*ESTEEM Academic Journal*
Vol. 14, Special Issue, Jan 2018, 45-52

# Least Significant Bit (LSB) Steganography Technique on Digital PNG Formatted Images

Abdul Hakim Shukor[1], Noratikah Shamsudin[1] and Noraini Seman[1]
[1]*Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 41050 Shah Alam, Selangor, Malaysia*
**corresponding author: atikah@tmsk.uitm.edu.my*

**ARTICLE HISTORY**

Received
*10 November 2017*

Received in revised form
*17 November 2017*

Accepted
*25 January 2018*

**ABSTRACT**

*Steganography is the efficient technique to provide secure data transmission over the network, as the number of users increases effectively. The cryptography is also used to provide security to data over network, but transmission of secured message may be detectable to third party. From security point of view, steganography does not allow to detect the presence of hidden secret other than indeed user, over the communication channel. Here we are implementing the image steganography i.e image as the master file or cover media and secrete message can be text messages. This paper presents to provide the transfer of secret data embedded into master file to obtain new image , which is practically indistinguishable from the original image, so that other than the indeed user, cannot detect the presence of the secrete data sent. Here we use the Least Significant Bit (LSB) algorithm technique for hiding the secrete data by embedding the secrete data into a master file in sending station and we use reverse process of LSB encoding technique during the retrieval of the secrete data from the master file by the intended user. The PSNR of both techniques should be measured as performance characteristics of the steganography and comparing both the techniques.*

**Keywords:** *steganography; least significant bit; cryptography; master file; security.*

## 1. INTRODUCTION

There has been considerable interest in data hiding in the last decade. The advent of the digital age, the proliferation of computers and the widespread usage of the Internet have created digital multimedia as a host for reliable and secure data transfer. Data security is one of the most important issues for computer forensic experts. The confidential information incorporate, government offices needs to be kept secured from unauthorized users. Experts take necessary steps to keep data safe and secured from unintended users and one of the techniques used by the experts is steganography.

The word steganography is derived from the Greek words “stegos” meaning “cover” and “grafia” meaning “writing”, defining it as “covered writing”. Steganography is an art of hiding confidential information in some cover medium file. It can be used to hide secret message in digital multimedia files which can be retrieved later by intended users. This technique can be used for secret communication and confidential information sharing via internet medium. This method of secret message passing is seen to be misused in offices by leaking important company documents though some cover medium files. Encryption techniques along with steganography can provide high level of data security. The goal of this

p-ISSN 1675-7939; e-ISSN 2289-4934
© 2018 Universiti Teknologi MARA Cawangan Pulau Pinang

research is to develop an application which can hide a text message in image files and detect the presence of hidden message in image files using appropriate analysis method and also should be able to retrieve the hidden text successfully from image file containing data.

This research focuses on steganography using digital images. Images are the most popular cover objects used for steganography. In the domain of digital images many different file formats exist and for these file formats different algorithms exist. A lot of methods have been proposed for embedding data inside images such as spatial domain embedding, transform domain embedding and model based techniques. Spatial domain embedding, or more popularly known as the Least Significant Bit (LSB) technique is the most widely used technique because of the capability to hide large amounts of data, and because of the simple idea [1].

The least significant bit (in other words, the 8th bit) of some or all of the bytes inside an image is changed to a bit of the secret message. Digital images are mainly of two types which are 24 bit images and 8 bit images. In 24 bit images, we can embed three bits of information in each pixel, one in each LSB position of the three eight bit values. Increasing or decreasing the value by changing the LSB does not change the appearance of the image; much so the resultant stego image looks almost same as the cover image. Meanwhile, in 8 bit images, one bit of information can be hidden. There are a lot of variations in the LSB implementation, but they are still based on the same idea, which is exploiting the least significant bits of images to hide data. This research is based on the basic LSB technique, but it is complemented with some cryptography technique on the message to be hidden before applying the LSB technique in a Portable Network Graphics (PNG) formatted image in order to enhance its security.

Several well-established research works have been done in this field. In 2007, [2] presented a method of embedding information inside a picture and putting it on a site. User will then download the picture and uses steganography software to reveal the hidden message. The method is proposed to be implemented in mobile banking. Next, [3] has proposed a method of applying randomization concept for choosing the pixel in an image to hide data. A secret key is used to generate the random values making it hard for hackers to guess the random value. The last example is on work done by [4] that introduces one-third LSB embedding that will result in less alteration in pixel values of the cover image.

Although, LSB techniques can be applied in Bitmap Image File (BMP) and Graphics Interchange Format (GIF), but their resistance to statistical counter attack and compression are weak [4]. BMP file sizes are big and improper for network transmission while Joint Photographic Experts Group (JPEG) format images use compression algorithm that does not support direct LSB embedding into the spatial domain [5].

## 2. STEGANOGRAPHY OVERVIEW

The word steganography technically means covered or hidden writing. Its ancient origins can be traced back to 440 BC. Although the term steganography was only coined at the end of the 15th century, the use of steganography dates back several millennia. In ancient times, messages were hidden on the back of wax writing tables, written on the stomachs of rabbits, or tattooed on the scalp of slaves. Invisible ink has been in use for centuries for fun by children and students and for serious undercover work by spies and terrorists [6].

p-ISSN 1675-7939; e-ISSN 2289-4934
© 2018 Universiti Teknologi MARA Cawangan Pulau Pinang

The majority of today's stenographic systems use multimedia objects like image, audio, video etc as cover media because people often transmit digital pictures over email and other Internet communication. Modern steganography uses the opportunity of hiding information into digital multimedia files and also at the network packet level [4].

Hiding information into a medium requires following elements [7]:

1. The cover medium(C) that will hold the secret message.
2. The secret message (M), may be plain text, digital image file or any type of data.
3. The steganography techniques
4. A stego-key (K) may be used to hide and unhide the message.

## 3. STEGANOGRAPHIC PROCESS

The following formula provides a very generic description of the pieces of the steganographic process:

*cover_medium + hidden_data + stego_key = stego_medium*

In this context, the *cover_medium* is the file in which we will hide the *hidden_data*, which may also be encrypted using the *stego_key*. The resultant file is the *stego_medium* (which will, of course, be the same type of file as the cover_medium). The cover_medium (and, thus, the stego_medium) are typically image or audio files. In this study, we will focus on image files and will, therefore, refer to the *cover_image* and *stego_image*. Before discussing how information is hidden in an image file, it is worth a fast review of how images are stored in the first place. An image file is merely a binary file containing a binary representation of the color or light intensity of each picture element (pixel) comprising the image.

Images typically use either 8-bit or 24-bit color. When using 8-bit color, there is a definition of up to 256 colors forming a palette for this image, each color denoted by an 8-bit value. A 24-bit color scheme, as the term suggests, uses 24 bits per pixel and provides a much better set of colors. In this case, each pixel is represented by three bytes, each byte representing the intensity of the three primary colors red, green, and blue (RGB), respectively. The Hypertext Markup Language (HTML) format for indicating colors in a Web page often uses a 24-bit format employing six hexadecimal digits, each pair representing the amount of red, blue, and green, respectively. The color orange, for example, would be displayed with red set to 100% (decimal 255, hex FF), green set to 50% (decimal 127, hex 7F), and no blue (0), so we would use "#FF7F00" in the HTML code.

The size of an image file, then, is directly related to the number of pixels and the granularity of the color definition. A typical 640x480 pix image using a palette of 256 colors would require a file about 307 KB in size (640 • 480 bytes), whereas a 1024x768 pix high-resolution 24-bit color image would result in a 2.36 MB file (1024 • 768 • 3 bytes). To avoid sending files of this enormous size, a number of compression schemes have been developed over time, notably Bitmap (BMP), Graphic Interchange Format (GIF), and Joint Photographic Experts Group (JPEG) file types. Not all are equally suited to steganography. However, GIF and 8-bit BMP files employ what is known as lossless compression, a scheme that allows the software to exactly reconstruct the original image. JPEG, on the other hand, uses lossy compression, which means that the expanded image is very nearly the same as the original but not an exact

p-ISSN 1675-7939; e-ISSN 2289-4934
© 2018 Universiti Teknologi MARA Cawangan Pulau Pinang

duplicate. While both methods allow computers to save storage space, lossless compression is much better suited to applications where the integrity of the original information must be maintained, such as steganography. While JPEG can be used for stego applications, it is more common to embed data in GIF or BMP files.

The simplest approach of hiding data within an image file is called Least Significant Bit (LSB) insertion. In this method, we can take the binary representation of the hidden_data and overwrite the LSB of each byte within the cover_image [6]. If we are using 24-bit color, the amount of change will be minimal and indiscernible to the human eye. As an example, suppose that we have three adjacent pixels (nine bytes) with the following RGB encoding:

10010101 00001101 11001001
10010110 00001111 11001010
10011111 00010000 11001011

Now suppose we want to "hide" the following 9 bits of data (the hidden data is usually compressed prior to being hidden): 101101101. If we overlay these 9 bits over the LSB of the 9 bytes above, we get the following (where bits in bold have been changed):

10010101 00001100 11001001
10010111 00001110 11001011
10011111 00010000 11001011

Note that we have successfully hidden 9 bits but at a cost of only changing 4, or roughly 50%, of the LSBs.

## 4. METHODOLOGY

This section will discuss about the encryption and decryption process involved in the secret information of hidden image. It also discuss about the algorithm apply in the process of encoding and decoding the steganography image.

### *4.1 Encryption and Decryption Process*

In encryption the secret information is hiding in with any type of image file [7]. Figure 1 shows the steps of encoding data inside an image. This flowchart contains two connectors that connect to another flowchart. For the first step, user will need to specify their key, password, text to be hidden, and the image to be used as the cover [8]. The password and text will be concatenated to produce the plaintext. The plaintext will be sent to process A for the encryption. The returned ciphertext and the image will then are sent to another process, namely B, for embedding process. If the returned image is null, an error message will be displayed and the process ends. If it is not null, the user will be prompted to choose a location to save the image, and also specify the new image of file name. Key is not allowed to be zero. If the key supplied is one, the original plaintext will be the ciphertext. If not, the ciphertext will undergo through an encryption process called the Railfence encryption with the key supplied. The produced ciphertext will be returned.

p-ISSN 1675-7939; e-ISSN 2289-4934
© 2018 Universiti Teknologi MARA Cawangan Pulau Pinang

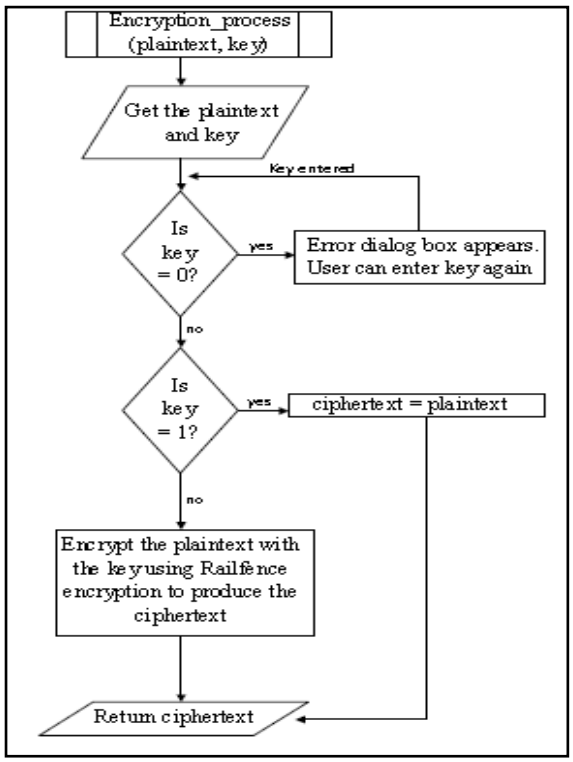


Figure 1: Flowchart of Encryption Process

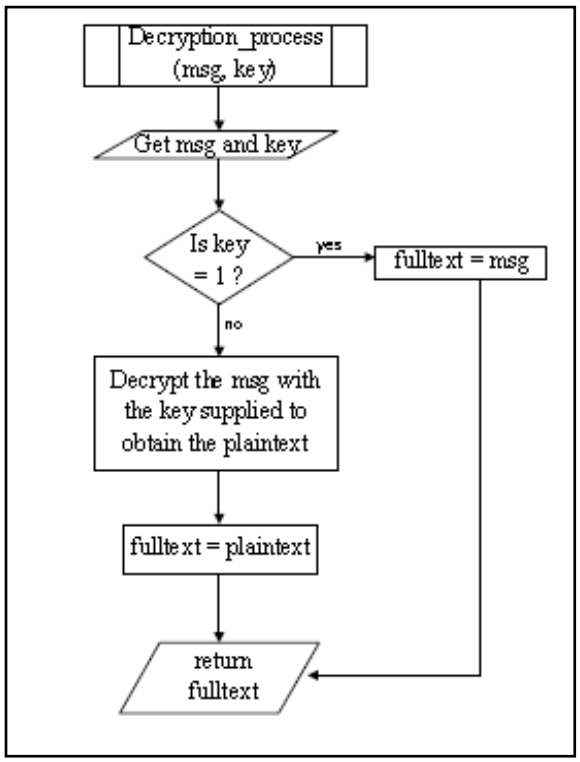


Figure 2: Flowchart of Decryption

Meanwhile, decryption is getting the secret information from image file [9]. Figure 2 shows the steps of decrypting the extracted message. If the key is one, a string variable 'fulltext' will hold the original value of the extracted message. If the key is any other value than one, the extracted message will be decrypted using the Railfence algorithm [10][11] with the key supplied. The decrypted message is known as plaintext. The variable 'fulltext' now will hold the value.

### *4.2 Embedding Algorithm*

There are many approaches available for hiding the data within an image: one of the simple least significant bit submission approaches is 'Optimum Pixel Adjustment Procedure' [12]. The simple steps for OPA explain the procedure of hiding the sample text in an image.

**Step 1**: A few least significant bits (LSB) are substituted with in data to be hidden.
**Step 2**: The pixels are arranged in a manner of placing the hidden bits before the pixel of each cover image to minimize the errors.
**Step 3**: Let n LSBs be substituted in each pixel.
**Step 4**: Let d= decimal value of the pixel after the substitution.d1 = decimal value of last n bits of the pixel.d2 = decimal value of n bits hidden in that pixel.
**Step 5**: If (d1~d2) <= (2^n)/2 then no adjustment is made in that pixel. Else
**Step 6**: If (d1d2) d = d + 2^n. The 'd' is converted to binary and written back to pixel . This method of substitution is simple and easy to retrieve the data and the image quality better so that it provides good security.

p-ISSN 1675-7939; e-ISSN 2289-4934
© 2018 Universiti Teknologi MARA Cawangan Pulau Pinang

The message of embedding procedure is given below:

$$S(i,j) = C(i,j) - 1, \text{ if } LSB(C(i,j)) = 1 \text{ and } m = 0 \quad (1)$$

$$S(i.j) = C(i,j), \text{ if } LSB(C(i,j)) = m \quad (2)$$

$$S(i,j) = C(i,j) + 1, \text{ if } LSB(C(i,j)) = 0 \text{ and } m = 1 \quad (3)$$

Where LSB (C (i, j)) stands for the LSB of cover image C (i,j) and m is the next message bit to be embedded.

The encoding process shows that the entire algorithm can be implemented by writing just a few lines of code. The algorithm works by taking the first pixel of the image and obtaining its LSB value (as per line 2 of the Algorithm) [13][14]. This is typically achieved by calculating the modulus 2 of the pixel value [15]. This will return a value 0 if then number is even, and a value 1 if the number is odd, which effectively tells us the LSB value. Then, its compare this value with the message bit that we are trying to embed. If they are already the same, then we do nothing, but if they are different then were place the pixel value with the message bit. This process continues whilst there are still values in the message that need to be encoded.

The decoder algorithm is: 1: for i = 1, ..., len (image string) do 2: message string = LSB (pixel string of the image) 3: end for. The decoding phase is even simpler. As the encoder replaced the LSBs of the pixel values in c in sequence, we already know the order that should be used to retrieve the data. Therefore, all we need to do is calculate the modulus 2 of all the pixel values in the stegogramme, and we are able to reconstruct m as m0 [16]. The above algorithms show the pseudo code of the decoding process. Note that, this time we run the loop for length of message instead of length of string [17][18]. This is because the decoding process is completely separate from the encoding process and therefore has no means of knowing the length of the message [19]. If a key was used, it would probably reveal this information, but instead we simply retrieve the LSB value of every pixel [20]. When we convert this to ASCII code, the message will be readable up to the point that the message was encoded, and then it will appear as gibberish when we are reading the LSBs of the image data.

## 5. EXPERIMENTAL RESULTS

As a performance measure for image distortion due to hiding of message, the well-known peak-signal-to noise ratio (PSNR), which is categorized under difference distortion metrics, can be applied to stego images. It is defined as:

$$PSNR = 10\log (Cmax)\ 2 / MSE \quad (4)$$

MSE = mean - square – error, which is given as:

$$MSE = 1 / MN ((S-C)\ 2.\ Cmax = 255 \quad (5)$$

where M and N are the dimensions of the image, S is the resultant stego-image, and C is the cover image.

p-ISSN 1675-7939; e-ISSN 2289-4934
© 2018 Universiti Teknologi MARA Cawangan Pulau Pinang

PSNR values below 30 dB indicate low quality (i.e., distortion caused by embedding is high). A high-quality stego image should strive for a PSNR of 40 dB, or higher. We consider RGB image as cover media (before the embedding of the secrete message) as shown in Figure 3 respectively. Text file or image taken as secrete message for the above technique.

Figure 3: Original Image

Figure 4: Stego Image

Figure 4 is the RGB image with text file respectively. By referring the both figures, we notice that human eyes cannot distinguish images before embedding the secrete message and after embedding the secrete message.

From the experiments, the performance measure for image distortion due to hiding of text message for each type of images is presented in Table 1 below.

Table 1: PSNR of Least Significant Bits Technique

| No. | Image Type | SNR | MSE | PSNR |
|---|---|---|---|---|
| 1. | Original | 61.09 | 0.056 | 62.80 |
| 2. | Stego | 62.55 | 0.012 | 57.67 |

It is possible to embed more secrete information into stego image compare to the original images. PSNR of stego image with text file as secrete message is more than the original image with text file and distortion rate is also less in stego images. Since stego image is better in terms of quality, PSNR is also higher compare to the original image.

## 6. RESULTS AND DISCUSSION

The security of steganography image process mainly resides in fact that the attacker does not know they are looking at a stego-image. If the message is hidden well enough in cover image, then the attacker will not know to check the image. However, if the attacker does try to decode the image, then they will potentially have to go through the entire key space to check for the key that decodes the image. The key space depends on image size and possible places to move or shuffle the hidden photo, but overall it is always a finite key space.

Steganography image is very applicable to this day and age as the Internet and social media are main communication avenues. A well-crafted stego-image could easily pass thousands of eyes without a second thought before it reached its destination and delivered a message. If an attacker did choose to decrypt the image, several methods of making the decryption much

p-ISSN 1675-7939; e-ISSN 2289-4934
© 2018 Universiti Teknologi MARA Cawangan Pulau Pinang

harder including the simple method, the shuffled method, and m-sequences were described in detail. Steganography image will be present in Cryptology in the years to come because it is a method that is impossible to break when it is not known that it is there.

## ACKNOWLEDGEMENTS

I would like express my special acknowledgment to my final year project student Abdul Hakim Shukor that done the project and be part of the author for this paper. Besides that, I would like to thank Dr Noraini Seman as a co-author that given many ideas in producing this paper and also other colleagues that support directly or indirectly in this project.

## REFERENCES

[1] A. Cheddad, J. Condell, K. Curran and P. Mc Kevitt, "*Biometric Inspired Digital Image Steganography*", vol. 159, 2008.
[2] M. Shirali-Shahreza. "*Improving Mobile Banking Security Using Steganography*", pp. 1-3, 2007.
[3] Dhobale, Patil, & Patil, "*Mms Steganography for Smartphone Devices*", vol. 4, pp. 513, 2010.
[4] S. Sarreshtedari, M. Ghotbi, and S. Ghaemmaghami, "*One-third Probability Embedding : Less Detectable LSB Steganography*", pp. 1002-1005, 2009.
[5] J. Fridrich, T. Pevny and Kadovsky, "*Statistically Undetectable JPEG Steganography: Dead Ends, Challenges, and Opportunities*", vol. 3, 2007.
[6] T. Hsien-Wen and C. Chin-Chen, "*Steganography Using JPEG-Compressed Images*", pp. 1-8, 2004.
[7] M. Kharrazi, H.T. Sencar and N. Memon, "*Image Steganography: Concepts and Practice*", pp. 12-18, 2004.
[8] K.B. Raja, C.R. Chowdary, Venugopal and L.M. Patnaik, "*A Secure Image Steganography Using LSB, DCT and Compression Techniques on Raw Images*", pp. 171-176, 2005.
[9] G. C. Kessler, *Steganography: Hiding Data within Data*. *Vermont: Burlington*., 2001. Retrieved March 12, 2011, http://www.garykessler.net/library/steganography.html.
[10] T. Morkel, J. H. P. Eloff and M. S. Olivier, "*An Overview of Image Steganography*", vol. 3, pp. 4-6, 2005.
[11] J. Silman, "*Steganography and Steganalysis: An Overview*", pp. 21-22, 2001.
[12] R. J. Anderson and Petitcolas, "*On the Limits of Steganography*", pp. 5-6, F.A.P 1998.
[13] B. Dunbar, "*Steganographic Techniques and Their Use in an Open-Systems Environment*", vol. 3, 2002.
[14] D. Artz, "*Digital Steganography: Hiding Data within Data*", vol. 9, pp. 12-16, 2001.
[15] G. Simmons, "*The Prisoners Problem and the Subliminal Channel*", pp. 7-9, 1983.
[16] J. Fridrich, M. Goljan and R. Du, "*Reliable Detection of LSB Steganography in Color and Grayscale Images*", vol. 27, 2001.
[17] J. Vijay Anand and G. D. Dharaneetharan, "*New Approach in Steganography by Integrating Different LSB Algorithms and Applying Randomization Concept to Enhance Security*", pp. 474 – 475, 2002.
[18] N. Memon, and R. Chandramouli, "*Analysis of LSB Based Image Steganography Techniques*", vol.1020, 2001.
[19] N. Deshpande and S. Kamalapur, "*Implementation of LSB Steganography and Its Evaluation for Various Bits*", pp. 173-176, 2009.
[20] M. Juneja, and S. Parvinder Singh, "*Designing of Robust Image Steganography Technique Based on LSB Insertion and Encryption*", pp. 302-305, 2009.

p-ISSN 1675-7939; e-ISSN 2289-4934
© 2018 Universiti Teknologi MARA Cawangan Pulau Pinang