

Modified Least Significant Bit (MLSB)

Mazen Abu Zaher (Corresponding author)

Faculty of Engineering Technology, Albalqa' Applied University, Amman, Jordan

P.O.Box:7266 (131/16) Zarqa, Zarqa, Jordan

E-mail: mazen.abuzaher@fet.edu.jo

Abstract

Now a day steganography plays a critical role in Information Security. Our work introduces modifications to existing steganography algorithm known as LSB "Least Significant Bit". The main idea of our work is to increase amount of data that can be hiding using LSB. In addition our algorithm encrypt data before hide it, which provide another level of protection over traditional LSB technique.

Keywords: Steganography, Least Significant Bit (LSB), ASCII

1. Introduction

Steganography is the science of writing hidden messages is such a way that no one except sender and intended recipient can realize there is a hidden message. Compared with cryptography we can say that, steganography's in security work as supplement to cryptography, not replace it. So to add another layer of protection we can encrypt the hidden message (Johnson & Jajodia, 1998). A survey of current steganography methods can be found in (Cheddad, Condell, Curran, & Kevitt, 2010) and (Katzenbeisser, & Petitcolas 2000).

The most widely used technique to hide data is the usage of LSB (Cox, Miller, Bloom, Fridrich, & Kalker, 2008). Although there are several disadvantages to this approach, the relative easiness to implement it, makes it a popular method. 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 transformation of a lossy compression algorithm. When using 24-bit color image, the least significant bit of each of the Red, Green, Blue color component can be used, so a total of 3 bits can be stored in each pixel.

In literature there are many algorithms to enhance LSB. The main motivation for those algorithms is to make LSB less detectable and more secure. Roque and Minguet (2009) introduce an algorithm to minimize color changes in the cover image using LSB. On the other hand Cvejic and Seppanen (2004) try to reduce embedding distortion of the host audio using LSB.

In this paper we introduce modifications over traditional LSB method to increase the amount of data that can be hiding in the cover image. In addition, to increase data protection our algorithm have a built in encryption technique. As LSB (Bandyopadhyay, Bhattacharyya, Ganguly, Mukherjeel, & Das, 2008) the output image of our algorithm will look identical to the cover image.

2. The Algorithm

To describe this work we will use a 24-bits color image as cover image. The hiding operation will be done on the basis of bit wise message hiding concept. Where the message's characters will be treated with their 8-bits ASCII codes (Kochhar, 2008), then the codes will be converted to 5-bits code using our algorithm, and then the 5-bits code will be hidden in the image using LSB method. In the cover image, each pixel is treated accordingly with its Red, Green, Blue values. This will generate a very small difference, for human eye.

In our algorithm we use just 5 bits to represent characters and numbers. To perform that we study characters ASCII representation to eliminate redundant bits.

According to our study if we look for ASCII representation we can note that, small letter character representation (hexadecimal) in the range between 61h to 7Ah, and capital letter character in the range between 41h to 5Ah. Finding relationship between them we can note that:

For small letter the last 4 bits can be just either 6 or 7. In case of 6 the first 4 bits of characters is incremented from 1h to Fh to represents characters from 'a' to 'o' as shown in Table 1. Then the last four bits (with value = 6) are incremented to be 7 and the first 4 bits again incremented from 0h to Ah to represents characters from 'p' to 'z' as shown in Table 2.

For capital letter the last 4 bits can be just either 4 or 5. In case of 4 the first 4 bits of characters is incremented

from 1h to Fh to represents characters from 'A' to 'O'. Then the last four bits (with value = 4) are incremented to be 5 and the first 4 bits again incremented from 1h to Ah to represents characters from 'P' to 'Z' (see Table 3).

We can note that there is a repeated sequence between capital and small letters representation. In the case of small letter if we take the binary representation of last four bits, it will be "0110" or "0111". For capital letter the last four bits will be "0100" or "0101". As we can see the last three bits are identical so we can discard them.

Accordingly we decide to use 5 bits to represent character where the first 4 bits representation will be the same as the first 4 bits representation in ASCII.

The fifth bit can be 0 or 1. If 0 then we represent the first sequence from 01h to 0Fh, in case of small letter characters from 'a' to 'o'. Else if the fifth bit equal 1; we represent the second sequence from 0h to Ah, in case of small letter characters from 'p' to 'z'.

But how we can distinguish between small letter, capital letter, numbers, and space. To perform that we use control symbol that can define the state of next character. Our control symbols are also 5-bits.

In 5-bits representation when the fifth bit is one the first 4 bits will range from 01h to 0Ah. As we can note the range from Bh to Fh is not used. So we use the successive from Bh to Fh as control symbol as shown in Table 4.

Flowchart of our text hide algorithm shown in Figure 1.

According to Table 4 and Figure 1, our algorithm deals with five cases:

(1) Small letter:

If we want to represent small letter first we must put the first 5-bits to be 1Bh then include our converted (5-bits) small letter text.

(2) Capital letter:

If we want to use capital letter first we must put the first 5-bits to be 1Ch then include our text.

(3) Space:

In the case of space we put the 5-bits to be 1Dh instead of 20h (ASCII representation of space) since space it is not in the range of ASCII characters.

(4) Numbers:

In case of numbers ASCII represents number from 0 to 9 as 30h to 39h. if we discard the last four bits we can represent numbers using just 4 bits. In our case the fifth bit will be 0. In our process to indicate that we deal with number then first we must use the control symbol 1Eh (first 5 bits) then write the number (each digit range from 0 to 9).

(5) End of text:

Finally to define the end of the text we use the control symbol 1Fh. In some cases we can ignore this control symbol.

In our algorithm before each new text format (capital, small, or numbers) the first 5-bits must be control symbol then we include the converted text sequence.

Next is the Pseudo code of our steganography algorithm to convert from 8-bits ASCII to 5-bits and hide the text.

```

Define x, z as byte
For each character from j=0 to end of text
  x=character[j]
  If x>=30h and x<=39h then ' number
    z=1Eh
    go to L1
  Else
  If x=20h then ' space
    z=1Dh
    go to L1
  Else

```

```

x=x AND F0h
If x=70h or x=60h then      ' small letter
  z=1Bh
  go to L1
  Else
  If x=50h or x=40h then    ' capital letter
    z=1Ch
    go to L1
  Else go to L2

L1: While not end of text

  hide the first 5-bits from z in the image using LSB technique

  z= character(j) AND 1Fh    ' discard the last 3 bits
  hide the first 5-bits from z in the image using LSB technique

  If character(j+1) the same case as character(j) (small capital, or number) and not space or end
of text Then
    j=j+1
    z= character(j) AND 1Fh    ' discard the last 3 bits
    go to L1
  end if

L2:

Next j

z= 1Fh                        'end of text
hide the first 5-bits from z in the image using LSB technique

```

To extract the hidden text in any image we must perform the following steps:

- 1) Using LSB algorithm extract the first bits from each pixel (RGB) in the image, which represent 3-bits for each pixel. Then group the resulted stream as five bits.
- 2) Compare the first five bits with control symbols to define the next character state.
- 3) If the control symbol represent characters (small (1Bh) or capital (1Ch)) then for each next five bits (which are not control symbol) perform OR with 60h to convert from 5-bit to 8-bit (ASCII).
- 4) If the control symbol represent space (1Dh) then store 20h (space in ASCII)
- 5) If the control symbol represent numbers (1Eh). Then OR next five bits with 30h (numbers in ASCII).
- 6) Repeat steps from 2 to 5 till control symbol end of text (1Fh).

In the next section we will show one scenario to describe our algorithm.

3. Experiment

The presented algorithm minimizes number of bits from 8 to 5. In addition we can consider it as encryption algorithm because if any one extract the bits from image he will not understand anything until decrypt it. So our algorithm performs two operations in one scheme.

In the following scenario we use 18 character length text to show how our algorithm work.

Plain text of 18 characters:

"STEGO with 05 bits"

ASCII representation need $(18 * 8) = 144$ bits. As following:

53h,54h,45h,47h,4fh,20h,77h,69h,74h,68h,20h,30h,35h,20h, 62h,69h,74h,73h

Our algorithm representation to the same message need $(22 * 5) = 110$ bits. As following (where ***Bold-Italic*** hexadecimal represent control symbols):

Ich,13h,14h,05h,07h,0fh,***Idh,Ibh***,17h,09h,14h,08h,***Idh,Ieh***,00h,05h,***Ibh***,02h,09h,14h,13h,***Ifh***

The first five bits from each element in the previous message will be stored in the cover image using LSB technique. This text need at least image with $(110/3)$ pixels (rounded to integer number = 37 pixels).

In our experiment we use an image with 160*160 pixel resolution. The image before and after text hidden shown in Figure 2 (A and B respectively).

As we can see the difference between traditional representation and our algorithm is 34 bits. This gap will increase as the text extends in size, which gives us the opportunity to hide larger amount of data compared with traditional LSB.

To extract the hidden message we perform the extract steps from 1 to 6. Accordingly the extracted message is:

53h,54h,45h,47h,4fh,20h,77h,69h,74h,68h,20h,30h,35h,20h, 62h,69h,74h,73h

This is the same as original message.

4. Conclusion

In this paper we introduce a new steganography algorithm. Our algorithm provide enhanced scheme over traditional LSB method. The main aim of the presented work is to increase amount of data to be hidden in the cover image and to increase security by performing data encryption.

The advantage of our steganography algorithm is that it can hide larger amount of data than traditional LSB. In addition, we can consider that our algorithm perform built in encryption, because of conversion process.

The main disadvantage of our algorithm if the text flips between small and capital in each character. Then the size will increase not decrease because of control symbols. But we can say; such scenario is rare.

References

- Bandyopadhyay, S. K., Bhattacharyya, D. , Ganguly, D. , Mukherjeel, S. & Das, P. (2008). A tutorial review on steganography. International Conference on Contemporary Computing.
- Kochhar, R. (2008). ASCII table. Retrieved September 19, 2010, from University of Wisconsin - Madison Web site: <http://www.neurophys.wisc.edu/comp/docs/ascii/>.
- Cheddad, A., Condell, J., Curran, K., & Kevitt, P. Mc. (2010). Digital image steganography: survey and analyses of current methods signal processing. (pp. 727-752), Vol. 90, Issue 3.
- Cox, I. J., Miller, M. L., Bloom, J. A., Fridrich, J., Kalker, T. (2008). *Digital watermarking and steganography(2nd. ed.)*. New York: Morgan Kaufmann.
- Roque, J. J., & Minguet, J. M. (2009). SLSB: Improving the steganographic algorithm LSB. Proceedings The Ibero-American Congress on Information Security (CIBSI). (pp. 398-408).
- Cvejic, N., Seppanen T. (2004). Increasing robustness of LSB audio steganography by reduced distortion LSB coding. Proceedings ITCC 2004 International Conference on Information Technology: Coding and Computing. (pp. 533 – 537). Vol.2.
- Katzenbeisser, S., &Petitcolas, F. A. (2000). *Information hiding techniques for steganography and digital watermarking*. (pp. 43-78). London: Artech House.
- Johnson, N. F., & Jajodia, S. (1998). Steganography: seeing the unseen. *IEEE Computer*, (pp. 26-34).

Table 1. Small letter ASCII from a to o

Character	ASCII in hexadecimal	Character	ASCII in hexadecimal
a	61h	h	68h
b	62h	i	69h
c	63h	j	6Ah
d	64h	k	6Bh
e	65h	l	6Ch
f	66h	m	6Dh
g	67h	n	6Eh

Table 2. Small letter ASCII from p to z

Character	ASCII in hexadecimal
p	70h
q	71h
r	72h
s	73h
t	74h
u	75h
v	76h
w	77h
x	78h
y	79h
z	7Ah

Table 3. Capital letter ASCII from a to z

Character	ASCII in hexadecimal	Character	ASCII in hexadecimal
A	41h	N	4Eh
B	42h	O	4Fh
C	43h	P	50h
D	44h	Q	51h
E	45h	R	52h
F	46h	S	53h
G	47h	T	54h
H	48h	U	55h
I	49h	V	56h
J	4Ah	W	57h
K	4Bh	X	58h
L	4Ch	Y	59h
M	4Dh	Z	5Ah

Table 4. Control Symbols

Hex representation	operation
1Bh	Define small letter
1Ch	Define capital letter
1Dh	Define space
1Eh	Define number
1Fh	Define end of text

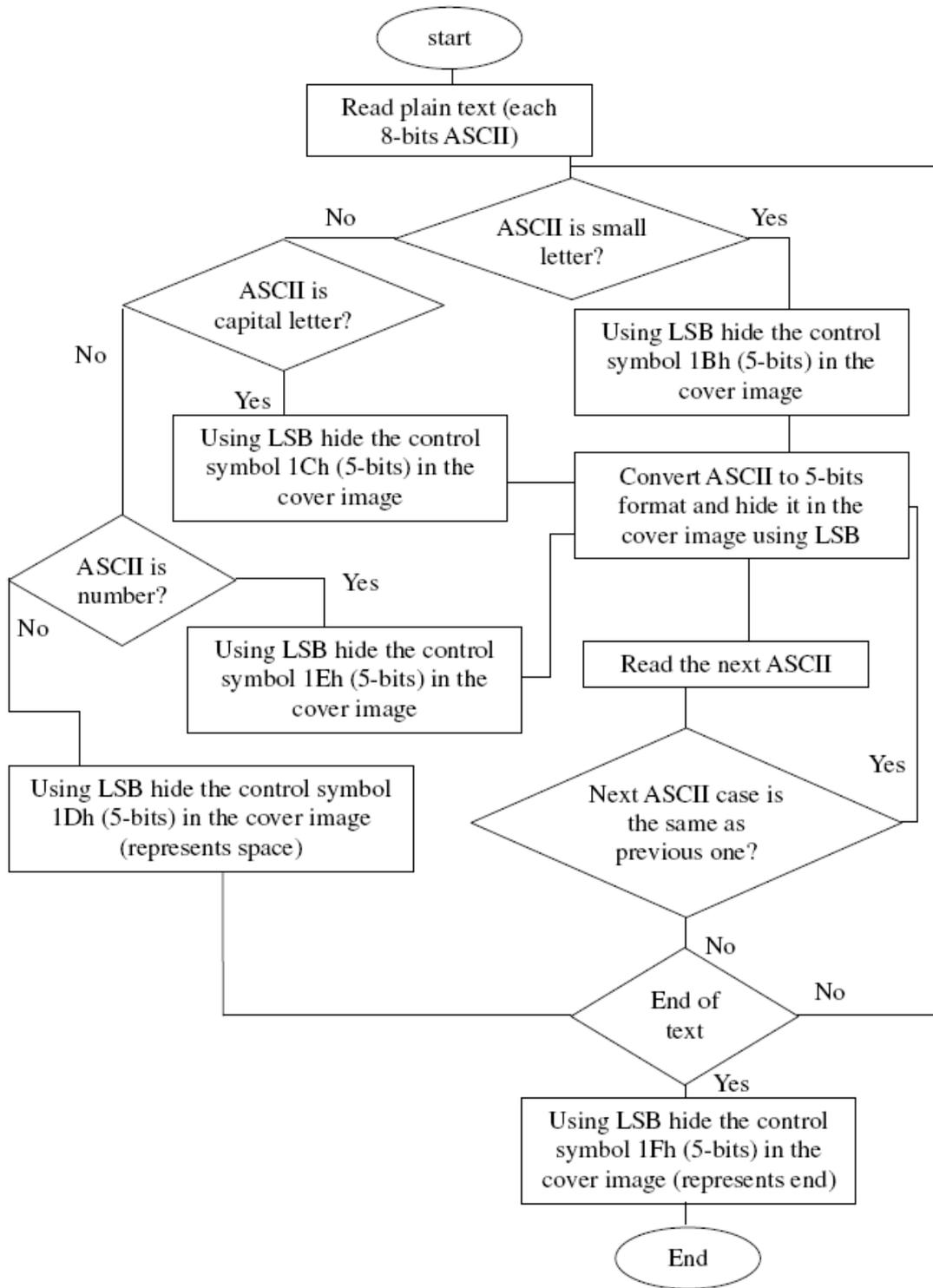


Figure 1. Hide algorithm flow chart

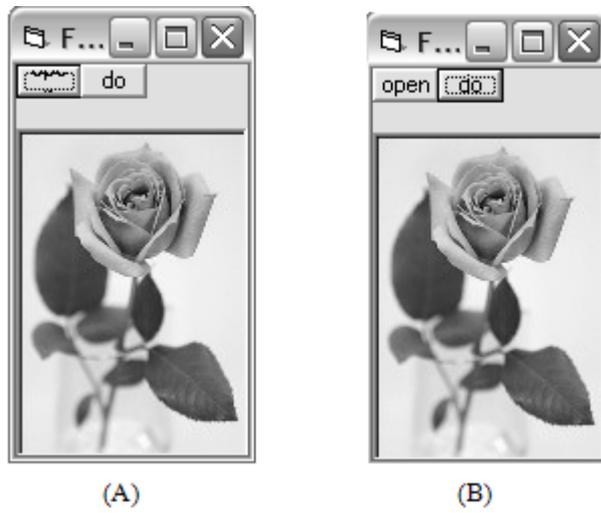


Figure 2. Algorithm implementation