Embedding stego-text in cover images using linked list concepts and LSB technique

Masoud Nosrati [*1], Ronak Karimi [2], Hamed Nosrati [3], Ali Nosrati [4]

[1, 2] Young Researchers Club, Kermanshah Branch, Islamic Azad University, Kermanshah, Iran.
[3, 4] Islamic Azad University, Kermanshah Branch, Kermanshah, Iran.
minibigs_m@yahoo.co.uk

Abstract: In this paper, we intend to introduce a steganography algorithm for embedding a message into a RGB 24-bit color image. It will be done by using the concepts of linked list data structure. It will help us to achieve some important advantages. First, we can create a "stego-key" by the address of message blocks. Second, it makes the detection of message harder. Also, there will be other benefits that are mentioned during the paper.
Another point about the presented algorithm is the flexibility. For example, it could be written in recursive way. To prove it, we wrote a recursive function called "Read()" for extracting the message from the cover image. At the end of paper, characteristics of this algorithm will be talked.
For embedding data, LSB (Least Significant Bit) technique is been used.
[Masoud Nosrati, Ronak Karimi, Hamed Nosrati, Ali Nosrati. Embedding stego-text in cover images using linked list concepts and LSB technique. Journal of American Science 2011;7(6):97-100]. (ISSN: 1545-1003). http://www.americanscience.org.

Keywords: Steganography; secure communication; data covering; carrier image; linked list; LSB.

1. Introduction

On line facilities are closely tied with the issues concerning availability, integrity, confidentiality and authentication of information exchanged over communication media, which has lead to the evolution of information hiding techniques for securing communication [1]. To do this, one of the common strategies is using steganography algorithms. The word steganography is derived from the Greek words "stegos" meaning "cover" and "grafia" meaning "writing" defining it as "covered writing" [2]. Steganography is one such pro-security innovation in which secret data is embedded in a cover [3]. In other words, steganography is the process of hiding a secret message within a larger one in such a way that someone cannot know the presence or contents of the hidden message. Although related, Steganography is not to be confused with Encryption, which is the process of making a message unintelligible - Steganography attempts to hide the existence of communication [4]. The notion of data hiding or steganography was first introduced with the example of prisoners' secret message by Simmons in 1983 [5].

In steganography we are faced with two types of components: message and carrier. Message is the secret data which should be hidden; and carrier is the context that hides the message in it. Carrier can be of any types of data such as text, image, audio, etc. Message with embedded hidden information is called "stego-text" [6].

In this paper, we are going to hide a binary message in an image as the carrier material which we call it "cover image". Message will be embedded sporadically with a structure like linked list, and random locations of its data blocks. By this, we are going to achieve two important goals:

a) Make the detection of message harder to gain to stricter security.
b) Create a security key for extracting message. Since the head of the message has a random location in the cover image, so the initial address of it can be used as a key.

For embedding the message in a 24-bit RGB image, we use the LSB (Least Significant Bit) technique. So, in the second section which is titled "Background" we will talk about basic concepts about RGB color space and the LSB technique fundamentals. In the third section that is named "Linked list structured message embedding", we will get into the structure of message and the number of pixels needed to store it. Main part of this section is devoted to embedding algorithm. In the fourth section, titled as "Discussion on features" we are going to talk about the characteristics, advantages and uses of this technique. Finally, "Conclusion" is placed in the end of paper.

2. Background

An image can be represented by a collection of color pixels. The individual pixels are represented by their optical characteristics like "brightness", "chroma" etc. Each of these characteristics can be digitally expressed in terms of 1s and 0s [7]. There are different color spaces that present different forms for storing images. A color space is a method by

which it is possible to specify, create and visualize color [8]. The most common color space among all is RGB (Red, Green, Blue). Each pixel in a 24-bit bitmap image in this space is described by 3 sets of 8 bits (3 bytes), that each set contains the intensity value of individual red, green and blue. Combination of these values forms the characteristics of the pixel. Figure 1 illustrates this matter.

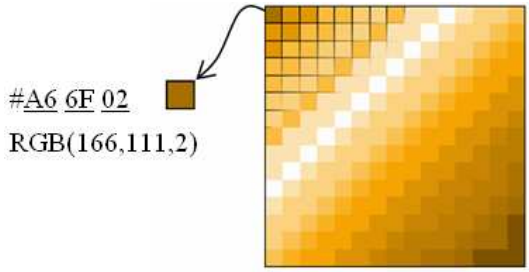

Figure 1. A pixel in RGB color space

Least Significant Bits (LSB) insertion is a simple approach for embedding information in image file. The simplest steganography techniques embed the bits of the message directly into least significant bit plane of the cover image in a deterministic sequence. Modulating the least significant bit does not result in human-perceptible difference because the amplitude of the change is small [9]. To hide a secret message inside an image, a proper cover image is needed. Because this method uses bits of each pixel in the image, it is necessary to use a lossless compression format, otherwise the hidden information will get lost in the transformations of a lossy compression algorithm. When using a 24-bit color image, a bit of each of the red, green and blue color components can be used, so a total of 3 bits can be stored in each pixel. For example, the following grid can be considered as 3 pixels of a 24-bit color image, using 9 bytes of memory:

(00100111 11101001 11001000)
(00100111 11001000 11101001)
(11001000 00100111 11101001)

When the character A, which binary value equals 10000001, is inserted, the following grid results:

(00100111 11101000 11001000)
(00100110 11001000 11101000)
(11001000 00100111 11101001)

In this case, only three bits needed to be changed to insert the character successfully. On average, only half of the bits in an image will need to be modified to hide a secret message using the maximal cover size. The result changes that are made to the least significant bits are too small to be recognized by the human visual system (HVS), so the message is effectively hidden [10].

As you see, the least significant bit of third color is remained without any changes. It can be used for checking the correctness of 8 bits which are embedded in these 3 pixels. In other words, it could be used as “parity bit”.

3. Link list structured message embedding

In this section we are going to embed the message in cover image with a structure like what linked lists place in the memory. As you know, linked list is a data structure like an array, but the most important difference between them is in their placement in RAM. Arrays sit in sequential order of memory places, but linked lists get sporadic addresses, and each item (which is called “node”) stores proposed data and also the address of the next item in the memory [11]. Using this concept, we will embed the separate bytes of message sporadically in cover image, so that, address of the next byte far apart in the image be placed just after embedding each byte. By this, two advantages will be achieved: First, non sequence of message structure makes the detection harder, and it increases the security level. Second, the address of first byte of message could be used as stego-key. As you know, while working with linked lists, always the address of first node is stored in a pointer for accessing the linked list data. Losing this address means losing the data stored in linked list. So, we can take this concept to work in steganography for creating a key for message. Figure 2 shows the structure of message in the cover image.

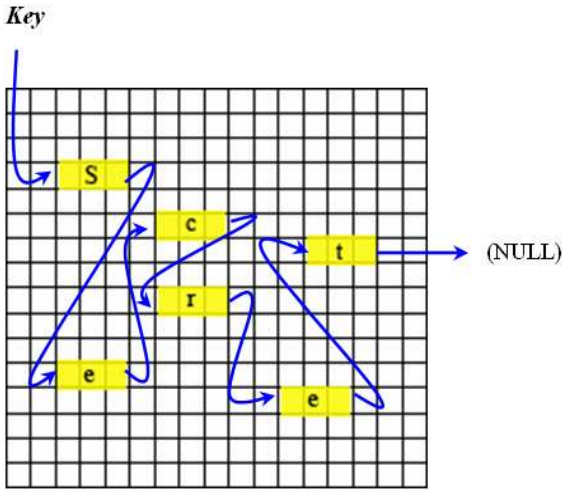

Figure 2. Embedding a linked list structured message in cover image

Another important point we have to consider is about the random location of message bytes. Each image provides a 2 dimension *(X,Y)* space for putting the message in it. Now, what we need is an algorithm to generate the address of no repeated locations. There are some algorithms for it. For example, you can suppose the image as a simple 1 dimension array, and then do the block scheming. Each block can be a set of pixels for storing a byte of message and also some extra pixels to store the next byte address. The number of pixels in a block depends on the size of image. For an image with *x*y* pixels, following equation can be used to determine the number of pixels needed for storing the address:

(I)

$$p = \left\lceil \frac{k}{3} \right\rceil : x * y \le 2^k$$

That *p* is the number of pixels for address. There are *(x*y)* items that should be addressed. So, we need *k* bits so that $x*y \le 2^k$. It can be concluded that the number of pixels will be equal to *p* in (I).

For example, suppose that there is a 20*30 pixels image. For addressing:

$$20 * 30 \le 2^{10}$$

$$p = \left\lceil \frac{10}{3} \right\rceil = 4(pixels)$$

So each block in this image should contain 3 pixels for a byte of message, plus 4 pixels for address of the next byte.

After block scheming, now you can use the random numbers generation algorithms to choose the blocks for storing data in them. We aren't going to describe these algorithms. Just suppose that there is a function which is named "*RandomBlocks()*" and it does the block scheming and random block selection. For more information about generating random numbers see [12].

Here, we present an algorithm for embedding the message in a cover image with linked list structure. It is done by a function which we call it "*write()*".

```
Class block
{
    block();  // Constructor
    void SetData(byte);  //Sets the byte for data part of current block
    void SetLink(block); //Sets the block for link part of current block
    byte GetData();  // returns the Data part of current block
    block GetLink();  // returns the Link part of current block
    block GetAddress();  // returns the address of current block
}
```

```
function Write(message)
{
    new=RandomBlock();
    new.SetData(First byte of message);
    key=GetAddress();
    previous=new;

    for each byte in message  // From second byte
    {
        new=RandomBlock();
        new.SetData(byte);
        previous.SetLink(new.GetAddress);
        previous=new;
    }
    previous.SetLink(NULL);
}
```

For reading the message a recursive algorithm is presented. It is done by *read()* function. While calling this function for the first time, *key* should be sent as a parameter of this function.

```
byte Read(block)
{
    if (block.GetLink==NULL) return block.GetData;
    else return read(block.GetLink);
}
```

4. Discussion on features

In this section, we are going to get into the characteristics of this algorithm. So, the following lines can be pointed out:

- Presented algorithm can be used for any kind of message embedding such as text, images and even the files; because all of them can be reached in bytes form.
- Considering equation (I), maximum bytes of the message that can be embedded in an image with *x*y* pixels (n_b) is calculated as follow:

(II)

$$n_b = \frac{\sum(pixels)}{p+3} = \frac{x*y}{p+3}$$

- It is easily possible to add new blocks of data in the chain of message. Also, removing them could be done easily.
- More than one message can be embedded in a cover image. It means we can have more than one chain of message with different keys. This feature could be very useful when the receiver of message is more than one, and each of them should receive their own message.
- This algorithm can be used as a layer of programming in the process of securing data.

5. Conclusion

In this paper, we talked about basic notions of steganography and also took a look at LSB embedding technique in RGB 24-bit color images. After that, a way for image block scheming was introduced, in order to create a structure like linked lists. Also, some rules were defined to set the size of each block. It was mentioned that the goal of block scheming was creating stego-key and making the detection of message harder. Finally, the algorithm for embedding the message was presented, and its characteristics were talked.

Corresponding Author
Masoud Nosrati
Department of Computer Engineering
Islamic Azad University, Kermanshah Branch, Young Researchers Club, Kermanshah, Iran.
E-mail: minibigs_m@yahoo.co.uk

References

1. Kapil Juneja, Archana, Meenakshi, Covering Data in Video Files by using Tree Dimensions of Data Security, International Conference on Science and Engineering, Rohtak, India, 2011.
2. Sara Khosravi, Mashallah Abbasi Dezfoli, Mohammad Hossein Yektaie, A new steganography method based HIOP (Higher Intensity Of Pixel) algorithm and Strassen's matrix multiplication, Journal of Global Research in Computer Science, Vol. 2, No. 1, 2011.
3. S. Katzenbeisser, F.A.P. Petitcolas, Information Hiding Techniques for Steganography and Digital Watermarking, Artech House, Norwood, MA, 2000.
4. Nick Nabavian, CPSC 350 Data Structures: Image Steganography, 2007, Available at: http://www1.chapman.edu/~nabav100/ImgSteganо/download/ImageSteganography.pdf
5. G. J. Simmons, "The prisoners' problem and the subliminal channel" in Proc. Advances in Cryptology (CRYPTO '83), pp. 51-67.Berglund, J.F. and K.H. Hofmann, 1967. Compact semitopological semigroups and weakly almost periodic functions. Lecture Notes in Mathematics, No. 42, Springer-Verlag, Berlin-New York.
6. Christian Cachin, Digital Steganography, Encyclopedia of Cryptography and Security, 2005.
7. Soumyendu Das, Subhendu Das, Bijoy Bandyopadhyay and Sugata Sanyal, "Steganography and Steganalysis: Different Approaches", International Journal of Computers, Information Technology and Engineering (IJCITAE), Vol. 2, No 1, June, 2008, Serial Publications.
8. Adrian Ford, Alan Roberts, Colour Space Conversions, 1998, Available at: http://www.poynton.com/PDFs/coloureq.pdf
9. Mohamed Amin, Muhalim and Ibrahim, Subariah and Salleh, Mazleena and Katmin, Mohd Rozi (2003) Information hiding using steganography. Project Report. Available at: http://eprints.utm.my/4339/1/71847.pdf
10. Robert Krenn, Steganography and steganalysis, Internet Publication, March 2004. Available at: http://www.krenn.nl/univ/cry/steg/article.pdf
11. Elis Horowitz, Sartag Sahni, Dinish Mehta: Fundamentals of Data Structures in C++, 2nd ed. Silicon Press, 2006.
12. Pierre L'Ecuyer: Comparison of Point Sets and Sequences for Quasi-Monte Carlo and for Random Number Generation. SETA 2008.

5/7/2011