	180.00	283.11
1024*768	294.91	354.22

performed on various resolution images and calculate embedding capacity of bits in cover image.
More specifically, StegSan offers the following features:

I. Adaptive encoding algorithm
II. File security
III. File compression
IV. Data encryption
V. File wiping

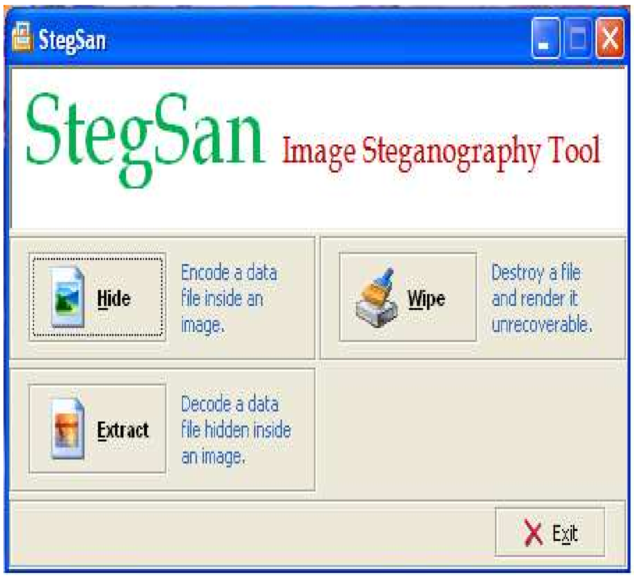
StegSan
StegSan Image Steganography Tool
Hide
Encode a data file inside an image.
Wipe
Destroy a file and render it unrecoverable.
Extract
Decode a data file hidden inside an image.
Exit

Fig:4 StegSan Tool

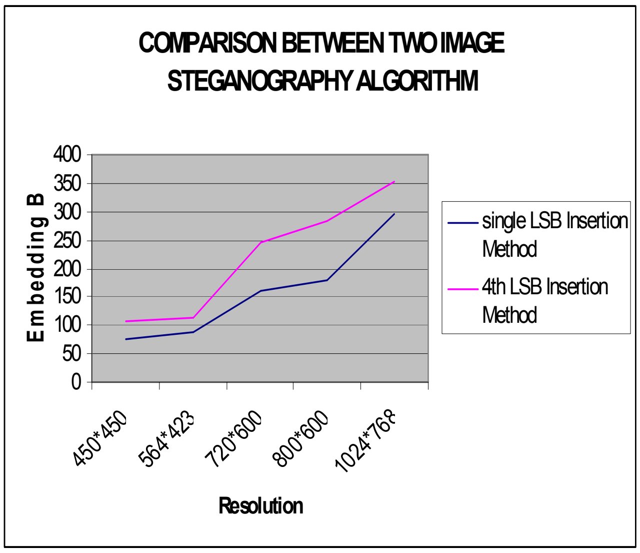

Fig: 5 Comparison Chart

Fig:6 Original Image

Fig:7 Stego Image

VII. CONCLUSION & FUTURE WORK

Through adaptive encoding, StegSan is able to utilize optimum hiding capacities in every cover image. This allows users to hide files of larger sizes while at the same time preserve the general appearance of any cover image used. Furthermore, the implementation of various security measures provides a high level of protection for the hidden data. Every pixel has its own hiding capacity according to adaptive encoding. The pixels of an image which are more tolerant to modifications are identified along with complex textures.

Although limited to lossless image formats, which are in any case standard and considerably widespread, StegSan may be useful in real-world applications especially in cases wherein large volumes of sensitive data need to be transmitted secretly over public communications channels such as the Internet. Although Steganography has emerged as a strong technique to secure our data from unintended recipients or intruders by reducing as much as possible the degree of suspicion, still this field requires a lot of research work and development of new techniques to achieve it. The technique presented in this paper requires a lot of enhancement and development to be carried out. Since this technique has been implemented only on four LSBs while maintaining the image fidelity, the future work requires embedding more data into the LSBs up to 5th or may be up to 6th place while maintaining the image fidelity.

REFERENCES

[1] F. A. P. Petitcolas, R. J. Anderson and M. G. Kuhn, "Information hiding-A survey," Proc. IEEE, vol. 87, pp. 1062-1078,1999.

[2] H. Wang and S. Wang, "Cyber warfare-Steganography vs. Steganalysis," Commun. ACM, vol. 47, no. 10, pp. 76-82, 2004.

[3] X. Zhang and S. Wang, "Steganography using multiplebase notational system and human vision sensitivity," IEEE Signal Processing Letters, vol. 12, pp. 67-70, Jan. 2005.

[4] Masoud Afrakhteh, Adaptive Steganography Scheme Using More Surrounding Pixels, proc IEEE VI-225 Volume I (ICCDA 2010)

[5] H. Noda, J. Spaulding, M. N. Shirazi, and E. Kawaguchi, "Application of bit-plane decomposition steganography to JPEG2000 encoded images," IEEE Signal Processing Lett., vol. 9, no. 12, pp. 410-413, Dec. 2002.

[6] D.-C. Wu and W.-H. Tsai, "A steganographic method for images by pixel-value differencing," Pattern Recognit. Lett., vol. 24, pp. 1613-1626,2003.

[7] C. K. Chan and L. M. Cheng, "Hiding data in images by simple LSB substitution," Pattern Recognition, pp. 469--474, Mar. 2004.

[8] N.!. Wu and M.S. Hwang,"Data Hiding: Current Status and Key Issues," International Journal of Network Security, Vol.4, No.1, PP. 1-9,2007.

[9] Y.K. Lee and L. h. Chen, "An Adaptive Image Steganographic Model Based on Minimum-Error LSB Replacement," In Proceedings of the Ninth National Conference on Information Security. Taichung, Taiwan, 14-15, pp.8-15, 1999.

[10] Chandramouli, R.; Memon, N.; , "Analysis of LSB based image steganography techniques," *Image Processing, 2001.Proceedings. 2001 International Conference on* , vol.3, no., pp.1019-1022 vol.3, 2001 doi: 10.1109/ICIP.2001.958299 URL:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=958299&isnumber=20708

[11] Hengfu YANG ,Xingming SUN , Guang SUN "A High-Capacity Image Data Hiding Scheme Using Adaptive LSB Substitution" VOL. 18, NO. 4, Dec. 2009.

[12] Chandramouli, R., Kharrazi, M. and Memon, N. (2004) "Image Steganography and Steganalysis: Concepts and Practice", In: Kalker, T. and Cox, I. et al. (ed.). Digital Watermarking Springer Berlin / Heidelberg 204-11.