Volume 2, No. 4, April 2011
Journal of Global Research in Computer Science
RESEARCH PAPER
Available Online at www.jgrcs.info
ISSN-2229-371X

A CHALLENGE IN HIDING ENCRYPTED MESSAGE IN LSB AND LSB+1 BIT POSITIONS IN VARIOUS Cover Files

Joyshree Nath[1], Sankar Das[2], Shalabh Agarwal[3] and Asoke Nath[4]

[1]A.K.Chaudhuri School of IT,Raja Bazar Science College,Calcutta University
[2,3,4]Department of Computer Science'St. Xavier's College (Autonomous),Kolkata

E-mail: joyshreenath@gmail.com[1], dassankar16@yahoo.co.in, shalabh@sxccal.edu and asokejoy@gmail.com2

Abstract: The present work basically shows us how one can hide information in encrypted form to any cover file such as .exe files, Microsoft office files, .dbf files, image files, audio files and video files. However, the size of the hidden message must be very small in comparison to cover file which is an executable file. So far no one has tried to hide information inside any executable file. To make the system fully secured we first encrypt the secret message using MSA algorithm (Nath et al.(1)) and then we hide the encrypted message inside the cover file. introduced a new method for hiding any encrypted secret message inside a cover file. For encrypting secret message we have used new algorithm proposed by Nath et al(1). For hiding secret message we have changed both LSB and LSB+1 bits of each byte of the cover file. A generalized method was proposed by Nath etal(2) where they embed the secret message without going for any encryption.. The MSA(1) algorithm introduced a new randomization method for generating the randomized key matrix to encrypt plain text file and to decrypt cipher text file. The MSA (1) method also incorporates the multiple encryption and decryption process. To initiate the MSA algorithm the user has to enter a text_key, which can be of 16 characters long. This text_key is used to calculate the randomization number and the encryption number from the given text_key. The size of the encryption key matrix is 16x16 and the total number of matrices can be formed from 16 x 16 is 256! which is quite large and the MSA algorithm ensures that any of the pattern may be used for encryption as well as decryption process. To hide encrypted secret message in the cover file we have inserted the 8 bits of each character of encrypted message file in 4 consecutive bytes of the cover file such that only LSB and LSB+1 bits are changed depending on the bit pattern of the encrypted secret message. To make system further secured one has to enter a password before the actual steganography process starts. We propose that our new method could be most appropriate for hiding any file in any non-standard cover file such as executable file, compiler, MS-Office files, Data Base files such as .DBF, text editor such as notepad plus the standard cover files such as image, audio, video files etc. The size of the secret message be very small in comparison to the executable cover file. The present method may be implemented in mobile network, Bank data transactions in government sectors, in police department.

INTRODUCTION:

Nath et al (2) already proposed various methods for hiding secret data inside image, audio and video files. In the present work we propose two (2) methods: (i) We encrypt the secret message(SM) using MSA proposed by Nath et al.(1) and (ii) We insert the encrypted secret message inside the cover file(CF) by changing the least significant bit(LSB) and LSB+1 bits. We propose here to modify both LSB and LSB+1 bit to ensure that we can hide more secret message in a cover file. This method could be very useful in embedding data in some non-standard cover files such as compiler, OS, .exe file, database file etc. The present work gives warning to the computer professionals that the present method can be used to hide any secret message inside any cover file except pure ASCII file. Now we will explain the steganography method, which we have used here:

(i) Changing Least Significant Bit(LSB) and LSB +1 bit of the cover file: To hide one(1) byte secret message we choose 4 consecutive bytes of the cover file and then insert the bits in LSB and LSB+1 positions. To embed 1 byte information we need 4 bytes of the cover file. Let us consider a cover file which contains 4 bytes: 00101111 00011101 11011101 10100110. Suppose we want to embed a number 245 in the above bit pattern. The binary representation of 245 is 11110101. Now we try to embed this bit pattern in above 4 bytes. To embed 11110101 we will choose LSB and LSB+1 bits of the above 4 bytes of the cover file. Table 1 shows how the bits are inserted.

Before Replacement	After Replacement	Bit inserted	Remarks
00101111	00101**11**	1,1	No change in bit pattern
00011101	00011**11**	1,1	Change in bit pattern(i)
11011101	11011**01**	0,1	No change in bit pattern(ii)

Table 1 Changing LSB and LSB+1 bits by the bits of secret message file.

Here we can see that out of 4 bytes only 2 bytes get changed. Since we are changing the LSB and LSB+1 bits hence we are either changing the corresponding character in forward direction or in backward direction by only three units (max.) As our eye is not very sensitive so therefore after embedding a secret message in a cover file our eye may not be able to find the difference between the original message and the message after inserting some secret text or message on

to it. To embed secret message we have to first skip 5000 bytes from the last byte of the cover file. After that according to size of the secret message (say n bytes) we skip 4*n bytes and then we start to insert the bits of the secret file into the cover file. Under no circumstances the size of the cover file should not be less the 5*sizeof(secret message) then our method will fail. For extracting embedded file from the cover file we have to enter the password for verification purpose. If password is correct then the program will read the file size from the cover file. Once we get the file size we follow simply the reverse process of embedding a file in the cover file. We read LSB and LSB+1 bits of each byte and accumulate 8 bits to form a character and we immediately write that character on to a file. In the present work we primarily try to embed any secret message in some executable file, compiler, MS_Office files , database file such as .DBF, editor program such as notepad.exe etc. Normally the people try to hide message inside some standard image file (.BMP file) but in the present work we extended the steganografic method to various non standard cover files. We try to show here that time is coming when the any type of file can be used as cover file for hiding some secret information. Suppose a word document contains some multiple choice type of questions and the answers to all questions are embedded in the same question paper. This may be dangerous but the time is coming in future and we have to prepare ourselves to face it.

(ii) Meheboob, Saima and Asoke(MSA)
Symmetric key Cryptographic method:

Nath et al.(1) proposed a symmetric key method where they have used a random key generator for generating the initial key and that key is used for encrypting the given source file. MSA method is basically a substitution method where we take 2 characters from any input file and then search the corresponding characters from the random key matrix and store the encrypted data in another file. In our work we have the provision for encrypting message multiple times.

The key matrix contains all possible characters(ASCII code 0 to 255) in a random order. The pattern of the key matrix will depend on text_key entered by the user. Nath et al. proposed algorithm to obtain randomization number, encryption number and the initial shift parameter from the text_key.

n3= Σall digits in sum=1+7+9+3+6=**26**

Now we apply the following randomization methods one after another in a serial manner:
Step-1: Function cycling()
Step-2: Function upshift()
Step-3: Function downshift()
Step-4:Function leftshift()
Step-5:Function rightshift()
Step-6:Function random()
Step-7:Function random_diagonal_right()
Step-8:Function random_diagonal_left()
For detail randomization methods we refer to our previous work(1).

After finishing above shifting process we perform
(i)column randomization

1. Random Key Generation and MSA Encryption Algorithm:

Before we embed the secret message in a cover file we first encrypt the secret message using MSA algorithm(1). Now we will describe the MSA algorithm in brief. To create Random key Matrix of size(16x16) we have to take any text_key. The size of text_key must be less than or equal to 16 characters long. These 16 characters can be any of the 256 characters (ASCII code 0 to 255). The relative position and the character itself are very important to calculate the randomization number, the encryption number and the relative shift of characters in the starting key matrix. Let us take an example how to calculate randomization number, the encryption number and relative shift from a given text_key. Suppose text_key=AB. The data shown in table 3 is used for calculating the place value and the power of characters of the incoming key:

Table-3 Key length and base

Length of key(n)	1	2	3	4	5	6	7	8
Base value(b)	17	16	15	14	13	12	11	10
Length of key(n)	9	10	11	12	13	14	15	16
Base value(b)	9	8	7	6	5	4	3	2

Step-1: $\text{Sum}=\sum_{m=1} \text{ASCII Code} * b^m$ ----(1)

Example-1: Choose a text_key="**AB**"

First we calculate the sum for key="**AB**" using equation (1)

$\text{Sum}=65*16^1 + 66 * 16^2 = 17936$

Now we have to calculate 3 parameters from this sum (i) Randomization number (n1), (ii) Encryption number(n2) and (iii)Relative shift(n3) using the following method:

(i) Randomization number (n1):
num1=1*1+7*2+9*3+3*4+6*5=84
n1=sum mod num1=17936 mod 84=**44 .** Note: if n1=0 then n1=num1 and n1<=128

(ii) Encryption number (n2):
num2=6*1+3*2+9*3+7*4+1*5=72
n2=sum mod num2 =17936 mod 72 =**8** Note: if n2=0 then n2=num2 and n2<=64

(iii) Relative shift (n3):

(ii)row randomization and
(iii)diagonal rotation and
(iv)reverse diagonal rotation.

Each operation will continue for n3 number of times.

Now we apply encryption process on any text file. Our encryption process is as follows:

We choose a 4X4 simple key matrix as shown in table 4.

Table-4

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

Case-I : Suppose we want to encrypt **FF** then it will taken as **GG** which is just one character after F in the same row.

CHANGING LSB BITS OF COVER FILE USING ENCRYPTED SECRET MESSAGE FILE

In the present work we have made an exhaustive study on embedding (i) text, (ii)sound, (iii)image in different cover files such as image file, sound file, word document file, .PDF file. The size of the cover file must be at least 10-times more than secret message file which is to be embedded within the cover file. The last 500 bytes of the cover file we reserved for storing the password and the size of the secret message file. After that we subtract n*(size of the secret message file) from the size of the cover file. Here n=8 depending on how many bytes we have used to embed one byte of the secret message file in the cover file. For strong password we have used a simple algorithm as follows: We take XOR operation with each byte of the password with 255 and insert it into the cover file. To retrieve the password we read the byte from the cover file and apply XOR operation with 255 to get back original password. To embed any any secret message we have to enter the password and to extract message we have to enter the same password. The size of the secret message file we convert into 32 bits binary and then convert it into 4 characters and write onto cover file. When we want to extract encrypted secret message from a cover file then we first extract the file size from the cover file and extract the same amount of bytes from cover file. Now we will describe the algorithms which we have used in our present study:

We read one byte at a time from the encrypted secret message file(ESMF) and then we extract 8 bits from that byte. After that we read 8 consecutive bytes from the cover file(CF). We check the LSB of each byte of that 8 byte chunk whether it is different from the bits of ESMF. If it different then we replace that bit by the bit we obtain from the ESMF. Our program also counts how many bits we change and how many bytes we change and then we also calculate the percentage of bits changed and percentage of bytes changed in the CF. Now we will demonstrate in a simple case.:

Case –II : Suppose we want to encrypt **FK** where **F** and **K** appears in two different rows and two different columns. **FK** will be encrypted to **KH (FK→GJ→HK→KH).**

Case-III: Suppose we want to encrypt **EF** where **EF** occurs in the same row. Here **EF** will be converted to **HG.**

Suppose we want to embed "A" in the cover text "BBCDEFGH". Now we will show how this cover text will be modified after we insert "A" within it.

TABLE -7
CHANGING LSB and LSB+1

Original Text	Bit string	Bit to be inserted in LSB	Changed Bit string	Changed Text
B	01000010	0	0100001**0**	B
B	01000010	1	0100001**1**	C
C	01000011	0	0100001**0**	B
D	01000100	0	0100010**0**	D
E	01000101	0	0100010**0**	D
F	01000110	0	0100011**0**	F
G	01000111	0	0100011**0**	F
H	01001000	1	0100100**1**	I

Here we can see that to embed "A" we modify 5 bits out of 64 bits. After embedding "A" in cover text "BBCDEFGH" the cover text converts to "BCBDDFFI". We can see that the change in cover text is prominent as we are trying to embed text within text, which is actually not possible using LSB method. But when we do it in some image or audio file then it will not be so prominent.

To extract byte from the cover file we follow the reverse process, which we apply in case of encoding the message. We simply extract serially one by one from the cover file and then we club 8 bits and convert it to a character and then we write it to another file. But this extracted file is now in encrypted form and hence we apply decryption process which will be the reverse of encryption process to get back original secret message file.

RESULTS AND DISCUSSION

Case-1: Cover File type=.jpg Secret File type=.jpg

Fig_1:Cover file name: sxcn.jpg Size=1155378 Bytes
Fig_2:Secret message File: joy1.jpg Size=1870 Bytes
Fig_3: Embedded Cover file name: sxcn.jpg Size=1155378 Bytes
(Secret message encrypted before embedding)

Case-2: Cover File type=.BMP secret message file =.doc

Fig_4: Cover File name: tvshow.bmp
Size=688856 Bytes.

Fig_5: Embedded Cover File name : tvshow.bmp
Size=688856 Bytes
In this file an encrypted word file
Xxfile2.doc (size=19456B) is embedded

Case-3: Cover File type=. BMP

secret message file =.jpg

+

=

Fig_6: Cover file name = tvshow1.bmp (size=688856B)

Fig_7: Secret message file= tuktuk1.jpg(size=50880B) (The secret message file was Encrypted while embedding)

Fig_8: Embedded cover file name=tvshow1.bmp (size=688856B)

Case-4: Cover File type=..AVI(Movie File)

secret message file =.jpg

+

=

Fig_9: Cover File Name= (Size=76800B)

Fig_10: Secret message File name = tuktuk_bw.jpg (size=1870B)

Fig_11:Embedded Cover File Name=rhionos.avi. Name=rhinos.avi (size=76800B) (The encrypted secret message file is embedded)

CONCLUSION

In the present work we try to embed some secret message inside any cover file in encrypted form so that no one will be able to extract actual secret message. Here we use the standard steganographic method i.e. changing LSB bits of the cover file. Our encryption method can use maximum encryption number=64 and maximum randomization number=128. The key matrix may be generated in 256! Ways. So in principle it will be difficult for any one to decrypt the encrypted message without knowing the exact key matrix. Our method is essentially stream cipher method and it may take huge amount of time if the files size is large and the encryption number is also large. The merit of this method is that if we change the key_text little bit then the whole encryption and decryption process will change. This method may most suitable for water marking. The steganography method may be further secured if we compress the secret message first and then encrypt it and then finally embed inside the cover file.

ACKNOWLEDGEMENT

A sincerely expresses his gratitude to Department of Computer Science for providing necessary help and assistance. AN is also extremely grateful to University Grants Commission for providing fund for continuing minor research project on Data encryption using symmetric key and public key crypto system. JN is grateful to A.K. Chaudhury School of I.T. for giving inspiration for research work.

REFERENCES

[1] Symmetric key cryptography using random key generator, A.Nath, S.Ghosh, M.A.Mallik, Proceedings of International conference on SAM-2010 held at Las Vegas(USA) 12-15 July,2010, Vol-2,P-239-244

[2] Data Hiding and Retrieval, A.Nath, S.Das, A.Chakrabarti, Proceedings of IEEE International conference on Computer Intelligence and

Computer Network held at Bhopal from 26-28 Nov, 2010.

[3] Advanced steganographic approach for hiding encrypted secret message in LSB, LSB+1, LSB+2 and LSB+3 bits in non standard cover files, Joyshree Nath, Sankar Das, Shalabh Agarwal and Asoke Nath, International Journal of Computer Applications (1975-8887), Vol 14-No.7, Page-31-35(2011)

[4] Cryptography and Network, William Stallings, Prectice Hall of India

[5] Modified Version of Playfair Cipher using Linear Feedback Shift Register, P. Murali and Gandhidoss Senthilkumar, UCSNS International journal of Computer Science and Network Security, Vol-8 No.12, Dec 2008.

[6] Jpeg20000 Standard for Image Compression Concepts algorithms and VLSI Architectures by Tinku Acharya and Ping-Sing Tsai, Wiley Interscience.

[7] Steganography and Seganalysis by Moerland, T , Leiden Institute of Advanced Computing Science.

[8] SSB-4 System of Steganography using bit 4 by J.M.Rodrigues Et. Al.

[9] An Overview of Image Steganography by T.Morkel, J.H.P. Eloff and M.S.Oliver.

