Data Hiding Algorithm using Variable Block Size in Cover Image File

Prasita Mukherjee
Department of computer science
St.Xavier's college
Kolkata, India

Sourasekhar Banerjee
Department of computer science
St.Xavier's college
Kolkata, India

Asoke Nath
Department of computer science
St.Xavier's college
Kolkata, India

ABSTRACT
Data hiding inside any standard or non standard cover file is now a common area of interest across the globe. Nath et al already developed several methods for hiding any secret message inside any standard or non standard cover file. In the present work the authors have introduced a new method of hiding any kind of small secret message such as text, image, audio file inside any cover file which is mainly .bmp file. The authors divide the entire cover file into number of blocks and then inserted the bits of the secret file after a series of shift operations on them in the RGB components of pixels of cover image file. After inserting any secret message inside a cover file it was found that there is no significant change in the stego file. The shifting operation on the bits of the secret file in addition to the random generation of blocks, and processing them in an order, where a definite number of bits of the secret file to embed are reserved for each block will make the entire process fully secured. The size of the cover image should be at least 10-20 times larger than the secret message file so that the entire process should be almost unbreakable.

General Terms
Block,embed,dembed

Keywords
data hiding, audio file, stego file, secret file, pixels

1. INTRODUCTION
Data hiding is a method of hiding secret messages into a cover-media such that an unintended observer will not be aware of the existence of the hidden messages. In this paper we have selected a cover image of *.bmp format. Cover images with secret messages embedded in them are called stego images. The primary criteria of data hiding is that the quality of original cover and the stego file should be same. Normal human eye should not be able to find any difference between two image files. Nath et al already developed several methods[1-13] for hiding any kind of data inside any kind of cover file such as .bmp, .jpg, .avi, .wav, .doc etc. In the present paper the authors have introduced which is based on LSB insertion method starting from any block of initial cover file. To insert any secret message the authors first select a big cover image file (near about 1200x900 pixels). The image is then divided into several number of blocks of the same size.Depending on type of secret image the number of blocks are generated. The row-factor and column-factor are calculated. From row-factor and column-factor the number of rows and number of columns in each individual block is calculated separately. Four standard shifting operations such as left_shift(), right_shit(), up_shift() and down_shft() are applied to the bits of the secret file before embedding it inside the file. The least significant bit(LSB) of some or all the bytes inside a block of pixels of RGB component of cover image file is changed by substituting the bits of the secret message file. The first 300 rows of the cover image file is not used. Similarly the last 300 rows are used to store the key elements such as size of the secret message file, column factor, row factor and number of blocks of cover image file.To retrieve secret message from a stego file one has to read key elements from last 300 bytes of pixels of the stego file and start doing the reverse process. It means extracting bits from stego file and then converting to each 8 bits block to byte and then writing onto out file. In the present work the authors have used

Cover file- *.BMP format file

Secret message file -*.jpg,*.png,*.WAV,*.mp3, *.txt, *.doc,*.docx,*.rtf .

In the results section the authors have given the original cover file, secret message file and also the stego file. The difference in bytes of both cover file and the stego file is also shown in tabular form. The entire work was done in MATLAB.

2. METHOD USED IN THE PRESENT WORK
In the present paper the authors have made an exhaustive study on embedding (i) text, (ii) sound, (iii) image in a cover image(mainly *.BMP). The size of the cover file must be at least 10 times bigger than the secret message file which is to be embedded within the cover file. Last 300 rows of the cover image is reserved for storing key elements like size of the secret file, column and row factor of cover image, and total number of blocks of cover image. In the present approach one cannot use a secret file less than 100 bytes and greater than the ((number of rows of the cover image)-600)*number of columns.The block diagram of the present method is shown Fig-1 and Fig-2.

One has to read the actual row number and column number of the cover picture.It means row_actual= (total_row-300_row_begin-300_row_last). There will be no change in column number.The detailed description of the present method will now described in step by step manner.

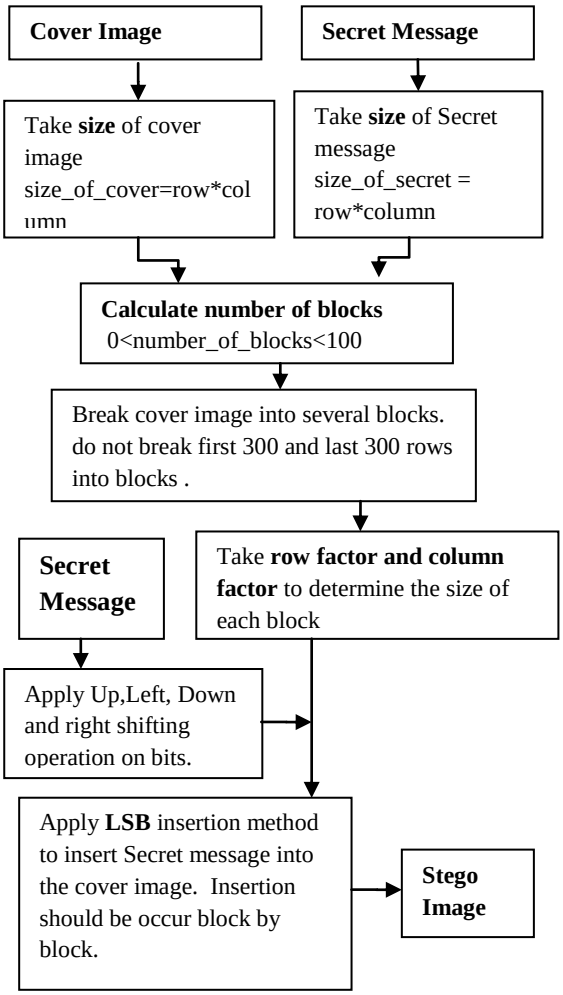


Fig 1: Block Diagram to hide secret image inside an cover image file

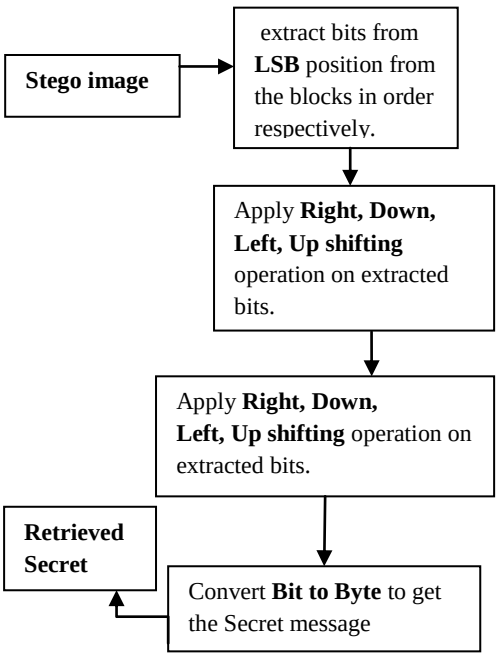


Fig 2: Block Diagram to Retrieve Secret message from Stego file.

3. BLOCK DIVISION ALGORITHM

Step 0: Start

Step 1: read cover image file (row_c,column_c)

*Step 2: fc1=row_c * column_c*

Step 3: read secret message file(row_s,colmn_s)

*Step 4: fs2=row_s * column_s.*

Step 5: is fs2 < 100000 then go to step 4 else go to step 6

Step 6: Calculate c=mod(fs2,10))

*Step 7: Calculate fs2=fs2*10+c, go to step3*

Step 8: Calculate nob=fix (fc1/fs2) ' nob=number of blocks

Step 9: if nob >100 then go to step 8 else go to step 10

Step10:Calculate rem=mod(nob,10) ' rem=remainder of nob divided by 10

Step 11: Calculate nob=fix (nob/10) +rem

Step 12 : Go to step 7

Step 13: print nob

Step 14: return nob

Step 15: end

4. EMBED ALGORITHM

4.1 . Bits of the secret file

The bits of secret message file is embedded into the cover image. The blocks are processed as shown in the block diagram. The bits are embedded in continuous pixels(i.e. no gapping between the pixels are present).

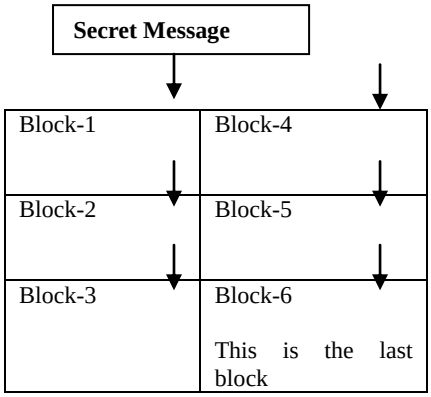


Fig 3: An example of block processing where the number of blocks are 6, with row_factor=3, column_factor=2.

(After block 3,
go to block 4)

Description of Function

stego_embed(i_file,s_file,o_file**)**

Where

i_file : The cover image file.

s_file : The secret file.

o_file : The temporary stego image file.

Step 0: Start

Step 1: y=imread(i_file) 'reading all pixels of i_file and storing in arry y()

Step 2 : [r c d]=size(y) 'where r=number of rows, c-number of columns, d=1 for B/W and 3 for color image

Step 3 : r1=301 'starting row for block-processing and embedding

Step 4 : r2=r-300

Step 5: r_act=(r2-r1)+1 'the entire accessible region, in terms of rows for embedding operation

Step 6 : [nob]=no_of_blocks(i_file,s_file)

Step 7 : [row_factorcol_factor]=determine(nob)

Step 8 :r_final=(r1-1)+fix(r_act/row_factor) 'ending row for block-processing and embedding

Step 9 : c1=1 'starting column for block-processing and embedding

Step10:c_final=fix(c/col_factor) 'ending column for block-processing and embedding

Step 11: [n]=byte_to_bit(s_file,'out.txt')

*Step 12: If (n/8)>(r_act*c) then print ''Insertion not possible'', Go to step 134.*

Step 13: fp1=fopen('out.txt','r')

Step 14: str_file=fread(fp1)

Step 15: n1=n/8 'no. of bytes

Step 16 : ins=1

Step 17 :i=1

Step 18 : j=1

Step 19 : x(i,j)=str_file(ins)

Step 20 : ins=ins+1

Step 21 : j=j+1

Step 22 : If j<=8 then go to step 19

Step 23 :i=i+1

Step 24 : If i<=n1 then go to step 18

Step 25 : x1=up_shift(x,n1)'performs shift operations on the bits

Step 26 : x2=left_shift(x1,n1)

Step 27 : x3=down_shift(x2,n1)

Step 28 : x4=right_shift(x3,n1)

Step 29 : ins=1

Step 30 :i=1

Step 31 : j=1

Step 32 :x_final(ins)=x4(i,j) 'the bits after shift operations

Step 33 : ins=ins+1

Step 34 : j=j+1

Step 35 : If j<=8 then go to step 32

Step 36 :i=i+1

Step 37 : If i<=n1 then go to step 31

Step 38 : Calculate n1=fix(n/nob) ' n1= parts of secret message for each block

Step 39 :block_fac=0 'initially we set it to zero. It is incremented as each of the blocks are processed

Step 40 : p=1 'regulates the bits embedded in a particular block

Step 41 : n3=0

Step 42 : If block_fac= nob then go to step 86

Step 43 : If block_fac =(nob-1) then go to step 57

Step 44 :i=r1

Step 45 : j=c1

Step 46 : k=1

Step 47 : If p>(n1+n3) then go to step 51

Step 48 : If x_final(p)=0 and mod(y(i,j,k),2) =1 then y(i,j,k)=y(i,j,k)-1

Step 49:If x_final(p)=1 and mod(y(i,j,k),2) =0 then y(i,j,k)=y(i,j,k)+1

Step 50 : p=p+1

Step 51 : k=k+1

Step 52 : If k<=d then go to step 47

Step 53 : j=j+1

Step 54 :If j<=c_final then go to step 46

Step 55: i=i+1

Step 56 : If i<=r_final then go to step 45

Step 57: i=r1

Step 58 : j=c1

Step 59 : k=1

Step 60 :If p<=(n1+n2+n3) then go to next step. Else go to step 64.

Step 61 : If x_final(p)=0 and mod(y(i,j,k),2) =1 then y(i,j,k)=y(i,j,k)-1.

Step 62 : If x_final(p)=1 and mod(y(i,j,k),2) =0 then y(i,j,k)=y(i,j,k)+1

Step 63 : p=p+1

Step 64 : k=k+1

Step 65 : If k<=d then go to step 60.

Step 66 : j=j+1

Step 67 :If j<=c_final then go to step 59

Step 68: i=i+1

Step 69 :Ifi<=r_final then go to step 58

Step 70 : p=n1+n3+1 'is set after each block is processed in order to determine which bit to embed in the next block

Step 71 : n3=n3+n1 'is set after each block for proper calculation of p

Step 72 :block_fac=block_fac+1

Step 73: If block_fac>=row_factor then go to step 78. 'rows and columns are modified according to the value of block_fac

Step 74: r1=r_final+1

Step 75: r_final=r_final+fix(r_act/row_factor)

Step 76: c1=1

Step 77: c_final=fix(c/col_factor), Go to step 85.

Step 78: If mod(block_fac,row_factor)<>0 then go to step 82.

Step 79: c1=c_final+1

Step 80: c_final=c_final+fix(c/col_factor)

Step 81: r1=301

Step 82: r_final=(r1-1)+fix(r_act/row_factor), Go to step 85.

Step 83: r1=r_final+1

Step 84: r_final=r_final+fix(r_act/row_factor)

Step 85: Go to step 44.

Step 86: imwrite(y,o_file) 'writes the contents of the embedded image file to another file

Step 87: fp1=fopen('store_info.txt','w') 'file to store information about the secret message file

Step 88: r=n/8 'to store the number of bytes(i.e. size of the secret file)

Step 89: p=1

Step 90: If r<=0 then go to step 95.

Step 91: d=mod(r,10)

Step 92: size_arr(p)=d 'the number stored in d is copied t an array size_arr()

Step 93: p=p+1

Step 94: r=fix(r/10), Go to step 90.

Step 95:i=p-1

Step 96:d=size_arr(i)+48 'the numeric character is extracted from array size_arr()

Step 97:i=i-1

Step 98:Ifi>=1 then go to step 96

Step 99:r1=row_factor

Step 100: p=1

Step 101: If r<=0 then go to step 106.

Step 102: d=mod(r1,10)

Step 103: rowsize_arr(p)=d

Step 104: p=p+1

Step 105: r1=fix(r1/10), Go to step 101.

Step 106:i=p-1

Step 107:d=size_arr(i)+48

Step 108:i=i-1

Step 109:Ifi>=1 then go to step 107

Step 110:c=col_factor

Step 111: p=1

Step 112: If c<=0 then go to step 117 .

Step 113: d=mod(c,10)

Step 114: colsize_arr(p)=d

Step 115: p=p+1

Step 116: c=fix(c/10), Go to step 112.

Step 117:i=p-1

Step 118:d=size_arr(i)+48

Step 119:i=i-1

Step 120:Ifi>=1 then go to step 118

Step 121:n=nob

Step 122: p=1

Step 123: If n<=0 then go to step 128.

Step 124: d=mod(n,10)

Step 125: blocksize_arr(p)=d

Step 126: p=p+1

Step 127: n=fix(n/10) then Go to step 123.

Step 128:i=p-1

Step 129:d=size_arr(i)+48

Step 130:i=i-1

Step 131:Ifi>=1 then go to step 129

Step 132:Write the value of every d in the file corresponding to the file pointer fp1. Also after exiting from every loop do fprintf(fp1,' ')

Step 133: Call function stegano_file (o_file,'store_info.txt',file_fin) for final embed operation

Step 134:End

4.2 Information about the secret file

In function stegano_file(o_file, txt_file, file_fin)the information required to access the secret file,i.e. the size of the file, row_factor, coulumn_factor and number of blocks (nob) for the respective file is embedded in the cover file.

Function : stegano_file(o_file,txt_file,file_fin)

Where

o_file → Temporary embedded image file

txt_file → File that contains information about the secret file

file_fin → Final embedded image file.

Step 0: Start

Step 1: y=imread(o_file) 'reads the image file and stores pixels in array y()

Step 2: [r c d]=size(y) ' r=number of rows, c=number of columns, d=depth

Step 3: r1=r-299 'stores the row-range for the lower 300 pixels, as in order to embed information about the secret file, we have to use that range

Step 4 : [n]=byte_to_bit(txt_file,'output.txt')

Step 5: fp1=fopen('output.txt','r')

Step 6:str_file=fread(fp1)

Step 7: p=1 'acts as a regulator to store the number of bits

Step 8 :i=r1

Step 9 : j=1

Step 10: k=1

Step 11: If p>n then go to step 15

Step 12 : If str_file(p)=0 and mod(y(i,j,k),2) <>0 then

y(i,j,k)=y(i,j,k)-1

Step 13: If str_file(p)=1 and mod(y(i,j,k),2) =0 then y(i,j,k)=y(i,j,k)+1
Step 14: p=p+1. Go to step 16.

Step 15: Break from loop. Go to step 22.

Step 16 : k=k+1

Step 17: If k<=d then go to step 11

Step 18: j=j+1

Step 19: If j<=c then go to step 10

Step 20:i=i+1

Step 21: If i<=r then go to step 9

Step 22:imwrite(y,file_fin) 'writes the embedded image file into another file

Step 23 : End

5. DEMBED ALGORITHM

5.1 Extract secret message from Stego Image

It is known to the sender that the information required to process is hidden within the last 300 rows of the stego image. The order of block processing and shifting is also known. In the present study the secret message is encrypted before embedding inside cover message file. In future study the authors will apply encryption process before embedding inside cover file.

Function:stego_dembed(stego_file,out_file)

Where
stego_file→ The embedded image file.
out_file→ The secret file after extraction operation.

Step 0: Start

Step 1:Callfunctiondembed_this(stego_file,'deco.txt')'to extract information about the secret file

Step2: Obtain[sz,row_factorcol_factor]=extract_them ('deco.txt')

Step 3 : y=imread(stego_file)

*Step 4 : n=8*sz 'the number of bits of the file.*

Step 5 :x_ini=zeros(n,1) 'declare an array x_ini()

Step 6 : [r c d]=size(y)

Step7:fp1=fopen('temp.txt','w') 'opens a file temp.txt to store the bits of the secret message file

Step 8 : r1=301

Step 9 : r2=r-300

Step 10: r_act=(r2-r1)+1

Step 11: r_final=(r1-1)+fix(r_act/row_factor)

Step 12: c1=1

Step 13: c_final=fix(c/col_factor)

Step 14: n1=fix(n/nob)

Step 15: n2=mod(n,nob)

Step 16: p=1

Step 17: block_fac=0

Step 18: n3=0

Step 19: If block_fac<>nob then go to the next step. Else go to step 62

Step 20: If block_fac<> (nob-1) then go to the next step. Else go to step 34

Step 21: i=r1

Step 22: j=c1

Step 23: k=1

Step 24: If p<=(n1+n3) then go to the next step Else go to step 28.

Step 25: d1=mod(y(i,j,k),2)+48 'extracts the bits of the secret file

Step 26: x_ini(p)=d1 'the bits are stored in the array x_ini()

Step 27: p=p+1

Step 28: k=k+1

Step 29: If k<=d , go to step 24

Step 30: j=j+1

Step 31: If j<=c_final , go to step 23

Step 32: i=i+1

Step 33: If i<=r_final , go to step 22

Step 34: i=r1

Step 35: j=c1

Step 36: k=1

Step 37: If p>(n1+n2+n3) then go to step 41.

Step 38: d1=mod(y(i,j,k),2)+48

Step 39: x_ini(p)=d1

Step 40: p=p+1

Step 41: k=k+1

Step 42: If k<=d then go to step 37

Step 43: j=j+1

Step 44: If j<=c_final go to step 36

Step 45: i=i+1

Step 46: If i<=r_finalthen go to step 35

Step 47: block_fac=block_fac+1

Step 48: If block_fac>=row_factor then go to step 53.

Step 49: r1=r_final+1

Step 50: r_final=r_final+fix(r_act/row_factor)

Step 51: c1=1

Step 52: c_final=fix(c/col_factor). Go to step 60.

Step 53: If mod(block_fac,row_factor)<>0 then go to step 58.

Step 54: c1=c_final+1

Step 55: c_final=c_final+fix(c/col_factor)

Step 56: r1=301

Step 57: r_final=(r1-1)+fix(r_act/row_factor)

Step 58: r1=r_final+1

Step 59: r_final=r_final+fix(r_act/row_factor)

Step 60: p=n1+n3+1

Step 61: n3=n3+n1, Go to step 19

Step 62: ins=1

Step 63: i=1

Step 64: j=1

*Step 65: x1(i,j)=x_ini(ins) 'the bits of the secret message are transferred to an sz*8 2-dimensional array to perform the shift operations*

Step 66: ins=ins+1

Step 67: j=j+1

Step 68: If j<=8 then go to step 65

Step 69: i=i+1

Step 70: If i<=sz then go to step 64

Step 71: x2=right_shift(x1,sz) 'the shift operations are performed in the reverse order

Step 72: x3=down_shift(x2,sz)

Step 73: x4=left_shift(x3,sz)

Step 74: x5=up_shift(x4,sz)

Step 75: i=1

Step 76: j=1

Step 77: fprintf(fp1,'%c',x5(i,j)) 'the extracted bits are written into a file

Step 78: j=j+1

Step 79: If j<=8 then go to step 77

Step 80: i=i+1

Step 81: If i<=sz then go to step 76

Step 82: [n3]=bit_to_byte('temp.txt',out_file) 'they are converted into bytes in order to obtain the final output file

Step 83: End

5.2. Extract information about secret message

All information related to the secret file are stored in the lowermost 300 pixels of the cover image. The receiver doesn't know the limits. So, he extracts all the pixels from the specified part and stores it in a text file.

Function :dembed_this(i_file,out_file)

Where

i_file → The embedded image file.

out_file → The output text file.

Step 0: Start

Step 1 : y=imread(i_file)

Step 2 : [r c d]=size(y)

Step 3 : r1=r-299

Step 4 : fp1=fopen('temp.txt','w')

Step 5 :i=r1

Step 6 : j=1

Step 7 : k=1

Step 8 : d1=mod(y(i,j,k),2)+48 'extracts the bit

Step 9: fprintf(fp1,'%c',d1) 'stores it in the file temp.txt

Step 10: k=k+1

Step 11: If k<=d then go to step 8

Step 12: j=j+1

Step 13: If j<=c then go to step 7

Step 14: i=i+1

Step 15: If i<=r then go to step 6

Step16:[n3]=bit_to_byte('temp.txt',out_file) 'converts the bit-file into bytes in order to obtain the information

Step 17: End

5.3 Information from the extracted text file

The above algorithm extracts all pixels within the specified portion of the image and stores the result in a text file. This algorithm extracts the essential information required to dembed the secret file from the text file, i.e. the size of the secret file, number of blocks(nob), row_factor and column_factor.

Function :extract_them(file)

file : The text file.

Step 0:Start

Step1: fp1=fopen(file,'r') 'opens the file that contains information

Step2 : fp2=fread(fp1)

Step3 : n=length(fp2) 'the length of the file is stored in n.

Step 4 : p=1

Step 5 : s=0

Step 6 : c=0 'to keep a track of the number of variables required to extract. In this case maximum value of c is 4, as the variables are : size, number of blocks, row_factor and column_factor

Step 7 : s1=zeros(n,1) 'an array s1() that contains n zeros

Step 8 :i=1

Step 9 : s=0 'used to compute the value for each variable

Step 10: If fp2(i)=' ' or fp2(i)='0' or fp2(i)='1' or fp2(i)='2' or fp2(i)='3' or fp2(i)='4' or fp2(i)='5' or fp2(i)='6' or fp2(i)='7' or fp2(i)='8' or fp2(i)='9' , then go to the next step. Else go to step 29.

Step 11: If fp2(i) <>' ' then go to the next step. Else go to step 14. 'space is used as a delimiter for the values corresponding to the variables

Step 12: s1(p)=fp2(i) 'the characters corresponding to the variable are put in the array s1() till space is reached

Step 13: p=p+1 Go to step 30.

Step 14: ch=base2dec(s1,10) 'this is an inbuilt function that converts from character to decimal

Step 15: i1=1

*Step 16: s=s*10+ch(i1)*

Step 17: i1=i1+1

Step 18: If i1<=(p-1) then go to step 16

Step 19: p=1

Step 20: c=c+1

Step 21: If c<>1 then go to step 23.

Step 22: sz=s , go to step 30.

Step 23: If c<>2, then go to step 25.

Step 24: rf=s then go to step 30.

Step 25: If c<>3 then go to step 27.

Step 26: cf=s , go to step 30.

Step 27: If c<>4 then go to step 29.

Step 28: nb=s

Step 29: Break out of loop, Go to step 32.

Step 30: i=i+1

Step 31: If i<=n then go to step 9

Step 32: End

6. RESULTS AND DISCUSSIONS

The above method tested on various types of files such as .bmp, .jpg, .doc etc as secret message file and .bmp as cover file. In this section the data hiding done on some .bmp file(Cover file) and secret message is some .jpg file.

6.1 .jpg file embedded inside .bmp file:

Cover Image= papr.bmp

Secret image= papr1.jpg

Stego Image= papr2.bmp

Image extracted from stego file= papr3.jpg

6.1.1 Data hiding:

(size=5.49mb,row=1600,col=1200)

Fig4.1: Cover Image

(size=49.5kb,row=960,col=720)

Fig 4.2: Secret Image

(size=5.49mb,row=1600,col=1200)

Fig 4.3: StegoImage

6.1.2 Data Extraction:

(size=5.49mb,row=1600,col=1200)

Fig 4.4: Stego image

(size=49.5kb,row=960,col=720)

Fig 4.5: Extracted Secret Image

6.1.3Comparison between cover and stego image

The files are **different**

The first difference between these files is at byte offset 35. The differences are below shown starting from byte offset 0

Showing only the first 2000 bytes

C:\...11\Documents\MATLAB\i2.bmp - C:\...ocuments\MATLAB\image7.bmp

```
BM6.W...  42 4D 36 E4 57 00 00 00  -  BM6.W...  42 4D 36 E4 57 00 00 00
..6...(.  00 00 36 00 00 00 28 00  -  ..6...(.  00 00 36 00 00 00 28 00
..@.....  00 00 40 06 00 00 B0 04  -  ..@.....  00 00 40 06 00 00 B0 04
........  00 00 01 00 18 00 00 00  -  ........  00 00 01 00 18 00 00 00
....W...  00 00 00 E4 57 00 00 00  -  ....W...  00 00 00 E4 57 00 00 00
  ..  ..    00 00    00 00  -  ........  C4 0E 00 00 C4 0E 00 00
........  00 00 00 00 00 00 00 00  -  ........  00 00 00 00 00 00 00 00
...!(.!(  00 00 19 21 28 19 21 28  -  ...!(.!(  19 21 28 19 21 28
.!(.!(.!  19 21 28 19 21 28 1B 21  -  .!(.!(.!  19 21 28 19 21 28 1B 21
(.!(. '.  28 1B 21 28 1A 20 27 1A  -  (.!(. '.  28 1B 21 28 1A 20 27 1A
 '.").!*  20 27 1C 22 29 1C 21 2A  -   '.").!*  20 27 1C 22 29 1C 21 2A
. *. +..  1E 20 2A 1E 20 2B 1E 1F  -  . *. +..  1E 20 2A 1E 20 2B 1E 1F
-.,.,.-.  2D 1D 1E 2C 1D 1D 2D 1D  -  -.,.,.-.  2D 1D 1E 2C 1D 1D 2D 1D
.-../!!1  1D 2D 1F 1F 2F 21 21 31  -  .-../!!1  1D 2D 1F 1F 2F 21 21 31
$$4%%5&'  24 24 34 25 25 35 26 27  -  $$4%%5&'  24 24 34 25 25 35 26 27
5&'5&'5'  35 26 27 35 26 27 35 27  -  5&'5&'5'  35 26 27 35 26 27 35 27
(6'*8(+9  28 36 27 2A 38 28 2B 39  -  (6'*8(+9  28 36 27 2A 38 28 2B 39
%,;%,;#+  25 2C 3B 25 2C 3B 23 2B  -  %,;%,;#+  25 2C 3B 25 2C 3B 23 2B
<$,=&OB*  3C 24 2C 3D 26 30 42 2A  -  <$,=&OB*  3C 24 2C 3D 26 30 42 2A
4EK\en~.  34 45 4B 5C 65 6E 7E 04  -  4EK\en~.  34 45 4B 5C 65 6E 7E 04
........  8D 9A 98 A6 AD A8 B3 BA  -  ........  8D 9A 98 A6 AD A8 B3 BA
........  AD C3 C4 B4 D1 D1 BF D0  -  ........  AD C3 C4 B4 D1 D1 BF D0
........  CF BB D0 CF BB D4 D3 BF  -  ........  CF BB D0 CF BB D4 D3 BF
```

Stego Image Cover Image

Fig 4.6: Image comparison

6.2. rtf file embedded inside .bmp file:

Cover Image=papr.bmp

Secret Message=papr1.rtf

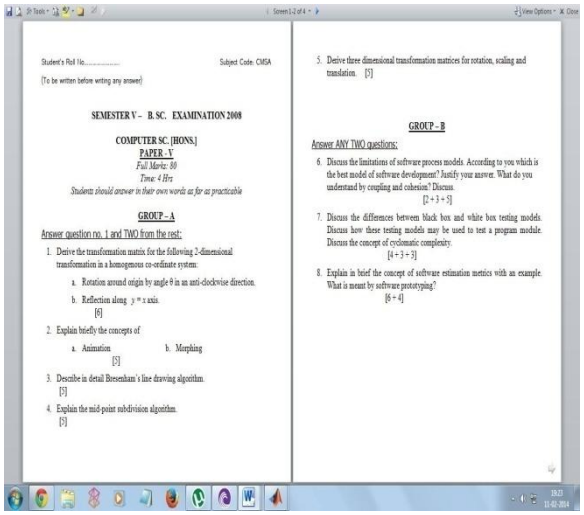
Stego Image=papr4.bmp

Retrieved Message=papr2.rtf

6.2.1 Data hiding:

(size=5.49mb,row=1600,col=1200)

Fig5.1 Cover Image



Student's Roll No..... Subject Code: CMSA

(To be written before writing any answer)

SEMESTER V – B. SC. EXAMINATION 2008

COMPUTER SC. [HONS.]

PAPER - V

Full Marks: 80

Time: 4 Hrs

Students should answer in their own words as far as practicable

GROUP – A

Answer question no. 1 and TWO from the rest:

1. Derive the transformation matrix for the following 2-dimensional transformation in a homogenous co-ordinate system:
   a. Rotation around origin by angle θ in an anti-clockwise direction.
   b. Reflection along $y = x$ axis. [6]
2. Explain briefly the concepts of
   a. Animation b. Morphing [5]
3. Describe in detail Bresenham's line drawing algorithm. [5]
4. Explain the mid-point subdivision algorithm. [5]
5. Derive three dimensional transformation matrices for rotation, scaling and translation. [5]

GROUP – B

Answer ANY TWO questions:

6. Discuss the limitations of software process models. According to you which is the best model of software development? Justify your answer. What do you understand by coupling and cohesion? Discuss. [2 + 3 + 5]
7. Discuss the differences between black box and white box testing models. Discuss how these testing models may be used to test a program module. Discuss the concept of cyclomatic complexity. [4 + 3 + 3]
8. Explain in brief the concept of software estimation metrics with an example. What is meant by software prototyping? [6 + 4]

(size=53kb)

Fig 5.2: Secret Message

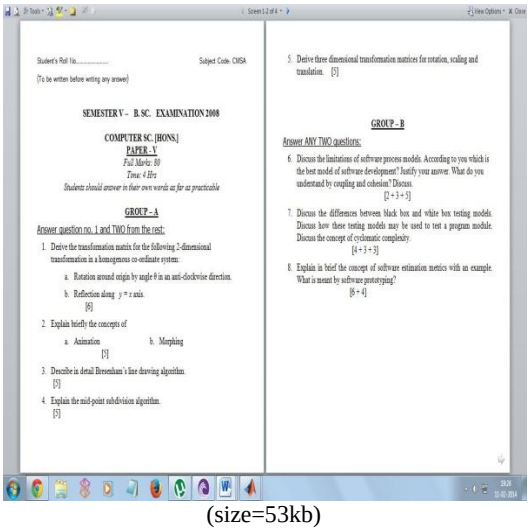
(size=5.49mb,row=1600,col=1200)

Fig 5.3: Stego Image

6.2.2 Data Extraction:

(size=5.49mb,row=1600col=1200)

Fig 5.4: Stego Image



SEMESTER V - B.SC. EXAMINATION 2008

COMPUTER SC. [HONS.]

PAPER - V

GROUP - A

GROUP - B

(size=53kb)

Fig 5.5: Retrieved Message

6.2.3 Comparison between cover and stego image

The first difference between these files is at byte offset 35. The differences are below shown starting from byte offset

Showing only the first 2000 bytes

C:\...ll\Documents\MATLAB\i2.bmp									C:\...ocuments\MATLAB\image7.bmp								
BM6.W...	42	4D	36	E4	57	00	00	00	BM6.W...	42	4D	36	E4	57	00	00	00
..6...(.	00	00	36	00	00	00	28	00	..6...(.	00	00	36	00	00	00	28	00
..@.....	00	00	40	06	00	00	B0	04	..@.....	00	00	40	06	00	00	B0	04
....	00	00	01	00	18	00	00	00	....	00	00	01	00	18	00	00	00
....W...	00	00	00	E4	57	00	00	00	... ...	00	00	00			00	00	00
.. ..			00	00		00	00		....	C4	0E	00	00	C4	0E	00	00
....	00	00	00	00	00	00	00	00	....	00	00	00	00	00	00	00	00
...!(.!(	00	00	19	21	28	19	21	28	.!(.!(			19	21	28	19	21	28
.!(.!(.!	19	21	28	19	21	28	1B	21	.!(.!(.!	19	21	28	19	21	28	1B	21
(.!(. '.	28	1B	21	28	1A	20	27	1A	(.!(. '.	28	1B	21	28	1A	20	27	1A
'.").!*	20	27	1C	22	29	1C	21	2A	'.").!*	20	27	1C	22	29	1C	21	2A
. *. +..	1E	20	2A	1E	20	2B	1E	1F	. *. +..	1E	20	2A	1E	20	2B	1E	1F
-..,..-.	2D	1D	1E	2C	1D	1D	2D	1D	-..,..-.	2D	1D	1E	2C	1D	1D	2D	1D
.-../!!1	1D	2D	1F	1F	2F	21	21	31	.-../!!1	1D	2D	1F	1F	2F	21	21	31
$$4%%5&'	24	24	34	25	25	35	26	27	$$4%%5&'	24	24	34	25	25	35	26	27
5&'5&'5'	35	26	27	35	26	27	35	27	5&'5&'5'	35	26	27	35	26	27	35	27
(6'*8(+9	28	36	27	2A	38	28	2B	39	(6'*8(+9	28	36	27	2A	38	28	2B	39
%,;%,;#+	25	2C	3B	25	2C	3B	23	2B	%,;%,;#+	25	2C	3B	25	2C	3B	23	2B
<$,=&OB*	3C	24	2C	3D	26	30	42	2A	<$,=&OB*	3C	24	2C	3D	26	30	42	2A
4EK\en~.	34	45	4B	5C	65	6E	7E	84	4EK\en~.	34	45	4B	5C	65	6E	7E	84
....	8D	9A	98	A6	AD	A8	B3	BA	....	8D	9A	98	A6	AD	A8	B3	BA
....	AD	C3	C4	B4	D1	D1	BF	D0	....	AD	C3	C4	B4	D1	D1	BF	D0
....	CF	BB	D0	CF	BB	D4	D3	BF	....	CF	BB	D0	CF	BB	D4	D3	BF

Stego Image Cover Image

Fig 5.6: Image comparison

6.3 Histogram analysis

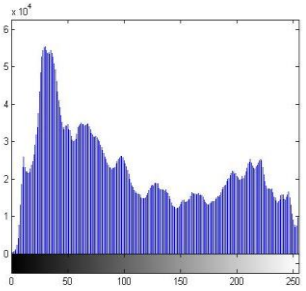


Fig 6.1: Cover Image

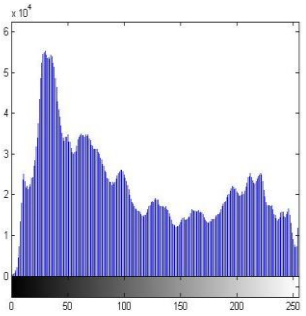


Fig 6.2: Stego Image

The histogram analysis of the two images doesn't reveal any difference at all to the human eye, which concludes that the damage is minimal.

6.4 Line graph analysis

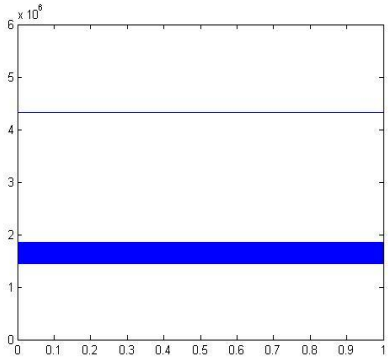


Fig 7: Line graph analysis

The line graph analysis where the authors have intentionally plotted the difference between two pixels corresponding to the

7. CONCLUSION AND FUTURE SCOPE

In the present study the authors tried to embed some secret message of any format inside a cover file of *.bmp format. The authors applied block processing over here, so that the bits of the secret message are entered in the LSB position according to the blocks where specific number of bits are reserved for each block. Before embedding the bits, we have applied four shifting operations to the bits of the secret file. The random generation of blocks in addition to the shifting operations makes the hidden data more secure. In the present study the authors did not applied any encryption process on secret message file. The authors are working on embedding

encrypted message the details of accessing the information are known to the receiver. The method may be further secured if we apply encryption. In our present work we have tried to break the image into several blocks and insert the secret message into the blocks by using LSB insertion method. There are many future scopes of this approach :

1. 1. Implement different algorithm in different blocks (suppose LSB+1 algorithm in first block LSB+3 algorithm in second block etc.)
2. Implement the size of blocks in unequal manner.
3. Use cryptography to encrypt secret message.
4. 4. Work can be extended to *.jpeg, *.png format cover files as well.

When embed a file in one cover file then there is permanent damage to the cover file. This damage must be minimal

8. ACKNOWLEDGEMENT

The authors sincerely express their gratitude to Department of Computer Science for providing necessary help and assistance. One of the authors AN is very much grateful to Fr.Dr. John Felix Raj, Principal of St. Xavier's College(Autonomous0, kolkata for giving facility to research work in steganography.

9. REFERENCES

[1] Data Hiding and Retrieval : AsokeNath, Sankar Das, AmlanChakraborti, published in **IEEE**"Proceedings of International Conference on Computational Intelligence and Communication.

[2] Advanced steganographic approach for hiding encrypted secret message in LSB, LSB+1, LSB+2 and LSB+3 bits in non standard cover files: JoyshreeNath, Sankar Das, Shalabh Agarwal and AsokeNath, International Journal of Computer Applications, Vol14-No.7,Page-31-35, Feb(2011).

[3] Advanced Steganography Algorithm using encrypted secret message : Joyshree Nath and AsokeNath International Journal of Advanced Computer Science and Applications, Vol-2, No-3, Page-19-24, March(2011).

[4] A Challenge in hiding encrypted message in LSB and LSB+1 bit positions in any cover files: executable files, Microsoft office files and database files, image files, audio files and video files: JoyshreeNath, Sankar Das, Shalabh Agarwal and AsokeNath: JGRCS, Vol-2,No.4,Page:180-185,April (2011).

[5] New Data Hiding Algorithm in MATLAB using Encrypted secret message: Agniswar Dutta, Abhirup Kumar Sen , Sankar Das , Shalabh Agarwal and AsokeNath : Proceedings of IEEE CSNT-2011 held at SMVDU(Jammu) 03-06 June2011,Page262-267(2011).

[6] An efficient data hiding method using encrypted secret message obtained by MSA algorithm: JoyshreeNath, MeheboobAlamMallik , Saima Ghosh and AsokeNath: Proceedings of the International conference Worldcomp 2011 held at Las Vegas(USA), 18-21 Jul(2011), Page 312-318, Vol-1(2011).

[7] A new randomized data hiding algorithm with encrypted secret message using modified generalized Vernam Cipher Method: RAN-SEC algorithm, Rishav Ray, JeeyanSanyal, Tripti Das, KaushikGoswami, Sankar Das and AsokeNath, Proceedings of IEEE International conference: World Congress WICT-2011 held at Mumbai University 11-14 Dec, 2011, Page No. 1215-1220 (2011).

[8] A new Challenge of hiding any encrypted secret message inside any Text/ASCII file or in MS word file : RJDA Algorithm, Rishav Ray, JeeyanSanyal, Debanjan Das and AsokeNath**,** Proceedings of IEEE CSNT-2012 conference held at Rajkot May 11-13, 2012, Page:889- 893(2012).

[9] A new data hiding algorithm with encrypted secret message using TTJSA symmetric key crypto system,SayakGuha, Tamodeep Das, Saima Ghosh,JoyshreeNath, Sankar Das, AsokeNath,Journal of Global Research in Computer Science, Vol 3, No.4, Page-11-16(2012).

[10] Advanced Digital Steganography using Encrypted Secret Message and Encrypted Embedded Cover File, Joyshree Nath, Saima Ghosh and AsokeNath, International Journal of Computer Applications(IJCA0975-8887), Vol 46, No-14, May ,(2012).

[11] Advanced Steganography Algorithm Using Randomized Intermediate QR Host Embedded with Any Encrypted Secret Message : ASA_QR Algorithm, Somdipdey, KalyanMondal, JoyshreeNath, AsokeNath, International Journal of Modern Education and Computer Science, No.6, Page 59-67, 2012.

[12] Data hiding algorithm using two-way encryption and embedding in a cover file – A new method for sending password or confidential message, JoyshreeNath ,Saima Ghosh and AsokeNath, Proceedings ofInternational Conference Worldcomp 2012 at lasVegas, USA,IPCV-12, Page-414-420(2012).

[13] A New Technique to Hide Encrypted Data in QRCodes™, SomdipDey, JoyshreeNath and AsokeNath,Proceedings of International Conference Worldcomp 2012 held at Las Vegas, USA, ICOMP-12, Page-94,101(2012).