International Journal of Applied Research 2017; 3(3): 907-911

ISSN Print: 2394-7500
ISSN Online: 2394-5869
Impact Factor: 5.2
IJAR 2017; 3(3): 907-911
www.allresearchjournal.com
Received: 20-01-2017
Accepted: 21-02-2017

Souma Pal
Research Scholar, Department of Computer Science & Engineering, University of Calcutta, West Bengal, India

Samir Kumar Bandyopadhyay
Professor, Department of Computer Science & Engineering, University of Calcutta, West Bengal, India

Correspondence
Souma Pal
Research Scholar, Department of Computer Science & Engineering, University of Calcutta, West Bengal, India

Bi-Level image steganography

Souma Pal and Samir Kumar Bandyopadhyay

Abstract
In our modern age, science is going to develop as much as the valuable data are becoming more insecure due to different causes to hamper it. Hence, to protect the valuable data, a technique, Steganography which is more advanced technique of cryptography is introduced. It is art and science of secure information communication where the secret data or confidential file are hidden in the cover file so that the outside world except the authenticated recipient(s) are completely unaware of the existence of the secret data or confidential file. The secret data are embedded within the cover file in such a way the human eyes except the intended recipient(s) cannot distinguish among original data and embedded data, they think nothing special. This paper emphasises on bi-level image steganography by applying transform domain and as well as spatial domain both with remembrance of payload or embedding capacity of a pixel.

Keywords: Steganography, gray code, payload, FFT.

Introduction
Steganography means "secret writing". The necessity of "secret writing" was introduced from ancient age, but the procedure was different. The age is going ahead as much as the upgrading approach is introducing in our modern society. Cryptography is a technique of "secret writing". But it is not as much as efficient like steganography because the encrypted data are appeared suspicion to malicious users and there are possibilities of being decrypted or being suppressed. Hence the intended information cannot reach to the actual destination effectively. In steganography, the secret data are hidden in an intelligent fashion within a cover file so that it is represented to malicious users as a normal view. No abnormalities are caught by the malicious users' eyes. So the extraction of original message from the stego message becomes more difficult to them. There are various techniques proposed in steganography like text, image, audio, video and so on.

Review Works
Steganalysis is the art of how to detect the existence of hidden information or hidden message in a multimedia document, and to discriminate stego object and non-stego-object with little or no knowledge about the steganography algorithm. The goal of steganalysis is to collect any evidence about the presence of embedded message [1]. If the steganography is the art of hiding messages into multimedia documents; the steganalysis is the art of detecting such the hidden messages. A message can be hidden in a document only if the content of a document has high redundancy [2]. Although the embedded message changes the characteristics and nature of the document, it is required that these changes are difficult to be identified by an unsuspecting user. On the other hand, steganalysis develops theories, methods and techniques that can be used to detect hidden messages in multimedia documents. The documents without any hidden messages or hidden information are called cover documents and the documents with hidden messages are named stego documents [3-4]. The multimedia document may be text, image, audio or video, this paper will concentrate on the image. Also stego document or stego object in this paper will be the stego image which is the image with hidden message or hidden information, and the cover document or cover object is the cover image which is the original image without and hidden message. The primary step of steganography and steganalysis process is to identify the image that the secret message will be hidden in, which will called cover image, then use any steganography algorithm to embed the message in the cover image, sometimes by the help of secret key, so

the cover image become stego image as shown in Figure 1. After that steganalysis process determines whether that image contains hidden message or not and then try to recover the message from it. In the cryptanalysis it is clear that the intercepted message is encrypted and it certainly contains the hidden message because the message is scrambled. But this may not be true in the case of steganalysis. The stego image may or may not be with hidden message. The steganalysis process starts with a set of unknown information streams. Then the set is reduced with the help of advanced statistical methods [5].

Proposed Work

This paper highlights on embedding a text (any length of 1byte or more) within a gray image file. The locations where the encryption will be done are determined by applying transform domain method, i.e. FFT(Fast Fourier Transform) and then the selected locations of the image are replaced by the bits to be embedded by implementing spatial domain method, i.e., LSB (Least Significant Bit) substitution method. And lastly, the selected locations where the encryption has been done are embedded. So, here two level embedding techniques are implemented by applying transform domain and spatial domain both.

In encryption procedure, at first, read a gray cover image of size 128X128, 256x256, 512X512, 1024X1024 or any, and if necessary, resize the image in the following sizes. Then the image is divided into (8X8) no. of blocks because the ASCII value of each character is 1byte or 8bits. No. of blocks are determined by image dimensions/(8X8), e.g., if image dimension is 512x512, then No. of block =(512x512)/(8x8)=4096. Again it is also to remember about embedding capacity or payload that no. of LSBs of a pixel can be replaced by input bits. The no. of characters to be embedded depends on embedding capacity or payload. The maximum bits of the text to be embedded within the image file is determined by 8XNo.of blocks X no. of characters. e.g., maximum bits or size of text= 8x4096X16=524288bits or 64 KB. The input text are converted into gray code. The cover image is splitted into (8X8) size of blocks and find out FFT (Fast Fourier Transform) of every pixel value of each block. Calculate (8X no. Of character / no. of embedded bits of a pixel) no. of largest FFT values among all pixel values of each block and mark the corresponding coordinates of the pixel values. Now the input bits are spatially encrypted in the marked locations of each block of the cover image using LSB (Least Significant Bit) substitution method and finally, the stego image is created. The marked coordinates where the bits are encrypted of each block are formed in the form of (10*row + column) and then combined into an empty image file whose dimensions are (no. of block+2)X8. The 1st row and 1st column of the file contains the value of no. of character to be embedded. The 2nd row and 1st column of the file contains the value of no. of embedded LSBs bits of a pixel and from 3rd row and onwards of the file contain coordinates values in the following form., e.g., if the coordinate value of a pixel (5,2), the cell of the file contains 52(10*5+2). The data of the newly created file will be encrypted and this is the 2nd level of encryption procedure. At first, every cell value of the file is reversed and then is added with its corresponding column value of the file. Thus this file is embedded. The stego image and the newly created image are then transferred to the intended recipient (s).

In decryption procedure, at first, newly created image file is opened and then the column value is subtracted from corresponding cell value and the actual pixel values where the bits have been concealed are obtained by reversing the difference values. By taking the LSB of the pixel values and following the reverse procedure, the bit streams are obtained. Then the bit streams are converted into gray to binary form and the resulting bit stream are actual input string which is the secret data.

Procedure

A. Encryption Procedure

1. Read a gray image as a cover image and calculate its size and resize it if necessary.
2. Calculate no. of blocks to be divided =image dimensions/ (8X8). Each block size is 8X8 because ASCII value of every character contains 8 bits.
3. Split the cover image into blocks of size (8X8).
4. Calculate FFT value for each element (pixel value) of every block of the image.
5. Generate a random integer number within 1 to 8 range which represents no. of characters to be embedded in each block and read an integer of power of 2 (range:0-3) which represents embedding capacity., i.e., no. of LSBs of a pixel of a block are embedded with bits of a character.
6. Find (8Xno.of characters to be embedded) no. of largest FFT values among all pixel values of each block and as well as mark the corresponding coordinates in the form of (10*row+ column) of each block of the cover image.

```
loc(1).block<-zeros(1,8/ no. of embedded bits of a pixel)
for c=1 to no. of blocks do
{k<-1 a[][]<-0 b[]<-0 for i=1 to 8 do {for j=1 to 8 do {a(1,k)<-f(c).block(i,j) k<-k+1}}
b<-sort(a,'descend')
ch<-1
cc<-1
for l=1 to (8*no. of characters / no. of embedded bits of a pixel) do
{flag<-1 for i=1 to 8 do {for j=1 to 8 do {if b(1,l)=f(c).block(i,j) then {num(c).block(ch,cc)<-p(c).block(i,j) loc(c).block(ch,cc)<-10*i+j loc1(c).block (ch,cc)<-10*i+j f(c).block(i,j)<-NaN if(mod(l,8 / no. of embedded bits of a pixel)=0)then {ch<-ch+1 Cc<-1} else {cc<-cc+1 } flag<-0 break}} if flag=0 then { break }}}}
```

7. Input string randomly whose No. of bits to be embedded = (8XNo.of blocks X no.of characters) and length of the string = (no. of characters X no. of blocks)
8. Convert the string into gray code.
9. Embed every bit in every marked location of each block of the cover image by LSB substitution method.

```
cc<-1 j<-1
for c=1 to no. of blocks do
{for ch=1 to no. of characters do {for i=1 to 8/ no. of embedded bits of a pixel do {col<-mod(loc(c).block(ch,i),10) row<-floor(loc(c).block(ch,i)/10) bin<-dec2bin(p(c).block(row,col),8) binnum<-str2num(bin) for k=length(bin) to 1 by -1 do {bin_mat(1,k)<-mod(binnum,10) Binnum<-round (binnum/10)}
m<-length(bin) x<-1 for x=1 to no. of embedded bits of a pixel do {bin_mat(1,m)<-smg(cc,j) m<-m-1 j<-j+1}
```

bin_rev<-bin2dec(num2str(bin_mat)) p(c).block(row,col)<-bin_rev bin[][]<-0 bin_rev[][]<-0 bin_mat[][]<-0} if cc<len then { cc<-cc+1 j<-1}}}

10. Create the stego image of same size of cover image.
11. Create a file which contains the location of the pixels where the bits are encrypted except The 1st row and 1st column of the file which contains the value of no. of character to be embedded.
12. Then the elements of the file are reversed and then added with the corresponding column value.

for i=1 to (no. of characters*no. of blocks+2) do {for j=1 to 8/ no. of embedded bits of a pixel do{ V<-q(i,j) cl<-mod(v,10) rw<-floor(v/10) vn<-(10*cl+rw)+j q(i,j)<-vn}}

B. Decryption Procedure

1. Read the stego image file and the file containing the locations of the encrypted bits.
2. Subtracts the corresponding column value from every cell value of that file and then reverses the cell value which is equivalent to actual locations where the bits are embedded.

(ar ac)<-size(ff) for i=1 to ar do {for j=1 to ac do {v<-ff(i,j)-j cl<-mod(v,10) rw<-floor(v/10) vn<-(10*cl+rw) ff(i,j)<-vn}}

3. 1st row and 1st column value of the file is the number of characters to be embedded.
4. 2nd row and 1st column value of the file is the number of embedded LSBs of a pixel.
5. Calculate no. of blocks to be divided =image dimensions/(8X8). Each block size is 8X8 because ASCII value of every character contains 8 bits.
6. Split the cover image into blocks of size (8X8).
7. Then split the file containing the actual locations where the bits are embedded into no. of blocks and each block size is (8x no. of characters to be embedded per block). 8 represent ASCII value of every character.
8. Decrypt the embedded data from each block of the image file using the pixel value from corresponding block.

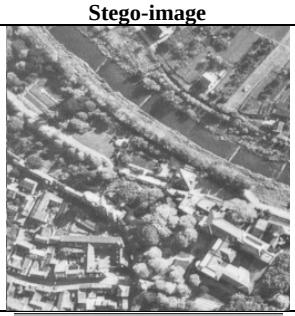
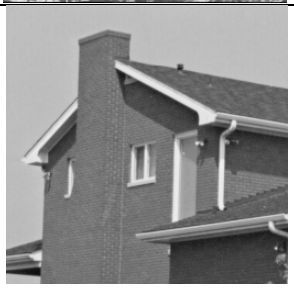
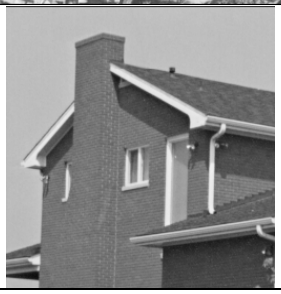
y<-1 j<-1 for c=1 to no. of blocks do {for cc=1 to no. of characters do {for i=1 to 8 / no. of embedded bits of a pixel do {col<-mod(reloc(c).block(cc,i),10) row<-floor(reloc(c).block(cc,i)/10) bin<-dec2bin(st(c).block(row,col),8) binnum<-str2num(bin) for k=1 to no. of embedding bits of a pixel do {d(y,j)<-mod(binnum,10) binnum<-round(binnum/10) j=j+1} bin[][]<-0 binnum[][]<-0} if y<len then {y<-y+1 j<-1}}}

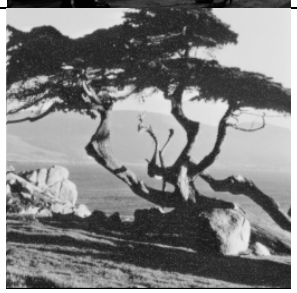
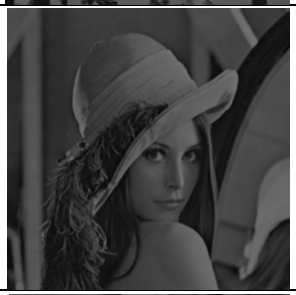
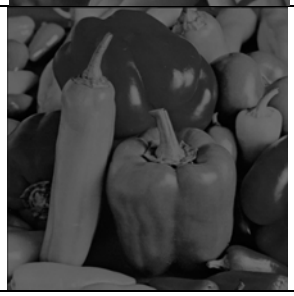
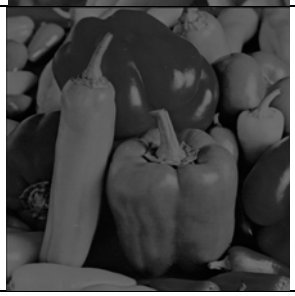
9. Then convert gray to binary form of every data.
10. Finally, obtain the actual bit stream which was encrypted.

Result

Name of the cover image file	Resolution of the image	Size of secret data	PSNR	MSE
Aerial.pgm	256X256	16KB	44.1388	2.5073
House.pgm	256X256	8KB	51.1524	0.4987
Couple.pgm	256X256	7KB	51.7023	0.4394
Tree.pgm	256X256	6KB	52.3811	0.3758
Sailboat.pgm	512X512	64KB	44.1575	2.4965
Lena.pgm	512X512	32KB	51.1502	0.4990
Elaine.pgm	512X512	28KB	51.7252	0.4371
Peppers.pgm	512X512	24KB	52.3919	0.3749

Output

Image file	Original image	Stego-image
Aerial.pgm		
House.pgm		

Couple.pgm		
Tree.pgm		
Sailboat.pgm		
Lena.pgm		
Elaine.pgm		
Peppers.pgm		

Conclusions

This paper has described not only the bi-level steganographic approaches with applying both transform domain and spatial domain, but also it has mentioned the payload of a pixel which means how many LSBs of the pixel can be replaced by the input bits so that the stego image looks like the cover image. Here, payload of a pixel is 2/8=1/4.i.e., maximum 2bits out of 8 bits of a pixel are

possible to replace and hence, maximum 16 characters can be embedded within a block of the cover image. Moreover, the cover image and the stego image sizes are remained same and another image file is created for storing encrypted locations.

References

1. Amritharanjan R, Akila R, Deepikachowdavarapu P. A comparative analysis of Image Steganography, International Journal of Computer Applications (0975-8887) 2010; 2(03).
2. Najme Maleki, Mehrdad Jalali, Majid Vafaei Jahan. Adaptive and non-adaptive data hiding methods for gray scale images based on modulus function. Egyptian Informatics Journal. 2014; 15:115-127.
3. Srinath NK, Usha BA, Narayan K, Tushara CK. Analysis of Data Embedding Technique in Image Steganography- A Survey. International Journal of Advanced Research in Computer and Communication Engineering. 2014; 3(6):2014.
4. Souma Pal, Prof. Samir Kumar Bandyopadhyay. Image Steganography. XOR Masking in LSB, Global Journal of Engineering Science and Researches, 2015; 2(6). ISSN-2348-8034.
5. Souma Pal, Prof. Samir Kumar Bandyopadhyay. Another Vision of Image Steganography, International Journal of Scientific and Applied science (IJSEAS), 2015; 2. ISSN-2395-3470