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
A.K.Chaudhuri School of IT, Calcutta University
India

Sankar Das
Dept. of Comp. Sc., St. Xavier's College (Autonomous), Kolkata, India

Shalabh Agarwal
Dept. of Comp. Sc., St. Xavier's College (Autonomous), Kolkata, India

Asoke Nath
Dept. of Comp. Sc., St. Xavier's College (Autonomous), Kolkata, India

ABSTRACT

In digital steganography normally image, audio or video files are the standard cover files or the host files for embedding secret message such as text, image, audio or video. Nath et al.(2) explored the standard method for hiding secret message inside standard cover files such as image, audio or video files. In the present work we have shown how we can hide secret message in encrypted form in some non standard cover files such as .exe, .com, .pdf, .doc, .xls, .mdb, .ppt files. However, the size of the secret message must be very small in comparison to cover file. The secret message is converted to encrypted form using MSA algorithm(2) and then we hide the encrypted message inside the non standard cover file. To hide encrypted secret message we insert the 8 bits in 2 consecutive bytes of cover file in LSB, LSB+1, LSB+2 and LSB+3 positions. This method could be very effective to hide some information in some executable file. To make the entire process secured we have introduced the password when we hide message and while encrypting the secret message we have to input some text_key. While hiding secret message in cover file we embed 1 byte information in two consecutive bytes of the cover file. There is a risk factor that it may damage the cover file in such a way that the embedded cover file may not behave in proper way as it was behaving before insertion of secret message. However, there is one advantage that we can embed more data in a cover file. We propose that our new method could be most appropriate for hiding any file in any non-standard cover file such as executable file such as .EXE or .COM file, compiler, MS-Office files, Data Base files such as .MDB, .PDF file. Our method will now give open challenge to all user that it is possible to hide any small secret message inside in any non standard cover files. The present work shows that we can hide information in almost all files except pure text or ASCII file. The only restriction is the size of secret message should be extremely small in comparison to cover file. The present method may be implemented in digital water marking in any legal electronic documents, Bank data transactions, in government sectors, in defense, in schools and colleges.

General Terms

Steganography

Keywords

Steganography, Cryptography, Data encryption and decryption.

1. INTRODUCTION

The present work shows how to embed secret message inside some non standard cover files such as executable file, Microsoft office files, access database file etc. The secret message(SM) is encrypted using MSA(Meheboob,Saima and Asoke) algorithm proposed by Nath et al.(1). The encrypted secret message substituted inside the cover file(CF) by changing the LSB, LSB+1, LSB+2 and LSB+3 bits of the cover file. Nath et al(2) already proposed this steganographic method for embedding SM into CF but there the SF was inserted as it is in the CF and hence the security of steganography was not very high. The present work basically used the same method which was proposed earlier but in a more secured manner. Without knowing the encryption algorithm the decrypted message can not be restored to original secret message. The standard steganographic cover files are now well known to everyone. But in the present work we try to embed almost any type of file inside any type of non standard cover file(CF). Now we will first describe our steganography method for embedding any type of SM inside any type of CF and then we will describe the MSA encryption method which we have used to encrypt the secret message and to decrypt the extracted data from the embedded cover file.

(i) Insertion of bits in LSB, LSB+1, LSB+2 and LSB+3 bit position in a cover file:

Here we substitute the bits of the SF in to LSB, LSB+1, LSB+2 and LSB+3 bit positions of each byte of the cover file. The present method will take less space in the cover file. Let us consider 2 bytes in a cover file : 00101111 00011101. Suppose we want to embed a number 236 in the above bit pattern. The binary representation of 236 is 11101100. Now we want to insert this bit pattern in above 2 bytes. To embed 11101100 we will choose LSB, LSB+1, LSB+2 and LSB+3 bits of the above 2 bytes of the cover file. Table 1 shows how the bits are inserted.

Now we want to show what happens to cover file text after we embed 11101100 in LSB, LSB+1, LSB+2 and LSB+3 bit position.

Table 1 Changing LSB, LSB+1,LSB+2 and LSB+3 bits of CF

Before Replacement	After Replacement	Bits inserted	Remarks
00101111	0010**1100**	1,1,0,0	Change

			in bit pattern
00011101	000111**11**	1,1,1,0	Change in bit pattern

Here we can see that 2 bytes get changed. However, this change may not be prominent in case of audio, image or video files as human eye or ear is not very sensitive so therefore after embedding a secret message in a cover file our eye or ear may not be able to find the difference between the original file and the file after inserting some secret text or message on to it. To embed secret message we 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 2*n bytes. After that we start to insert the bits of the secret file into the cover file. However, the size of the cover file should be large in comparison to the secret message.
To extract SF from CF we have to enter the password while embedding a secret message file. Once we get the file size we follow simply the reverse process of embedding a file in the cover file. We read LSB, LSB+1, LSB+2 and LSB+3 bits from each byte and accumulate 8 bits to form a character and then we immediately write that character on to an output file.

In table 2 we have shown a comparative study of different cover files as well as secret message files.

Table 2 : Comparison of Cover File and Secret message File

Sl. No	Cover file type	Secret file type
1	.EXE	Any small file
2.	.DOC	Any small file
3.	.XLS	Any small file
4.	.PPT	Any small file
5.	.MDB	Any small File
6.	.PDF	Any small file
7.	CMD.EXE	Any small file
8.	.COM	Any small file

In table 2 we have shown mainly the non standard cover files which normally the people are not using as the host/cover file. However, our method is not applicable for hiding data in some ASCII file.

(ii) Meheboob, Saima and Asoke (MSA) Symmetric key Cryptographic method:
MSA cryptography based on symmetric key cryptography which means that same key is used encryption as well as decryption purpose. So therefore the key should not be public. In public key cryptosystem such as in RSA we use 2 keys one for encryption and one for decryption. The problem of Public key cryptosystem is that we have to do massive computation for encrypting any plain text. Some times these methods may not be suitable such as in sensor networks where the computation time should be minimum. Nath et al.(1) proposed an algorithm called MSA 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 2 characters read from input file and then search the corresponding characters in the random key matrix and store the corresponding encrypted data in another file. MSA algorithm provides the provision for multiple encryption and multiple decryption. 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.(1) proposed algorithm to obtain randomization number, encryption number and the shift parameter from the initial text_key. We have given a exhaustive trial run on text_key and we found that it is very difficult to match the three above parameters for 2 different Text_key which means if some one wants to break our encryption method then he/she has to know the exact pattern of the text_key otherwise it will not be possible to obtain two sets of identical parameters from two different text_key. For pure text file we can apply brute force method to decrypt small text but for any other file such any binary file we can not apply any brute force method and it does not work.

2. RANDOM KEY GENERATION AND MSA ENCRYPTION ALGORITHM:

To create Random key Matrix of size(16x16) we have to enter 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 is very important in our method to calculate the randomization number , the encryption number and the relative shift of characters in the starting key matrix. Suppose text_key entered=AB. We choose table-3 for calculating the place value and the power of characters of the incoming key:

Table 3: Length of text_key and base value

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

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

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

We first create 16 X 16 (total 256 characters) key matrix which contains all characters (ASCII code 0-255) and then we give a relative shift(n3) to all elements in the matrix.

After that 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 Nath et al(1).

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

Table-4 :Key matrix(4X4)

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.

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.**

In the present work the last 5000 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=4 depending on how many bytes we have used to embed one byte of the secret message file in the cover file. To embed 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 2 consecutive bytes from the cover file(CF). We check the LSB and LSB+1 bit of each byte of that 2 byte chunk whether it is different from the bits of ESMF. If it is 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 :
Suppose we want to embed "A" in the cover text "AD". Now we will show how this cover text will be modified after we insert "A" within it. The entire process is shown in Table-5.

Table -5 Changing LSB, LSB+1, LSB+2 and LSB+3 bits

Original Text	Bit string	Bit to be inserted in LSB & LSB+1	Changed Bit string	Changed Text
A	01000001	0,1,0,0	01000100	D
D	01000100	0,0,0,1	01000001	A

Here we can see that to embed "A" we modify the cover file AD to DA. We can see that the change in cover text is prominent. AD is modified to DA. For text file this change is noticeable 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.

3. RESULTS AND DISCUSSION

Case-1: Cover File type=.jpg, secret File type=.jpg -- [sxc1.jpg (Fig-1) + joy3.jpg (Fig-2)=sxc1.jpg (Fig-3)]

Fig-1:Cover File name = sxc1.jpg (size=1155378B)

+

Fig-2: Secret message file name = joy3.jpg (size=634B)

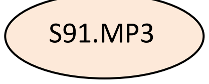
=

Fig-3: Embedded cover file name = sxc1.jpg (size=11553778)

Joy3.jpg is hidden inside this file.

Case-2: Cover File type=.mp3, secret message file =.jpg -- [S91.mp3 (Fig-4) + joy1.jpg (Fig-5) = S91.mp3 (Fig-4)]

Fig-4: Cover file name=S91.mp3 size=110771B)



+

Fig-5:Secret message file name=joy1.jpg (size=1870B)

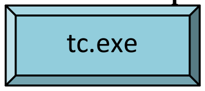
=

Fig-6: Embedded cover file name = s91.mp3 (size=110771B)

joy1.jpg is embedded inside this file

Case-3: Cover file=.exe file, secret file=.c file [tc.exe (Fig-7) + wordext1.c(Fig-8)=tc.exe(Fig-9)]

Fig-7:Cover file name=tc..exe (size=290249B).
It is Turbo-C compiler



+

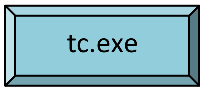
Fig-8: secret file name=wordext1.c (size=978B)

```
/*Write a program to extract words from any file*/
        #include<stdio.h>
        main()
        {
        FILE *fp1,*fp2;
        char file1[50],file2[50],ch;
        long int n,nw,flag,lc;
        clrscr();
        printf("\nEnter File name form where words to be
extracted=");
        gets(file1);
        printf("Enter Word file name=");
        gets(file2);
        fp1=fopen(file1,"rb");
        fp2=fopen(file2,"wb+");
        n=nw=lc=0;
        flag=1;
        while(fscanf(fp1,"%c",&ch)>0)
        {
        n++;
                if((ch>='A' && ch<='Z') || (ch>='a' &&
ch<='z') || ch=='-')
                {if(flag==1)
                        {
                        nw++;
                        flag=0;
                        }
                fprintf(fp2,"%c",ch);
                }
                else if(flag==0)
                {
                fprintf(fp2,"\n");
                flag=1;
                }
        }
        fclose(fp1);
        rewind(fp2);
        while(fscanf(fp2,"%c",&ch)>0)
        {printf("%c",ch);
                if(ch=='\n')
                {lc++;
                        if((lc%20)==0)
                        {
                        printf("\nPress any key to
continue-->");
                        getch();
                        clrscr();
                        }
                }

        }
        printf("\nSize of <%s>=%ld\n",file1,n);
        printf("Number of words extracted=%ld\n",nw);
        getch();
        }
```

=

Fig-9: embedded cover file name = tc.exe (size=290249B)



wordext1.c is embedded inside this turbo-c compiler.

The compiler works perfectly ok even after we embed this c-program file.

Case-4: Cover File=.doc file, secret message=.c file. [mydoc2.doc (Fig-10) + xxfile.c (Fig-11) = mydoc2.doc]

Fig-10: Coverfile=mydoc2.doc (size=22528B)

To,
Dr. Amlan Chakrabarti
A.K.Chaudhuri School of I.T.
Raja Bazar Science College

Date: 05/07/2010

Dear Dr. Chakrabarti,

How are you? I hope you will be now very busy with your work. I was trying to go and meet you in Science College but because of tremendous work pressure I am unable to meet you. I am leaving Kolkata on 10-th July at 8:00PM. My lecture on 13-th July at 12:20PM-12:40PM(Las Vegas Standard Time). On 14-th July I have to chair one full 2hrs session. My return ticket on 16-th of July and I will return back to Kolkata on 18-th July at 10:20PM. If everything goes right direction then I will again join college on 19-th July. As soon as I return back I will contact you at Science College. I have to finalize the workshop on Matlab.
Amlan I have some queries. If possible you answer it :

(i) When the result of 6-th of MCA will be declared?
(ii) When Joyshree and the students of MCA final year can join M.tech in your Dept as I will be out from 10/07/2010 to 18/07/2010.
(iii) Any idea about the amount of fees to be deposited at time of admission in M.Tech?

I am now totally tied up with our second work on Crytography. It is almost at the finishing stage. I want to send it before I leave from Kolkata. I to also prepare the slides of my lecture to be delivered at Las Vegas.

With kind regards.

Yours sincerely

Asoke Nath
Nataraj Housing
381 & 382A M.G.Road
Kolkata-700 082
Phone: 24020909

+

Fig-11: secret message file=xxfile1.c (size=460B)

```
/*Write a program to create a file which contains all
characters whose ASCII  Code 0-255 */
    #include<stdio.h>
    main()
    {
     int i;
    char file1[50];
    FILE *fp1;
    clrscr();
    printf("\n\n\nEnter your File Name :");
    gets(file1);
    fp1=fopen(file1,"wb");
    for(i=0;i<=255;i++)
    fprintf(fp1,"%c",i);
    for(i=0;i<=255;i++)
    fprintf(fp1,"%c",i);
        fcloseall();
        printf("\n\nFile is created. Press any key to finish the
program--->");
        getch();
```

=

mydoc2.doc (embedded cover file) size=22528B

4. CONCLUSION

In the present work we have tried to hide any secret message inside any cover file such as executable file, Microsoft Office file, Database file and common cover file type such as image, audio, video etc. The security of our method is also very high as the message which we embed that is in encrypted form. It means even if someone extracts the message from the cover file but will not be able to able to decipher it. Brute force method will not help the intruder to decipher the encrypted message. Our method may be used in Bank, Defense and Government sectors where the Confidentiality of data is very important. In Mobile communication also this method may be implemented. The time coming when it will be possible to hide any secret message within pure ASCII file.

5. ACKNOWLEDGEMENT

AN,SD and SA sincerely express their 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 doing research work

6. REFERENCES

[1] Symmetric key cryptography using random key generator, A.Nath, S.Ghosh, .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 Int.ational conference on Computer Intelligence and Computer Network held at Bhopal from 26-28 Nov, 2010.

[3] William Stallings, "Cryptography and Network Security: Principles and Practices." 3rd edition, 2003.

[4] Steganography at http://en.wikipedia.org/wiki/ Steganography

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

[6] An Overview of Steganography by Shawn D. Dickman

[7] Digital Steganography: Hiding data within Data, IEEE Internet Computing, May/June 2001, pp. 75-80.

[8] Cyber Warfare: steganography vs. Steganalysis, Communications of the ACM, Vol 47, Issue 10, Oct 2004, PP. 76-82.