Volume 3, No. 4, April 2012

Journal of Global Research in Computer Science

ISSN-2229-371X

RESEARCH PAPER

Available Online at www.jgrcs.info

PUBLIC-KEY STEGANOGRAPHY BASED ON MODIFIED LSB METHOD

Vivek Jain[1], Lokesh Kumar[2], Madhur Mohan Sharma[3], Mohd Sadiq[4], Kshitiz Rastogi[5]
[1]Assistant Professor of CSE department, IMS Engineering College, Ghaziabad, 201009, India
lokeshk_gautam@yahoo.com
[2]Assistant Professor of MCA department, IMS Engineering College, Ghaziabad, 201009, India
[3, 4, 5] Student of CSE department, IMS Engineering College, Ghaziabad, 201009, India

Abstract: Steganography is the art and science of invisible communication. It is a technique which keeps the existence of the message secret. This paper proposed a technique to implement steganogaraphy and cryptography together to hide the data into an image. This technique describes as: Find the shared stego-key between the two communication parties by applying Diffie-Hellman Key exchange protocol, then encrypt the data using secret stego-key and then select the pixels by encryption process with the help of same secret stego- key to hide the data. Each selected pixel will be used to hide 8 bits of data by using LSB method.

Keywords: Steganography, Public-key, Stego-key, Diffie-Hellman, LSBs

INTRODUCTION

When we communicate through insecure network the main issue arises is security. For the secure communication we use different techniques such as cryptography, Steganography etc. Cryptography was created as a technique for securing the secrecy of communication and many different methods have been developed to encrypt and decrypt data in order to keep the message secret. Unfortunately it is sometimes not enough to keep the contents of a message secret, it may also be necessary to keep the existence of the message secret. The technique used to implement this, is called Steganography [1].

Cryptography:

Cryptography is the science of secret writing. The goal of cryptography is to make data unreadable by a third party. Cryptography algorithms are divided into two parts, secret-key (symmetric) and public-key (asymmetric) algorithms [2]. Symmetric algorithms are used to encrypt and decrypt original messages (plaintext) by using the same key. While Public-key encryption algorithms work in a different way. In these algorithms, there is a pair of keys, one key is known to the public, and is used to encrypt information to be sent to a receiver who owns the corresponding private key which is used to decrypt the information.

Steganography:

Steganography is the art and science of communicating in such a way that the presence of a message cannot be detected [3]. Steganography provides a means of secret communication which cannot be removed without significantly altering the data in which it is embedded. The message can be hide in audio, video, text or any other digitally represented code. The hidden message will be confidential unless an attacker can find a way to detect it. The hidden message may be plaintext, cipher text or anything that can be represented as a bit stream. Images are ideal for information hiding because of the large amount of redundant space is created in the storing of images [4]. There are two types of image compressions, lossy compression and loss less Compression; lossless compression formats (e.g. GIF and BMP formats) maintain the original image's integrity. Lossy compression (e.g. JPEG format) may not maintain the original image's integrity [5]. Lossless compression maintains the original image data exactly, hence it is preferred. Simple steganographic techniques have been in use for hundreds of years, but with the increasing use of files in an electronic format new techniques for information hiding have become possible. To make a steganographic communication even more secure the message can be encrypted before being hidden in the carrier [6]. Cryptography and Steganography can be used together. Steganography and cryptography are both used to ensure data confidentiality [7]. However the main difference between them is that with encryption anybody can see that both parties are communicating in secret. Steganography hides the existence of a secret message and in the best case nobody can see that both parties are communicating in secret.

RELATED WORK

The most of today's steganographic systems use images as cover object because people often transmit digital images over email and other communication media. The main method to apply steganography technique on images is LSB technique which described below.

The Lsb Technique:

In least significant bit technique data can be hiding in an image by changing the LSB of each color byte as data bit [8]. In a 24- bit image, one can store 3 bits in each pixel by changing a bit of each of the red, green and blue color components, since they are each represented by a byte. A 600×500 pixel image, can thus store a total amount of 900,000 bits or 112,500 bytes of embedded data. As an

example, suppose that we have three adjacent pixels (9 bytes) with the RGB encoding.
10110011 01101101 11001101
10011010 01001111 11001111
10010111 01010010 10011011

When the number 330, can be which binary representation is 101001010 embedded into the least significant bits of this part of the image. 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)
10110011 0110110**0** 11001101
10011010 0100111**0** 11001111
1001011**0** 0101001**1** 1001101**0**

Here the number 330 was embedded into the grid, only the 5 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. The human eye cannot perceive these changes - thus the message is successfully hidden. With a well-chosen image, one can even hide the message in the LSB without noticing the difference [9].

Diffie-Hellman Key Exchange Protocol:

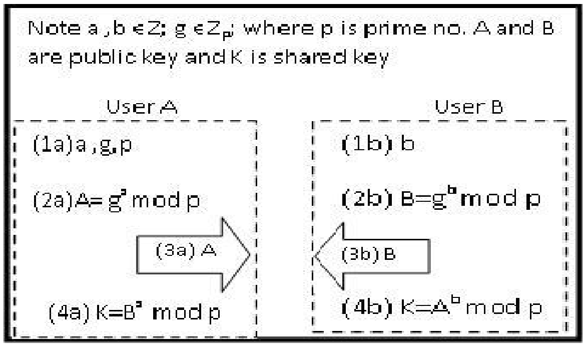
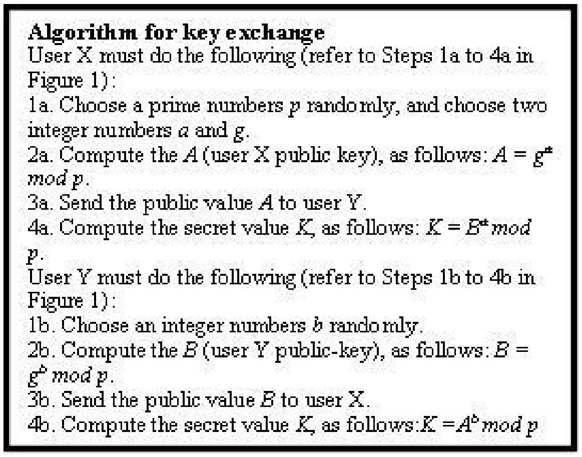


Figure: 2 (a)

The security of the Diffie-Hellman key exchange protocol is based on the strength of the discrete algorithm and the size of the key used (refer to Figure 2(b)). However the Diffie-Hellman protocol is considered secure against brute force attack if *p* and *g* are chosen properly [10]. Currently, the solving of the Diffie-Hellman discrete logarithm problem will make many other public-key cryptosystem insecure. The Diffie-Hellman key exchange has a high level of the security because DH protocol depends on large prime numbers which exceeds 1024 bit.



Algorithm for key exchange
User X must do the following (refer to Steps 1a to 4a in Figure 1):
1a. Choose a prime numbers *p* randomly, and choose two integer numbers *a* and *g*.
2a. Compute the *A* (user X public key), as follows: $A = g^a \bmod p$.
3a. Send the public value *A* to user Y.
4a. Compute the secret value *K*, as follows: $K = B^a \bmod p$.
User Y must do the following (refer to Steps 1b to 4b in Figure 1):
1b. Choose an integer numbers *b* randomly.
2b. Compute the *B* (user Y public-key), as follows: $B = g^b \bmod p$.
3b. Send the public value *B* to user X.
4b. Compute the secret value *K*, as follows: $K = A^b \bmod p$

Figure: 2(b)

PROPOSED WORK

The proposed method describes as: Find the shared stego- key between the two communication parties (user A and user B) by applying Diffie Hellman Key exchange protocol. As shown by Figure 2(b), Diffie-Hellman key exchange protocol shows the technique for the key exchange between two parties (user A and user B) to get shared Stego-key values. As in Figure 2(b), User A must generate the keys (public and private keys) and use her private keys to give new public key and send it to user B's side. User B must obtain and issue new public keys. Then at the end the protocol, each side recovers his/her received public key to reach the shared values between them, that's mean user A and user B have arrived same sego-key value.

Then encrypt the information by using the shared stego-key which already generated in the previous step. Here we can use any standard encryption algorithm for encrypt the information. (K, PT) = CT where, K=shared stego key, PT=plain text (original data), CT=cipher text, PT^i $=i^{th}$ byte of original data, CT^i $=i^{th}$ byte of encrypted data. In the next step we select the pixel of image by using the shared stego- key for hiding the CT. The algorithm for selecting the pixel as follows: Let M= total no. of pixels in the cover image (m1*m2), K=Stego-key which is generated by Diffie-Hellman algorithm. We have to hide the information in a 24 bit image byte-wise. So we hide CT^i in image where i=1, 2, 3… n. (where n=size of data which should be less than or equal to M). s is the pixel number of image(m1*m2) which is calculated by co-ordinates(x, y) of that particular pixel as follows: s = (y*m1+x+1).

Algorithm at Sender Side:

```
Algo1: Take an array A [M] where initially A[i] =0 where (1<= i<= M)
  For (i =1 to n)
    Step 1: E (K, i) = j
    Step 2: s = j mod M
    Step 3: if (A[s] == 0)
```

```
        Call algo3
        Set A[s] =1
      Else
        s++
        If (s>M) s=1
Goto step 3
```

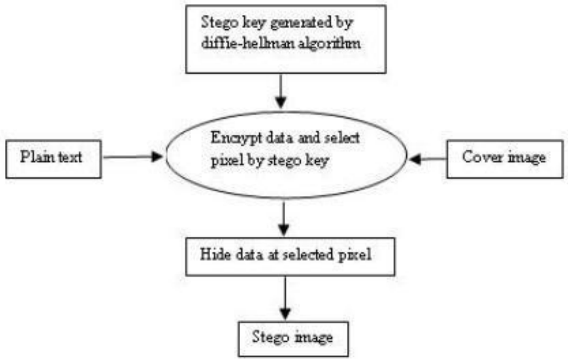

Stego key generated by diffie-hellman algorithm
Plain text
Encrypt data and select pixel by stego key
Cover image
Hide data at selected pixel
Stego image

Figure: 3(a)

//* we can easily calculate co-ordinate (x, y) from s as follows: x=(s-1) %m1 and y=(s-1)/m1 *//

Algorithm at receiver side:

Algo2: Take an array B [M] where initially B[i] =0 where (i=1 to M).

```
 For (i=1 to M)
     Step 1: E (K, i) =j Step
     2: s = j mod M Step 3:
     if (B[s] == 0)
             Call algo4
             Set B[s] =1
           Else
             s++;
              If (s>M)
                s=1
             Goto step3
      Step 4: PT^i = D (K, CT^i)
Step 5:  if (PT^i == ending flag) {exit}
```

//*ending flag is a special character which has already decided between sender and receiver. It shows the ending of message.*//

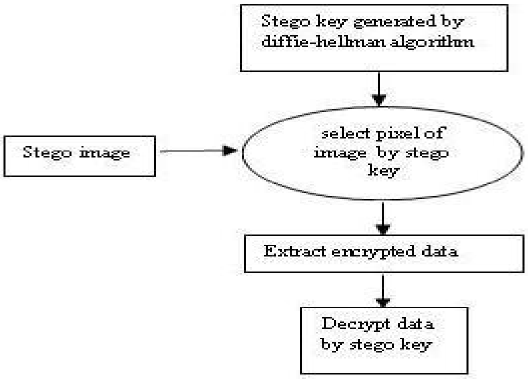

Stego key generated by diffie-hellman algorithm
Stego image
select pixel of image by stego key
Extract encrypted data
Decrypt data by stego key

Figure: 3(b)

Modified LSB technique:

r[M], g[M], b[M], r1[M], g1[M], b1[M] are the array which have the values of red, green, blue color of pixels in cover image and stego image respectively.

```
At sender side: Algo3:
Step 1:  t = r[s] AND 0xF8
Step 2:  t1 = CT^i AND 0x07
Step 3:  r1[s] = t OR t1
Step 4:  u= g[s] AND 0xF8
Step 5:  u1 = CT^i  AND 0x38
Step6:   u2 =u1 SHR3
Step7:   g1[s] = u OR u2
Step8:   v = b[s] AND 0xFC Step9:   v1 = CT^i  AND 0xC0
Step10: v2 = v1 SHR6
Step11: b1[s] = v OR v2
At receiver side: Algo4:
Step 1:  t1 = r1[s] AND 0x07
Step 2:  t2 = g1[s] AND 0x07
Step 3:  t3= b1[s] AND 0x03
Step 4:  t4 = t2 SHL 3
Step 5:  t5 = t3SHL 6
Step 6: CT^i = t1 OR t4 OR t5
```

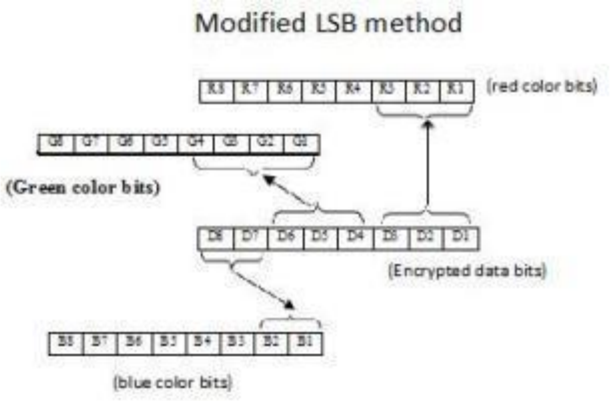


Figure: 3(c)

WORKING AND ANALYSIS

We have to hide some information in an image of size N*N. CT is the encrypted form of our plain text using key K (let K=1010110001) which is generated by diffi-hellman key exchange algorithm at both sender and receiver side. Suppose we have already hidden 204 bytes of our encrypted data and now we are hiding 205th byte of our encrypted data i.e. CT^{205}.

Suppose N=50, E
(K, 205) =41;
s = 41 mod 2500=41
Now assume A [41] ==0
/* now we hide the CT^{205} at the 41th pixel of image */

Suppose the r, g, b value of the 41th pixel of image are
01001011(75), 10101011(171), 01110110 (118)
respectively and CT^{205} is
11001011. t= r [41] AND 0xF8 =
01001000
t1 = CT^{205} AND 0x07
=00000011
r1[41] = t OR t1 = 01001011(75)
u = g [41] AND 0xF8 = 10101000
u1 = CT^{205} AND 0x38 =
00001000 u2 = u1 SHR3 =
00000001
g1 [41] = u OR u2 = 10101001(169)
v= b [41] AND 0xFC = 01110100(118)
v1 = CT^{205} AND 0xC0 = 11000000
v2 = v1 SHR6 = 00000011
b1 [41] = v OR v2 = 01110111(119)

As we can see that initial values of r, g, and b of 41th pixel were 75, 171 and 118 respectively and the values of r, g, and b of 41th pixel after hiding the data using this proposed algorithm are 75, 169, and 119. This minor change cannot be perceptualzing by human eye and so no one can suspect on our image that it has any hidden data.

Now we send our stego image by any transmission media to the receiver which has already stego key which is generated by diffie-hellman key exchange algorithm. Now receiver performs the following actions: Suppose receiver unhides the 205th byte of information. First he encrypts the 205 by using key K and gets 41.

t1= r1 [41] AND 0x07 = 00000011
t2= g1 [41] AND 0x07 = 00000001
t3 = b1 [41] AND 0x03 = 00000011
t4 = t2 SHL3 = 00001000
t5 = t3 SHL6 = 11000000
CT^{205} = t1 OR t4 OR t5 =
11001011.

Here we can see that the 205th byte of our encrypted data is exactly same as the data was at sender side. Now receiver

REFERENCES

[1]. Silman, J., “Steganography and Steganalysis: An Overview”, SANS Institute, 2001

[2]. Menezes, A., van Oorschot, P., and Vanstone, S., (1996) “Handbook of Applied Cryptography,”CRC Press, pp.4, 15, 516

[3]. William Stallings, (2006) “Cryptography and Network Security Fourth Edition”.

[4]. Johnson, N.F. & Jajodia, S., “Exploring Steganography: Seeing the Unseen”, Computer Journal, February 1998

[5]. Currie, D.L. & Irvine, C.E., “Surmounting the effects of lossy the can easily decrypt this CT by using key (K). compression on Steganography”, 19 Security Conference, 1996

[6]. C.E., Shannon, (1949), Communication theory of secrecy systems, Bell System Technical Journal, 28, 656-715

[7]. Dunbar, B., “Steganographic techniques and their use in an Open- Systems environment”, SANS Institute, January 2002

[8]. Al-Husainy, M. A., (2009) “Image Steganography by Mapping Pixels to Letters,” Journal of Computer Science, 5 (1): 33-38

[9]. Deshpande Neeta, Kamalapur Snehal, Daisy Jacobs “Implementation of LSB Steganography and Its Evaluation for Various Bits” Digital Information Management, 2006 1st International conference.pp 173-178,2007.

[10]. Diffie, W. and, Hellman, M. E., (1976) “New Directions in Cryptography,” IEEE Transactions on Information Theory, IT-22, pp. 644-654