ijecs open access www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 8 August, 2014 Page No. 7932-7937

Highly Secure Invertible Data Embedding Scheme Using Histogram Shifting Method

Parvathy S[1], Abubeker K M[2]

[1]P G Scholar Communication Engineering,
Amal Jyothi College Of Engineering, India
parvathy.s07@gmail.com

[2]Assistant Professor ECE Department,
Amal Jyothi College Of Engineering, India
kmabubeker@amaljyothi.ac.in

Abstract: **This work presents a simple, easy to implement secure invertible data hiding method utilizing the advantages of both cryptography and steganography. The secret message to embed is first encrypted using CAESAR cipher method, and this encrypted message is further compressed using Huffman encoding. These encoded bits are embedded inside a host image thus creating the stego image. Data hiding method used here is the Histogram shifting method. In this method the maximum point and the minimum point of the histogram of an image is selected and slightly modifies the pixel intensities to embed data into the image. In the conventional Histogram shifting data hiding method the embedding capacity is limited by the peak value of the histogram. This proposed technique can overcome this limitation by compressing the input secret data bits to hide, thereby making possible to embed more data. Also encrypting the secret data before compression gives an add-on advantage of increased security to the secret message. Finally the stego image is compressed before transmission which allows efficient utilization of the allotted band width. The performance of this method is tested by plotting capacity versus PSNR value of the marked image. Experimental results show that this method gives improved visual quality of the marked image. Also this method gives high capacity and PSNR value when compared to other conventional data hiding methods.**

Keywords: Cryptography, Steganography, CAESAR cipher, Huffman Coding**.**

1. Introduction

Data embedding has become an important tool for integrity protection in modern multimedia technologies. Data hiding is a secure means of communication taking place by inserting information within other information. Data hiding technology evolved from the ancient Greek technology called Stegnography. Steganography literally means concealed writing. Steganography is possible by hiding text messages within another text message or text within an image or image within another image etc. In this proposed method text data is embedded within an image.

Data hiding methods can be classified into two groups mainly reversible and irreversible. In irreversible data hiding method, the host image used for hiding the data cannot be recovered without any loss. An example for this type is the LSB technique, which includes bit replacement of the least significant bits [4]. But in reversible data hiding method it is possible to recover the host image without any artifacts, after extracting the hidden data. Reversible data hiding method suggested by Ni et.al [2] is one of the most efficient data hiding method both in capacity and visual quality for all types of images. This method has proved the PSNR value of the watermarked image to be always greater than 48dB. Another high capacity and minimum distortion method was put forward by Lin et.al [3] based on the difference between the adjacent pixels of the image histogram. Efficiency of all the above methods depends upon the content of the image and for certain images, quality was lower at the same capacity.

M.Fallahpour [1] modified the method suggested by Ni et.al by applying histogram shifting to image blocks or tiles. This method improved the quality of the stego image and also the embedding capacity. In this method the image is divided into 4 or 16 blocks, and region of non interests ie, RONI's are selected from the image blocks. Considering an RONI as a separate image, histogram shifting is done in that image block. Histogram shifting is done by finding the maximum and minimum value of the image histogram, and then shifting the pixel values between its maximum and minimum

frequency. Data is inserted at the highest pixel value to obtain maximum embedding capacity.

Our proposed method is a modification to the above method. Although the existing method provides high data payload but still its capacity is limited by the histogram peak value. So the aim of the proposed method was to increase the data embedding capacity of the existing method without affecting the quality of the marked image. And that has been achieved by compressing the input data with Huffman encoding. Also the proposed method gives high confidentiality and secrecy to the input message by encrypting the message by CAESAR cipher method. Finally the water marked image is compressed using the same Huffman coding, which allows lossless compression of the image, preserving the image information while reducing the image data. This will substantially reduce the transmission and storage resources. This method completely avoids loss of information.

Thus our proposed method can be applied to many areas like medical, military applications where we should not only maintain the confidentiality of the data being embedded but also need to protect the host data from getting any loss after data retrieval.

Figure 1 shows the block diagram indicating the various steps involved in the proposed method.

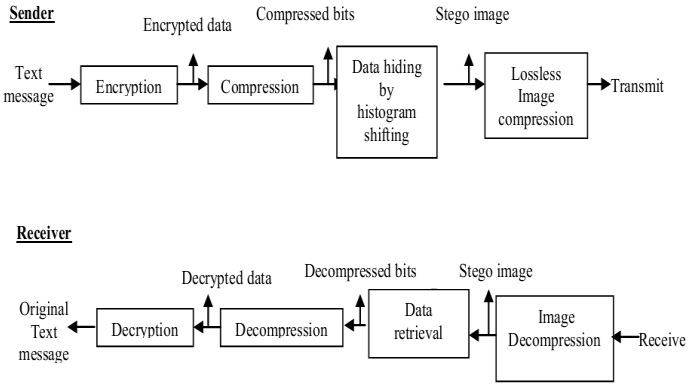


Figure 1: Block diagram

2. Proposed Method

The proposed method mainly involves four steps both at the sender and receiver side they are as follows:

- Step 1: Encryption of secret text message by CAESAR cipher method.
- Step 2: Compression of the encrypted data using Huffman coding which converts the encrypted data into binary bits.
- Step 3: Embedding the compressed bits into the host image by histogram shifting thus forming stego image
- Step 4: Stego image is compressed losslessly by Huffman encoding.

At the receiver side reverse of these processes take place and original secret text is recovered. Next section explains each of these steps in detail.

2.1 Encryption

Encryption is the new term given to cryptography in which a plain text is converted to a completely unrecognizable text called cipher text. Operation of this cipher text is controlled by a secret 'key' which is known only to the sender and the receiver. The same key is required for the receiver to recover the original message from the cipher text using a decryption algorithm.

Different types of encryption methods are used to encrypt data, here we use Caesar cipher technique for the encrypting purpose.

Caesar cipher is a simple shifting cipher in which each character in the secret text is 'shifted' to a certain number of places down the alphabet. Thus replacing each character in the text by another character cipher text is created. The secret key in this case is the number of characters used to shift the cipher alphabet

Consider an example. Let the secret message to embed is "Jack and Jill went up the hill". Let the secret shift key is '1'.

Plain text: Jack and Jill went up the hill
Cipher text: kbdl boe kjmm xfou vq uif ijmm

Mathematical analysis

First we convert the all the characters to corresponding numbers ie; 'a'=0, 'b'=1……'z'=25 and '!'=26, '@'=27, '$'=28, '*'=29….etc.

Formula for the encryption is given by:

$$E_j = (P_j + S)(mod\ Z) \quad (1)$$

E_j = jth character of encrypted text
P_j = jth character of plain text
S = shift
Z = length of the alphabet

Formula for the decryption is given by:

$$P_j = (E_j - S)(mod\ Z) \quad (2)$$

2.2 Data Compression

Data compression is the process in which the information is encoded into lesser number of bits than its actual representation. Data compression reduces both time and space. Compression methods can be classified mainly into two categories and they are Lossless and Lossy compression. Here we are interested in lossless compression of both data and image. Hence we have used Huffman encoding theorem which is a well known lossless compression algorithm[5].

Huffman coding effectively stores strings of data in binary code format. This method provides 'variable length coding ' in which the all characters in the data that you are encoding is transformed into binary bits according to the frequency at which each character is used. Thus the most frequently used character gets the least length codeword and the character that has been used least gets the maximum length code word (maximum 8 bits if you are dealing with uint8 data).

Let's consider a simple example to understand how Huffman coding works.

Let the string 'aaabbcd' be our input data to compress.
Length of the original string = 7. Let's calculate the probabilities of occurrence of each character in the string.

Probability, $P = \frac{frequency\ of\ \ occurance\ of\ a\ char}{length\ of\ the\ string}$ (3)

$P(a) = 0.42$, $P(b) = 0.28$, $P(c) = .14, P(d) = .14$

Now we need to construct the Huffman coding tree. For that arrange the probabilities of the chars in the decreasing order.

0.42 0.28 0.14 0.14

Consider each of these four chars as four different trees. Now we have to combine two trees at a time based on the least numbers first. It is shown in the following diagram.

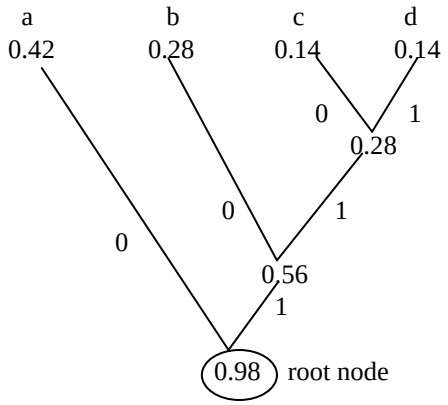


Figure 2: Huffman Coding Tree

Now form the root node, label the tree towards backward, or simply we can label '0' for the left branch and '1' for the right branch but in the backward manner. Thus we have created our Huffman coding tree.

Code words generated for our above characters are as follows:

a = 0, b=10, c=110, d=111

Length of the code words for each char is :

a=1, b=2, c=3, d=3

Average length of the code word is calculated by the formula

$Average\ length\ = \frac{1}{F(T)} \sum_{i=0}^{n} (L_{(i)} \times C_{(i)})$ (4)

$F(T)$ = total frequency / length of the string
n = no : of characters
$L_{(i)}$ = length of the code for ith char
$C_{(i)}$ = frequency of the ith char

In our example the average code word length can be calculated from the above equation and it is equal to:

$\frac{1}{7}(1 \times 3 + 2 \times 2 + 3 \times 1 + 3 \times 1) = 1.85$ bits

Thus the best possible expected code word length is 1.85 bits. Each character can be represented by an average length of 1.85 bits. Without compression each character required a minimum of 7 bits for its representation (since 7 bits are required for each char).

Now we can calculate the compression ratio using the formula:

Compression Ratio, CR= $\frac{uncompressed\ size(bits)}{compressed\ size\ (bits)}$ (5)

In our example compression ratio is equal to:

$$CR=\frac{7\times7}{2.85\times7} = 2.45$$

So by compressing the text data using Huffman coding it is possible to attain a CR ranging between 2 to 4. Data compression thus reserves space for extra data to send in our proposed method.
Decompression of the codes takes place at the receiver side with the help of the same Huffman tree, following the reverse of the compression steps.

2.3 Data Hiding

In our proposed method data hiding is done by histogram shifting method. The different steps in the data hiding process including the embedding and retrieval algorithm is explained in the following section.
After encrypting the secret data by Caesar cipher method we get a string of characters which is completely unrecognizable from the original text message. This encrypted string is then compressed using Huffman coding, which creates a string of binary digits, having lesser no: of bits than the original text representation.

We are going to hide our data in the image in the form of compressed data which is a binary sequence. First we need to select our gray scale image in which we are going to hide our data. Next we divide or cut the image into B_n non-overlapping blocks of n can be 4 or 16. From the blocks we can select the blocks with RONI and we treat each of those selected RONI blocks as a separate image and data hiding is done in each block. RONI selection is done for preserving high visual quality. Following section explains the embedding and retrieving algorithms.

Embedding Algorithm

Step 1

For a gray scale image with size $M \times N$ each pixel intensity $i \in [0,255]$. The histogram of the RONI image block is generated.From the histogram a maximum and minimum point is chosen. Let $(m1, m0)$ be such a max –min pair with $m1$ as the pixel intensity with maximum number of counts and $m0$ as the pixel intensity with minimum or zero pixel counts. Both $m1$ and $0 \in [0, 255]$.

Step 2

The image is scanned for pair $(m1, m0)$ and if:

(a) If $m1 > m0$ the gray values of the pixel between $m0$+1 and $m1$ are reduced by one (shifting the range of the histogram $[m0+1, m1]$ by 1 to the left). Thus a gap is created at grey level $m1$. The image is scanned again and the values of the pixels with grey value of $m1$-1 are added by 1, if the corresponding compressed data bits to embed are

'1', otherwise no modification is done.

(b) If $m0 > m1$, the grey values of the pixels between $m1$+1 and $m0$ -1 are incremented by one, creating a gap at pixel value $m1$+1. Then image is re-scanned and the values of the of the pixels with grey value of $m1$ are increased by 1 if the data bits to embed are '1', otherwise they will not be changed.

This completes the embedding section.The above algorithm is for a single peak-zero pair .We can select multiple pairs of max and min points in the same image block itself. This improves the data embedding capacity.

After finishing the data hiding process the embedded blocks are concatenated back together to form the stego image.this image further undergoes lossless compression, and the compressed imaged is transmitted to the receiver side.

For data retrieval, the value of the peak zero pair(s), no:of image blocks B_n and the cipher key are send as side information to the receiver side.

Retrieving Algorithm

The receiver gets the stego image in the compressed format and he need to first decompress it using the Huffman decoding algorithm.Thus the stego image is obtained and the image is divided into B_n blocks to select the RONI's, after which retrieval algorithm is applied to every block.

Following steps are done to extract the compressed message bits from the marked image and to recover the original cover image without any artifacts.

Step 1

Marked image is scanned in the same order as that used in the data embedding process.

For pair $(m1, m0)$ the image is scanned if:

(a) If $m1 > m0$, the pixel with grey value $m1$ indicates that the embedded data bit was 1 and it should not be modified. If the grey value of pixel is equal to $m1$-1, it indicates that the embedded data bit was 0. In this case its grey value has to be increased by 1. After that, the grey values of all pixels with grey values between $m0$ and $m1$-2 need to be increased by one.

(b) If $m0 > m1$, the pixel with grey value $m1$ indicates that the embedded data bit was 0 and they do not need to be modified. However, if the grey value of current pixel is equal to $m1$+1 it indicates the embedded data bit was 1. Then its grey value is reduced by 1. Also the grey values of all pixels with grey values between $m1$+2 and $m0$ are reduced by 1.

Thus we can recover the embedded compressed bits from the stego image. These compressed bits undergoes decompression which regenerates the cipher text.This encrypted data is then decrypted to recover the original secret message. Also the original image is obtained back losslessly.

2.4 Image Compression

Stego image obtained is losslessly compressed before sending **it** to the receiver**.** Lossless image compression is done using the same Huffman coding used for data compression. The data symbols considered here is the pixel intensities $[0, 255]$. Here also as in the above case a huffman tree is constructed and the pixel intensity that has the maximum frequency will be represented with the smallest code word and the pixel value having the lowest frequency will get the longest code word. Image compression reduces the transmission and storage resources. This method avoids any loss of information and hence the original image can be reconstructed losslessly.

3. Experimental Results And Analysis

The performance of the proposed data hiding has been tested on different test images and medical images. The results are analysed in the following section.

Let our secret message to hide is "jack and jill went up the hill". First step of the proposed method is data encryption by Caesar cipher method, and if the cipher key used is '1' we will get the encrypted text as :

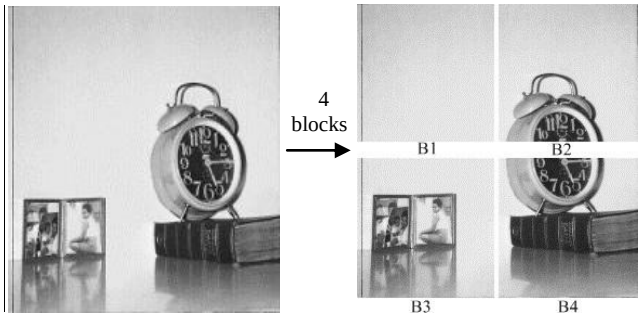
"kbdl boe kjmm xfou vq uif ijmm" (total 30 characters including white space)

After encryption data compression takes place by Huffman coding which results in the generation of binary bits having fewer bits than the original data. We need 7 bits (ASCII) for representing each character into binary, thus without compression we would require 7 X 30 = 210 bits for representing the cipher text into binary.

But only 110 bits were required for representing the same cipher text after compressing it with Huffman coding. Thus we have saved 100 bits as a result of compression. Using these saved bits an extra 14 characters can be send effectively.

For hiding data we have selected several test images including medical images. All the images taken were uncompressed gray scale images having different sizes $(M \times N \times 8)$.

Figure 3 shows a test image which is tiled into 4 blocks B1, B2, B3 and B4 and selection of RONI.



(a) (b)

Figure 3: RONI selection after image tiling

We can select B1 as the RONI and histogram shifting is done to this image block.

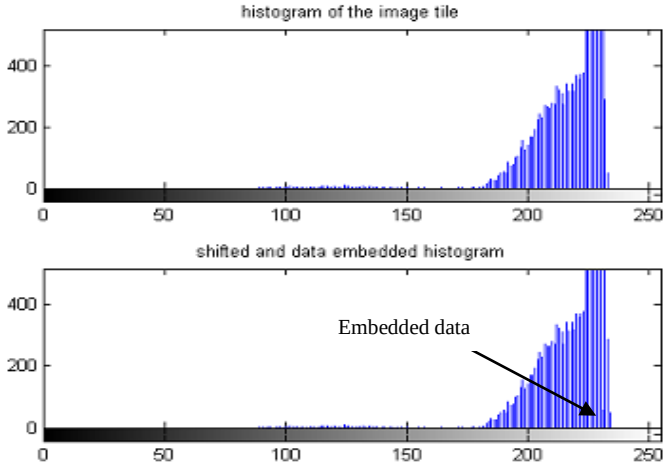


Figure 4: Image histogram & data embedded histogram

Max-min pair ie, $(m1, m0)$ of the B1 block is (230, 234). Thus 230 is the peak pixel value and is having the highest no:of pixel counts and it is equal to1889. So maximum data bits that can be transmitted with this image block is equal to its peak value which is equal to1889. In our experiment we first embedded input data in the image block B1 at its full capacity without data compression. Then we repeated our experiment for the same input data but after compressing it, and we took only 708 bits to hide it. Thus we have saved 591 bits and an extra 85 characters can be sent without any loss.

Figure. 5 shows the various images used for testing the performance of the proposed method.

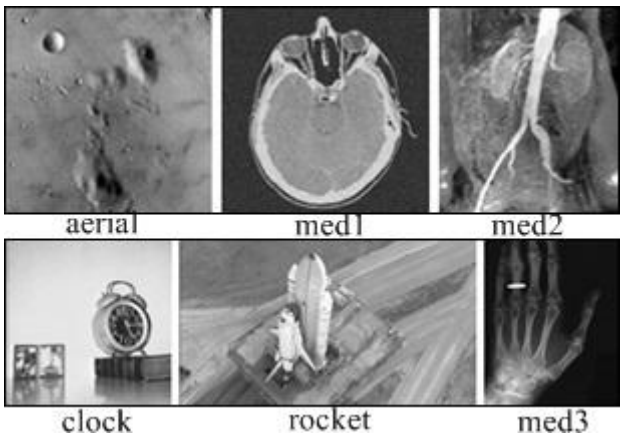


Figure 5: Various test images

The table 1 shows the results obtained on applying the proposed method on various test images shown above. Each image was tested by embedding data at its full capacity without data compression and with data compression and the results are analysed in terms of the extra characters obtained as a result of compression.

Table 1: Experimental results on various images at maximum capacity with and without compression

Images with RONI selected for embedding data	Max data bits embedded with full capacity (peak pixel count)	Max bits taken for embedding same data after compression	No :of Extra characters that can be send	CR
aerial	302	147	23	2.05
med1	852	436	60	1.9
med2	1299	708	85	1.83
clock	1889	1020	125	1.85
rocket	4966	2604	338	1.9
med3	16314	8547	1110	1.9

The following graph clearly explains our above concept by showing the comparison between input data with and without data compression.

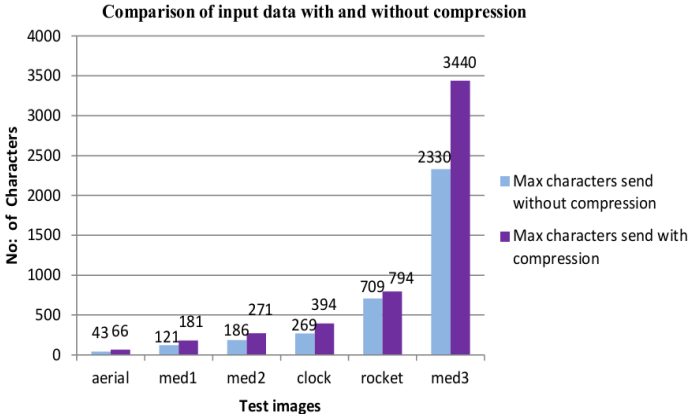


Figure 6: Comparison b/w input data with & without data compression

Encryption of the input data definitely gives more security to our secret message. Also lossless image compression using Huffman coding gives a compression factor of 2 to 5. An average code word length of 7.741 is obtained in almost all cases. That means instead of 8 bits required for representing each symbol in a grayscale image, we require only an average length of 7.741 bits thus reducing the file size.

Experimental results proved that PSNR value of the marked image is maintained more than 48dB in all cases and thus the visual quality of the marked image is found to be very high

4. Conclusion

A novel highly secure data embedding technique has been presented using both steganography and cryptography. The limitation of the existing method of having less payload capacity has been significantly improved in our proposed work by utilizing Huffman encoding. Moreover encryption of the secret message gives more security and confidentiality to the secret message. Finally compressing the image losslessly using the same Huffman algorithm allows reliable transmission of information using least amount of storage resources. Bandwidth requirement for transmitting an increased file size is also limited. Because of the less complexity of the algorithm its implementation is also very simple. method can be applied to many areas like medical, military applications where we should not only maintain the confidentiality and integrity of the data being embedded but

also need to protect the host data from getting any loss after data retrieval.

References

[1] M. Fallahpour, D. Megias, and M. Ghanbari, "Reversible and high-capacity data hiding in medical images," IET Image Process, vol. 5, Iss. 2, pp. 190-197, 2011.

[2] Ni, Z., Shi, Y.Q., Ansari, N., Su, W, "Reversible data hiding," IEEE Trans. Circuits Syst. Video Technol., 2006, 16, (3), pp. 354 – 362

[3] Lin, C.C., Tai, W.L., Chang, C.C.: 'Multilevel reversible data hiding based on histogram modification of difference images', Pattern Recognit. , 2008, 41, pp. 3582 – 3591.

[4] F. Shih, "Digital watermarking and stegnography, fundamentals and techniques", USSA: CRC Press, 2008

[5] Sayood, Khalid, Introduction to Data Compression, Sanfransisco Morgan Kaufmann 2000