# An Effective Compressed Image With Steganoghraphy

**Anwar J. Moosa**
*Babylon University, college of computer technology, dept of information networks*

## Abstract

steganograghy is the hiding for important and different structure of information in other medium without discovered method by an authorize.
In this work large colored image is hiding in smaller colored image (at least three times) by compressing embedded image with Huffman method and then using Oring bits to obtained a sequence of 0 and 1 values, Then with method that changed from Image Downgrading is used to hide this compressed image , so this method of steganography is more effective and have more secret by the number of secret operator between the two sides, this program is implemented in visual basic 0.6.

## الخلاصة

الأخفاء steganograghy هو اخفاء المعلومات المهمه والمختلفة الهيئات داخل وسائط أخرى بطريقه لاتسمح للمتطفل بأكتشافها وفي هذا البحث تم أخفاء صوره ملونه كبيره (أكبر بثلاث مرات ) داخل صوره ملونه صغيره ، حيث تم ضغط الصوره المراد أخفاؤها بأستخدام طريقة Huffman وطريقة Oring Bits لتتحول الصوره الى قيم من الصفر والواحد ليتم اخفاؤها داخل صوره الغطاء بأستخدام طريقه محوره من طريقة الأخفاء Image Downgrading . أثبتت الطريقه المقترحه كفاءه عاليه من حيث السعه(أخفاء صوره كبيره داخل صغيره) بالأضافه الى ان عملية الأسترجاع تعتمد على المعاملات السريه والمشتركه بين الطرفين المتعاملين مع الصوره والتي لايمكن معرفتها بسهوله، هذا البرنامج تم تنفيذه بلغة Visual Basic 0.6

## History

Hiding information inside images is a popular technique nowadays. An image with a secret message inside can easily be spread over the world wide web or in news groups. The use of steganography in newsgroups has been researched by German steganographic expert Niels Provos, who created a scanning cluster which detects the presence of hidden messages inside images that were posted on the net. However, after checking one million images, no hidden messages were found, so the practical use of steganography still seems to be limited.

To hide a message inside an image without changing its visible properties, the cover source can be altered in ”noisy” areas with many color variations, so less attention will be drawn to the modifications. The most common methods to make these alterations involve the usage of the least-significant bit or LSB, masking, filtering and transformations on the cover image. These techniques can be used with varying degrees of success on different types of image files.

## 1- Introduction

steganography is the art and science of hiding communication; a steganographic system thus embeds hidden content in unremarkable cover media so as not to arouse an eavesdropper’s suspicion. In the past, people used hidden tattoos or invisible ink to convey steganographic content. Today, computer and network technologies provide easy-to-use communication channels for steganography.

Essentially, the information-hiding process in a steganographic system starts by identifying a cover medium’s redundant bits (those that can be modified without destroying that medium’s integrity) [Andr 98] . The embedding process creates a *stego medium* by replacing these redundant bits with data from the hidden message.

Three different aspects in information-hiding systems contend with each other: capacity, security, and robustness. [Chen 2001] Capacity refers to the amount of information that can be hidden in the cover medium, security to an eavesdropper’s

inability to detect hidden information, and robustness to the amount of modification the stego medium can withstand before an adversary can destroy hidden information.
Information hiding generally relates to both watermarking and steganography. A watermarking system's primary goal is to achieve a high level of robustness—that is, it should be impossible to remove a watermark without degrading the data object's quality. Steganography, on the other hand, strives for high security and capacity, which often entails that the hidden information is fragile. Even trivial modifications to the stego medium can destroy it.

This research produce an effective method in steganography ,depending on hiding compressed image in covered image using developed Image Downgrading method.

## 1.1 Data compression

Finding the optimal way to compress data with respect to resource constraints remains one of the most challenging problems in the field of source coding. Data compression (or source coding) is the process of creating binary representations of data which requires less storage space than the original data [Keon96 ] [Kemp2002] .
Figure(1) shows the block diagram of data compression

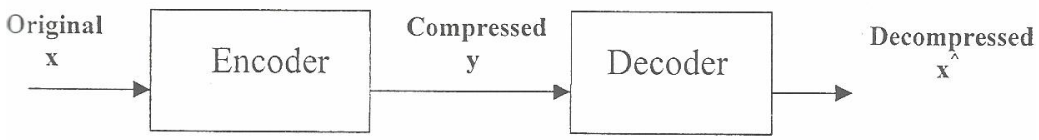



Figure (1) Block Diagram of Data Compression

The reason of compression is:
1- Conserving storage space.
2- Reducing time for transmission:-
Faster to encode, send, and decode them to send the original.
3- Progressive transmission:
Some compression techniques allow us to send the most important bits first so we can get a low resolution version of some data before getting the high fidelity version.
4- Reducing computation:
Use less data to achieve an approximate answer [Has2004 ].
There are many compression methods:

## 1.2 Huffman Method

In 1951, David Huffman and his MIT information theory classmates were given the choice of a term paper or a final exam. The professor, Robert M. Fano, assigned a term paper on the problem of 5nding the most efficient binary code. Huffman, unable to prove any codes were the most efficient, was about to give up and start studying for the final when he hit upon the idea of using a frequency-sorted binary tree, and quickly proved this method the most efficient.
The algorithm of Huffman is: [ Sal98] [Smith97] [Pull2003] [MÜll2003]

A fast way to create a Huffman tree is to use the heap data structure, which keeps the nodes in partially sorted order according to a predetermined criterion. In this case, the node with the lowest weight is always kept at the root of the heap for easy access.
Creating the tree:
1. Start with as many leaves as there are symbols.
2. Push all leaf nodes into the heap.
3. While there is more than one node in the heap:

- Remove two nodes with the lowest weight from the heap.
- Put the two nodes into the tree, noting their location.
- If parent links are used, set the children of any internal nodes to point at their parents.
- Create a new internal node, using the two nodes as children and their combined weight as the weight.
- Push the new node into the heap.

4. The remaining node is the root node.

Note it may be beneficial to generate codes with minimum length variance, since in the worst case when several long codeword have to be transmitted in a row it may lead to unwanted side effects (for example if e use a buffer and constant rate transmitter, it increases the minimum buffer size). To reduce variance every newly generated node must be favored among same weight nodes and placed as high as possible. This 1I balance the length, keeping same average rate.

## 1.3 Oring Bits

This method starts with a sparse string L1 of size n1 bits. In the first step,L1 is divided into k substrings of equal size. In each substring all bits are logically Ored, and the results (one bit per substring) become string L2 which will be compressed in step 2. For example, let substrings of L1, we divide into 16 substrings of size 4 each are:

L1=0000|0000|0000|0100|0000|0000|0000|1000|0000|0000|0000|0000
|0010|0000|0000|0000

After Oring each 4-bit substring we get the 16 -bit string

L2=0001|0001|0000|1000

In step2, the same process is applied to L2, and the result is the 4-bit string L3=1101, which is short enough so no more compression steps are needed. After deleting all zero substrings in L1 and L2 we end up with the three short strings

L1 =0100|1000|0010, L2= 0001 |0001|1000, L3= 1101.

The output stream consists of seven 4-bit sub strings instead of the igina1 16 (A few more numbers are needed, to indicate how long each sub string is).

The decoder works differently (this is an asymmetric compression method). It starts with L3 and considers each of its 1 bits a pointer to a sub string of L2 and each of its 0 bits a pointer to a sub string of all zero that's not stored in L2. This way string L2 can be reconstructed from L3, and string L1, in turn, from L2 [Sal98]. Figure (2) illustrates this process. The substrings shown in square brackets are the ones not contained in the compressed stream.

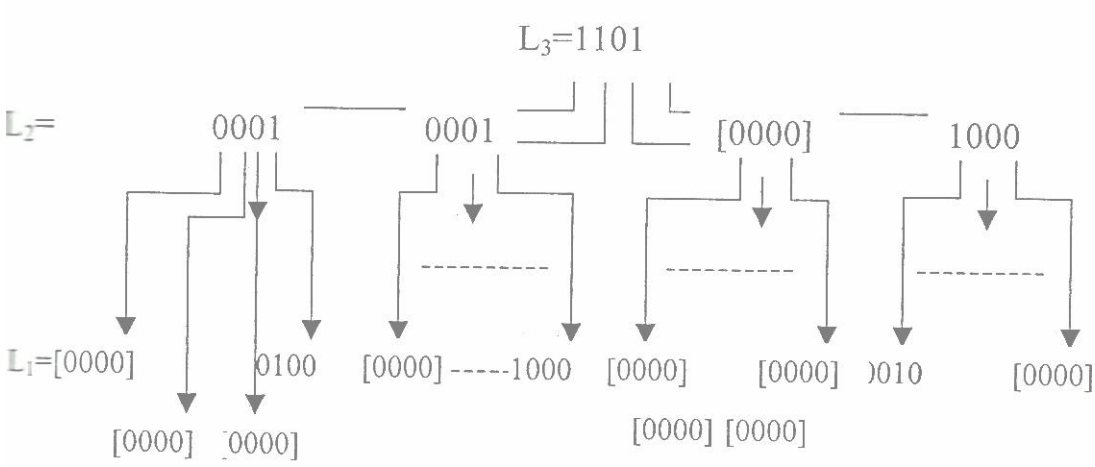
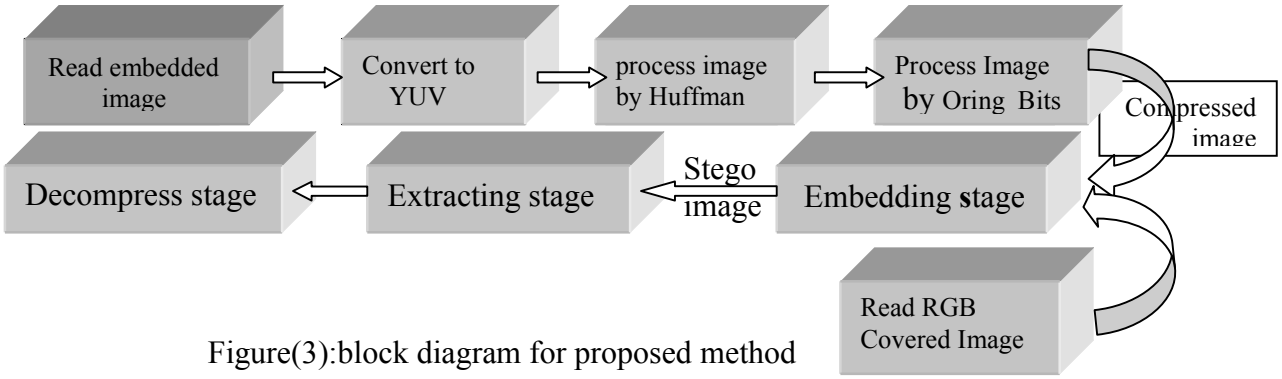



Figure (2): Reconstructing Process in Oring Bits.

## 2. The Proposal System

This proposed method is based on hiding compressed RGB colored image of type BMP in other RGB colored image of the same type. the following diagram explain this steps:




Figure(3):block diagram for proposed method

### 2.1 Read embedded image

Embedded image is RGB image and the data is read after the byte 54 (the length of header file in bmp image).

### 2.2 Convert to YUV

Y'UV signals are typically created from RGB (red, green and blue) source. Weighted values of R, G, and B are summed produce Y', a measure of overall brightness or luminance. U and V are computed as scaled differences between Y' and the B and R values [Poyn1999] [Mall&Joe] .

Defining the following constants:

$$\left.\begin{aligned} W_R &= 0.299 \\ W_B &= 0.114 \\ W_G &= 1 - W_R - W_B = 0.587 \\ U_{Max} &= 0.436 \\ V_{Max} &= 0.615 \end{aligned}\right\} \qquad (1)$$

Y'UV is computed from RGB as follows:

$$\left.\begin{aligned} Y' &= W_R R + W_G G + W_B B \\ U &= U_{Max}\frac{B - Y'}{1 - W_B} \\ V &= V_{Max}\frac{R - Y'}{1 - W_R} \end{aligned}\right\} \qquad (2)$$

The resulting ranges of Y', U, and V respectively are [0, 1], [$-U_{Max}$, $U_{Max}$], and [$-V_{Max}$, $V_{Max}$].

Equivalently, substituting values for the constants and expressing them as matrices gives:

$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

## 2.3 Process image by Huffman

The followed algorithm used for building the Huffman tree is:

1- Computing the probability for each color (Y') in the image.
2- Finding the two least probabilities according to the priority.
3- Giving value ‘0’ for character with the biggest value from step value ‘1’ for the other, this is on the condition that the two characters have two different values. In the condition of having the same value, upper (partial branch) gets value ‘0’ and the lower gets value ‘1’.
4- Sum the two probabilities.
5- The operation is repeated from step 2 to step 4 until the sum of chosen probabilities is equal to ‘1’.
6- Retrieve codeword for each character from right to left (probability).

Where the Huffman tree represents the dictionary that will associate pressed file and is used in the decoding operation.

## 2.4 Process Image by Oring Bits

After finding Huffman tree (Dictionary), all colors (Y') in the embedded image is coded according to the dictionary that has been obtained and according to the codeword for each color in the image. The result is a sequence of ‘0’ and ‘1’ saved in a file.

The file is inputted to another additional compression algorithm working on binary numbers; this method is Oring bits. This method rises from the compression rate.

After the Oring bits method is applied on the resulting file, the binary numbers are divided to blocks, each block includes eight bits (byte) and is converted to an integer number and saved in byte. The file is used then as a compressed file of image, mentioning that the dictionary will be associated with the image.

## 2.5 Embedding stage

In this stage compressed image (that have values of 0,1 ) will be hiding in colored RGB of covered image that is smaller than at least three time from embedded image there is no condition in this way for embedded image to be in conformity with covered image ,so the value of 0,1 (one bit) will be hiding in Least Bit Significant (LSB) for covered image in bit number N that accepted between the two sides (that is from 1 to 4 bit) ,and for great safety this value will be ciphered by XOR function with

secret key , Because covered image is colored RGB image so the hiding was with the following approach each Oring L1, L2, L3 (compressed embedded image) will be stored in one of pixel color depending on a sequential that generated randomly and stored with defined array that accepted between the two sides that contain the value 1,2,3 ,where 1 is R and 2 is G and 3 is B ,Such that each bit from L1 or L2 or L3 will be stored with matching pixel from R or G or B according to random number generated (1 or 2 or 3) , for example if the random number was 3 then oring L1 will be stored in R color.

The cause of use cipher is to increase the difficulties with retrieve embedded image, so the randomly behavior make the access to the hiding data are more difficulty ,and so the extracting stage is depending on some of the secret information that associated between the receiver and sender that the (Key) used in ciphering, the sequential numbers (S), the number of bit that specified for hiding (N) that from 1 to 4 in least bit significant ,L1 is the length of L1,L2 is the length for L2,L3 is the length for L3,and the number of hiding bits (M) that is 1.

The next scheme describe the embedding Algorithm:

## Embedding Algorithm

**Input** : The cover –image, The embedded-image M, N, L1, L2, L3, Key and S
**Output** : The Stego- image
Begin
Step1: Generate random positions by using the sequential (S) and put them in matrix
Step2: For each position in matrix with bit number N do
-Set the N low order bits of covered image Rc or Gc or Bc to zero.
-Cipher Re or Ge or Be of embed image by using "XOR" with secret key and shift right the results by 8-N.
-Add values from Rc,Gc,Bc of covered and L1, L2 , L3 of embedded image in Rs, Gs, Bs in stego-image
End for
End

## 2.6 Extracting stage

In this stage we extract embedded image from covered image by using the same input value with embedding algorithm that represents the associated value and accepted secret value between the two sides

The next scheme describe the Extracting Algorithm:

## Extracting Algorithm

**Input** : The Stego –image, L1, L2, L3, Key and S
**Output** : The embedded- image, The cover-image
Begin
Step1: Generate random positions by using the sequential (S) and put them in matrix
Step2: For each position in matrix with bit number N do

- Shift right Rs or Gs or Bs by 8-N.
- Put the shifted value of the Rs or Gs or Bs after decipher it using "XOR" with secret key into Rx, Gx , Bx in extracted image.

End for
End

## 2.7 Decompress stage

The decompression operation begins from the last byte which represents the Oring L2.the Oring L2 is converted to binary values. From the number of one's in binary value we determine the number of bytes for L2 which are also converted to binary in order to know the number of one's which give the number of bytes of L1 also converted to a binary value.

After determining each of Oring L3, L2, L1, now the Oring bits procedure is applied. As a result, we have a file as a sequence of 0 and 1, whichs the same result of the first case without Oring bits.

The resulting file is so compressed image. The image is read bit by bit in a sequential and is compared with the dictionary in order to get the original text.

After that we have to convert image from Y'UV to RGB to obtain the original image as follows [Poyn1999] [Mall&Joe]:

$$\left.\begin{aligned} R &= Y' + V\,\frac{1-W_R}{V_{Max}} \\ G &= Y' - U\,\frac{W_B(1-W_B)}{U_{Max}W_G} - V\,\frac{W_R(1-W_R)}{V_{Max}W_G} \\ B &= Y' + U\,\frac{1-W_B}{U_{Max}} \end{aligned}\right\} \qquad (3)$$

## 3. The results

We have tested the proposed method with the following (150 × 150)embedded image and (50 × 50) covered image:

Embedded image

covered image

And then we compute compression ratio (CR)for compressed embedded image equal (78) and peak signal to noise ratio (PSNR) is (21.05) for the same image ,the extracted image and the stego- image was as follows:

Extracted image

stego-image

The experimental result is PSNR for cover image and stego –image equal ( 41.67 db ), and Root Mean Square Error (RMSE) equal ( ٠.٤١ ). while for embedding image and extracted image the PSNR is (٣٦.81 db ) , and RMSE is (٠.79) .

The table (1) below explain all that results:

| Criteria | PSNR | RMSE | CR |
|---|---|---|---|
| embedded & extracted | 36.81 | 0.76 | _ |
| cover & stego | 41.67 | 0.41 | _ |
| embeded | 21.05 | _ | 78 |

Table (1) the results for the first experiment

And so we have tested the proposed method with the following (150 × 150)embedded image and (50 × 50) covered image:

Embedded image

covered image

And then we compute compression ratio for compressed embedded image equal (65) and peak signal to noise ratio (PSNR) is (٢7.02 ) for the same image ,the extracted image and the stego- image was as follows:

Extracted image

stego-image

The experimental result is PSNR for cover image and stego –image equal ( ٣٤.61 db ), and Root Mean Square Error (RMSE) equal ( ٠.92 ). while for embedding image and extracted image the PSNR is (٤1.61 db ) , and RMSE is (٠.٣8) .

The table (2) below explain all that results:

| Criteria | PSNR | RMSE | CR |
|---|---|---|---|
| embedded & extracted | 41.61 | 0.38 | _ |
| cover & stego | 34.61 | 0.92 | _ |
| embeded | 27.02 | _ | 65 |

table (2) the results for the second experiment

## 4. The conclusions:

A new approach is proposed to resolve two problems of substitution technique of image steganography. First problem is having low robustness against attacks which try to reveal the hidden message and second one is having low robustness against distortions with high average power.

It appears from the observation of results of the mentioned experiments that the proposed method success in the embedding and extracting process and so the method we used to compress the embedded image was successes by the results we get it for the compression ratio to the embedded image and so PSNR, and we see the increasing with compression ratio will increase the PSNR for stego and extracted image.

This proposed method is more effective because the two users must know the method that has been used for compression and so the method that used for decompress and so must know steganography method that has been used with its secret operators, that made this method is more secure.

## References

Andr 98 R.J. Anderson and F.A.P. Petitcolas, "On the Limits of Steganography," J. Selected Areas in Comm., vol. 16, no. 4,1998, pp. 474–481.

Chen 2001 B. Chen and G.W. Wornell , "Quantization Index Modulation: A Class Of Provably Good Methods for Digital Watermarking and Information Embedding," IEEE Trans. Information Theory, vol. 47, no. 4, 2001, pp.1423–1443

Keon96 W. Keong ,''Lossless and Lossy Data Compression", Nanyang Technological University,Singapore,1996.

Kemp2002 G.Kempe , "Computer Science Honours Research ReportCompression and computational Gene Finding"1,November 2002.

Has2004 F.Hasson, "Lempel-Ziv-Welch and Huffman compression",25 january 2004.

Sal98 D.Salomon, "Data Compression the complete reference", springverlag Newyourk,USA,1998.

Smith97 S.W.Smith, "The Scientist and Engieer's Guide to digital Signal Processing",1997.

Pull2003 J.M.Pullen , "Data Compression, Security Principles Integrity, Appropriate Use",2003.

MÜll2003 R.MÜller,"Image Compression"Part II:Image Processing ComputerGraphics and Image Processing" ,Winter Semester 2003.

Poyn1999 Poynton, Charles "YUV and luminance considered harmful.",1999 Mall&Joe Maller, Joe. "RGB and YUV Color", FXScript Reference