Research Journal of Applied Sciences, Engineering and Technology 4(5): 491-494, 2012
ISSN: 2040-7467
© Maxwell Scientific Organization, 2012
Submitted: October 26, 2011 Accepted: November 25, 2011 Published: March 01, 2012

LSB Replacement Stegnography in an Image using Pseudorandomised Key Generation

B. Karthikeyan, V. Vaithiyanathan, B. Thamotharan, M. Gomathymeenakshi and S. Sruti
School of Computing, SASTRA University, Thanjavur, Tamilnadu, India

Abstract: With the development in technology, data transfer of secret information plays a major role. We go for data security in order to prevent the data from hackers and malware infection. In this study, we dealt with the secret key stegnography method of hiding a plaintext in an image using a random key. The method adopted is highly secured and there is no explicit change in the original image even if long plaintext is inserted in the image.

Key words: EBCDIC, key, plaintext, pseudorandom generator, stegnography, stegoimage

INTRODUCTION

In the present world, communication is one of the important factors in the information technology. It is necessary to have the existence of the plaintext secret along with the cryptographic techniques. Implementation of this method is referred as Stegnography. Stegnography (Bret, 2002) refers to the art of secret communication. Despite of hiding the plaintext in another text, audio or using invisible ink, and image stegnography is considered more secured due to the limited power of human visual system. Stegnography is intended for secrecy and is not intelligible except to the persons who have the key.

Secret key Stegnography (Eric, 2003) is a stegnographic system that requires exchange of key prior to communication. It takes the cover message and embeds the secret information inside it using a secret key. Only the users who know the key can reverse the process and read the secret information. The advantage of this technique is that how much ever it is interception susceptible, only the parties who know the key can access the secret message.

In this study, the intention is to develop a stego-image by hiding the plaintext into the image using randomly generated key. With the help of EBCDIC code, the plaintext is converted into numbers, followed by the permutation by using the key. The resulting plaintext is XORed with the key and is inserted in the image. Randomizing the key using pseudorandom generator for every plaintext and using the key for insertion of plaintext in the image is the significant feature that increases the security level of secret communication.

METHODOLOGY

Proposed work: The study explains about the way in which information is made secret between two parties on the usage of key in the plaintext transferred and finally embedding the plaintext in an image, which is then transmitted.

Let us take a plaintext P of 256 characters. It is converted into integer numbers using EBCDIC code. It is in the form: P = P(i, j) where I = 1 to 16 and j = 1 to 16.

The key is generated by the pseudorandom generator which will be of the form : K(i, j) where I = 1 to 16 and j = 1 to 16, where each value is lying in the interval from 1 to 256. The upcoming algorithm elucidate about the generation of the random keys using pseudorandom generator.

Pseudorandom generator: The procedure generates a list of random numbers when maximum limit is provided as input. Every time the generator produces the values without any repetition.

Algorithm:

```
• Read the maximum limit of keys to be generated.
• For i ←1 to maximum limit
    ◦ For j ←1 to maximum limit
    • Generate a random number(r) within the
      maximum limit using nextInt() in java (Herbert
      Schildt, 2006).
    • If r is not in K then
        ◦ Include r in K array.
```

Once the key and the plaintext are available, the security level of the plaintext can be increased by performing operations like permutation and XOR operation with the help of the generated random keys.

Permutation: The plaintext is permuted using the key and is denoted as M (i, j) where I = 1 to 16 and j = 1 to 16. The permutation technique is as follows:

Corresponding Author: B. Karthikeyan, School of Computing, SASTRA University, Thanjavur, Tamilnadu, India

Let K(i, j) = D.

We can assume that D is of the form:

D = 16v+w

Integers v and w are lying in the interval from 1 to 256. When w = 0, the last row vth column element of the P will be placed as the ith row jth column element of the matrix M. On the other hand, when w $\neq$ 0, the (w+1)th row vth column element of the P will be placed as the ith row jth column element of M. The above procedure is the permutation process shown in Fig. 3.

XOR operation: XOR operation increases the strength of any stegnographic method. Initially every number in both plaintext (M) and the key are converted to its binary equivalent. XOR operation is performed between numbers in the plaintext and the key matrix, which are placed in the same position. Thus we XOR the resulting plaintext (M) with key (K):

$$M = M \oplus K.$$

We get the modified plaintext in M. The next step is to insert this modified plaintext into the desired image. Let an image I be of the form I(i, j) where i = 1 to 256 and j = 1 to 256. The forthcoming procedure is applicable for both color level and gray level image. The color level image cannot be taken as such for the process. For our convenience, any color level image is converted to gray scale and then processed. We go for the conversion because of simplicity in processing the image. Conversion can be done by any tool. In our work, we make use of MATLAB. An example syntax for conversion in MATLAB (Rudra, 2002) is I = rgb2gray(H), Here H represents the pixel values of the color level image. Thus I contains the equivalent pixel values for gray level image.

Each pixel value of the image is of 8 bits. The insertion of the plaintext in the image should be in such a way that there should not be any difference between the original image and the stegoimage to the human visual system. The method for inserting the plaintext in the image is as follows:

Let K(1, 1) = s

Let us spotlight the attention on the sth column of I, i.e., j = s. We will convert M_{11} into its binary format so that we get a string containing 8 binary bits. Then having the first six-bits of each value of I_{ir}, i = 1 to 4, we go on doing AND operation on the last two bits of the gray level value and the first two bits of the M_{11}. The computation of logical operation between the last two bits of each pixel in the image with the extracted plaintext is for acquiring additional data security from steganalysis through brute

Fig. 1: Actual image

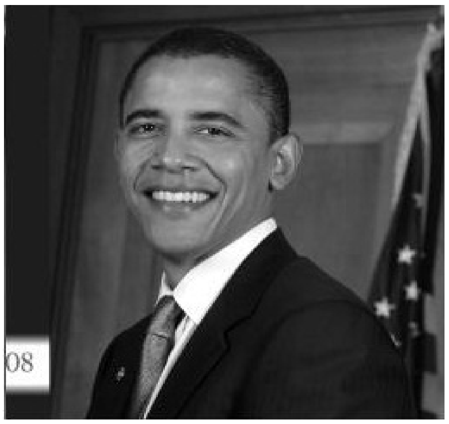
Fig. 2: Stegoimage after inserting the plain text

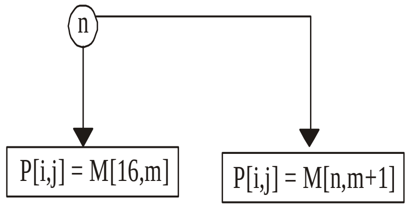

Fig. 3: Flow Chart of Permutation function(f( ))

force approach. Then we go for concatenating the resulting two bits in the 1st row, the next 2 bits of M_{11} after the AND operation in the next row, etc., till we reach the 4th row and tire out the eight binary bits. The embedding of the plaintext into the image goes column by column until entire plaintext is completed. The final image with the plaintext is termed as stegoimage, which is then transferred along with the key. The entire process is depicted in Fig. 4.

Decryption: In secret key stegnography, both insertion and retrieval of plaintext is performed using a single key. Decryption involves exact reverse process of encryption

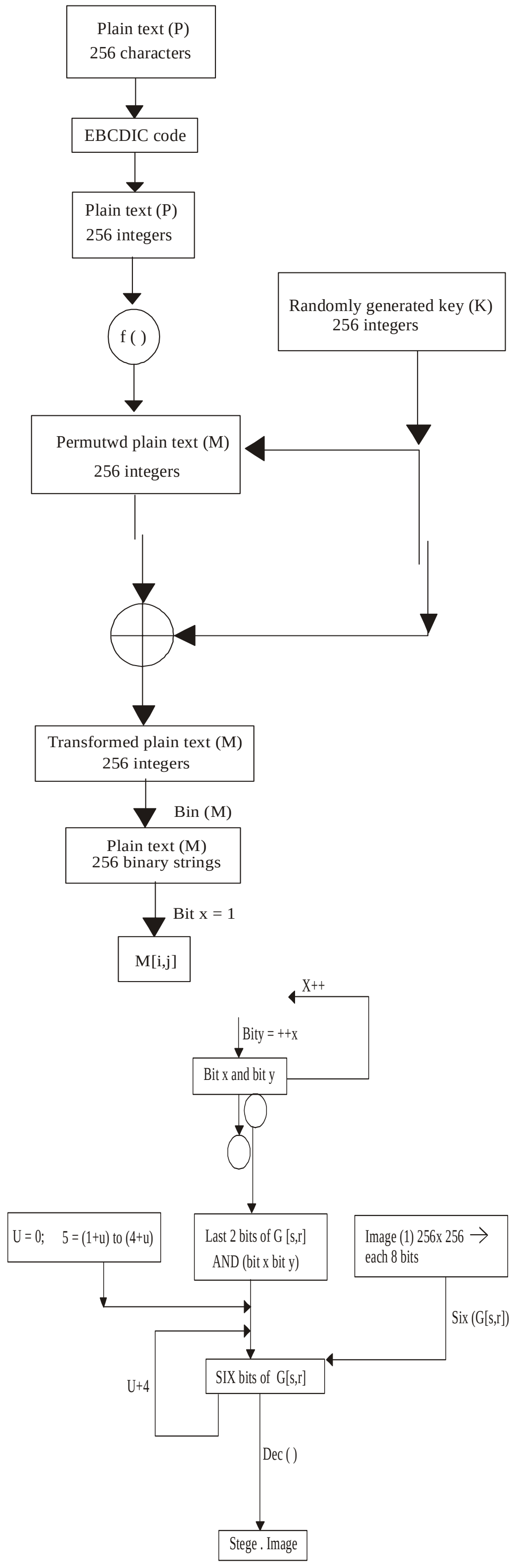

Fig. 4: The Flow chart of pseudo randomized key generation and proposed algorithm

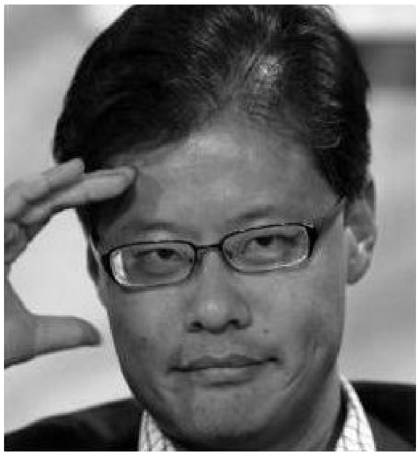

Fig. 5: Actual image

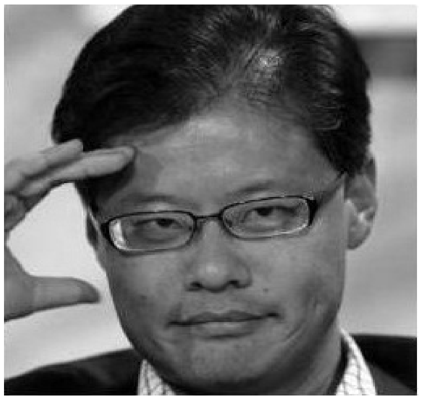

Fig. 6: Stegoimage after inserting the plain text

algorithm using the same key. The user at the other end can retrieve the plaintext from the Stegoimage till the key remains secret between the two end users. If the key is revealed, then the secrecy is not maintained.

The decryption process involves the following steps: The pixel values in stegoimage are converted into its binary equivalent. The last two bits concatenated at every pixel of the Stegoimage are collected to get the plaintext in binary format which is then converted into characters. Figure 1 and 5 depicts the original image where the plaintext of 256 characters has to be inserted. Figure 2 and 6 depicts the stegoimage thus formed after the insertion of plaintext using pseudorandomised key with XOR and logical operation.

RESULTS AND DISCUSSION

This study revolves around embedding the desired plaintext in an image using a randomized matrix as key. The plaintext is hidden column wise according to the algorithm followed for the concealing process. As the key

Table 1: Performance comparisons of image

Image	MSE	PSNR
Fig. 1	0.00178	75.64
Fig. 5	0.00086	78.77

generated is randomized in a matrix, the plaintext thus gets randomized while inserting it in the original image.

The XOR and AND operation performed during the operations strengthens the process by reducing the errors in the Stegoimage. For the Fig. 1 the MSE (Mean Square Error) (Bhabdhosh, 2006) is found to be 117 out of 256 X 256 pixels and for Fig. 5, it is found to be 88 out of 256 X 256 pixels. When the last two bits of each pixel in the original image is replaced with the two bits from each plaintext byte, it is simple LSB substitution. It is found that when we do MSB substitution, the Stegoimage is prominently seen for some insertion of foreign elements in the original image.

Table 1 provides the relationship between the error metrics of the image like Mean Square Error and Peak Signal to Noise Ratio. Generally for a perfect stegnographic process, the value of MSE should be minimum and the value of PSNR should be maximum. From the Table 1, it is understood that the above procedure reached its high level of security.

REFERENCES

Bret, D., 2002. A detailed look at Steganographic Techniques. SANS Institute, Infosec Reading Room.

Bhabdhosh, C., 2006. Digital Image Processing and Analysis, 8th Edn., Prentice Hall, India.

Eric, C., 2003. Hiding in Plain Sight, Steganography and the art of Covert Communication. Wiley Publishing, US.

Herbert, S., 2006. Java: The Complete Reference. 7th Edn., McGraw Hill, US.

Rudra, P., 2002. Getting Started With MATLAB: A Quick Introduction for Scientists and Engineers, Version 6, Oxford Press, India.