New Approach for Hiding Data in Colored Images

Associate. Prof Dr Saad Al-Ani
Oman Collage of Management &Technology

ABSTRACT

This research produces a new method for hiding text messages in a colored image by converting the text to a sequence of bits that are embedded within the image. The algorithm depends on a secret key between sender and receiver, and part of the pixels of the original image; it also depends on mathematical functions and matrices to increase the complexity. The embedded data will never affect the image in any recognized manner for the human eyes since the changes made on each group of pixels are no more than one or two least significant bits.

Key words: colored image, secret key, embedded data, modulation function.

1- Introduction

Steganography is an alternative to encryption for keeping data or correspondence confidential, whereas encryption leaves data visible but in a form that cannot be interpreted without the appropriate key [1]. Steganographic techniques allow communication between two authorized parties without an observer being aware that the communication is actually taking place; typically it uses pictures to embed confidential data [4]. Digital pictures (which contain large amounts of data) are used to hide messages on the Internet and on other communication media and that's so because the larger the cover message is (in data content terms—number of bits) relative to the hidden message, the easier it is to hide the latter [5]. The principle behind this is that pictures encoded as an array of binary words, the least significant bits in the words have very little effect on the subjective appearance of a picture, and consequently these bits can be used to store additional information without being noticed.

The strength points of this algorithm can be summarized as follows: First, the algorithm depends on the value of the actual image pixels in hiding and that means at most one or two bits are changed within each sub image of size n*n, sometimes we don't even have to change any bits when the value obtained from the pixels matches the values of the data we want to hide. Second, since we change a small number of bits within the whole image, the quality of the resulting image is very good and the changes can not be recognized. Third, the extraction of data is impossible for the attacker since the algorithm depends on a secret key to create the key matrices and that secrete key is periodically changed.

2- Matrices used in the algorithm

The algorithm uses many matrices in the hiding process. These matrices are:

*Ki(n*n):* the key matrix (will be explained in the next section).

*Gi(n*n):* the sub image matrix. Each element of Gi will be referred to according to its row and column as $P_{(r,c)}$.

*Bi(n*n):* the LSBs values matrix.(the creation of Bi will be explained within the algorithm)

The number (m) of bits used to hide the data could be two or three bits, so the elements of the matrix Bi will be all less than or equal to (2^m-1); If a three LSBs are used to hide the data, then the elements of Bi are all less than or equal to (2^3-1).

The size of the matrices (n*n) is determined depending on the size of the image and the number of bits we want to hide in that image; so n could be 2, then Ki, Gi, Bi will all be of size (2*2); and we are going to embed (m-bits) within (96 bit); (m could be 2 or 3 bits) If n=3 then Ki, Gi, Bi will all be of size (3*3), and we are going to embed m-bits within 216 bit; and if n=4, then we are going to embed m-bits within 384 bit; and so on.

3- Generating Key matrix

The key (Ki) is (nxn) matrix which is changed for each sub image (Gi). Ki is generated from a secrete word(s) known for the sender and the receiver. Assume that the secret key is "samule", so K1 is created from the letter "s" by converting it to ASCII code and taking part of it or repeating it to full K1(n*n) matrix.

Using the same method, K2 will be created using the ASCII code of the character "a", and so on.

The word "samule" gives six different key matrices which can be used repeatedly in sequential manner as much as we need (depends on how many bits we want to hide).

4- The Embedding Algorithm

First we have to determine the values of n and m to be used in the algorithm, so, assume n=4 and m=3, then:

1- Read a colored (RGB) image, divide the image into (4 x 4) sub images Gi, (i=1,2,..) ; (each sub image contains 16 pixels).
2- Determine the position in which we will start hiding the data; This is determined by using a random generator function.
3- For each sub image Gi, the following process will be done:

3-1- Convert the least three bits from the blue color byte to decimal for each pixel $P_{(r,c)}$ in Gi, the results will be saved in Bi (4x4) decimal matrix.

All elements of Bi are in the range $(0 \ldots 2^m-1)$.

3-2- The embedding process starts with the following calculations:

$$R_{(i,j)} = (K_{(i,j)} * B_{(i,j)})$$

$$i,j=1,2,\ldots,n$$

$$\text{Sum} = \sum_{i=1}^{n}\sum_{j=1}^{n} R_{(i,j)}$$

$$M = \text{Sum} \bmod 8$$

3-3 - To hide the following bits 0101101011100….., convert each three bits to the equivalent decimal number (i.e 010 is converted to D= 2), then find V and the sign S as follows:

$$V = | M - D |$$
$$S = \text{Sign}(M - D)$$

3-4 If the sign S is negative, add the value of V to one of the pixels $P_{(r,c)}$ in the sub image Gi, the values of (r,c) are calculated depending on the values of (i,j) of the point $Bi_{(i,j)}$ where:

$Bi_{(i,j)} >= (7-V)$ and $Ki_{(i,j)} = 1$

Then:

Blue of $\boldsymbol{P_{(r,c)}}$ ***= Blue*** $\boldsymbol{P_{(r,c)} + V}$

3-5 Otherwise (if S is positive) subtract the value of V from the pixel $P_{(r,c)}$ in the sub image Gi, the values of (r,c) are calculated depending on the values of (i,j) of the point $Bi_{(i,j)}$ where:

$Bi_{(i,j)} > V$ and $Ki_{(i,j)} = 1$

Then:

Blue of $\boldsymbol{P_{(r,c)}}$ ***= Blue*** $\boldsymbol{P_{(r,c)} - V}$

This process will force the value of modulation function to be equal to the embedded data.

5- Data Extracting

When the receiver gets the image, he will use the same random generator to find the start position, then he'll find Bi, calculates the modulation value by using the same key (Ki) and the same process. The modulation value (M) is the embedded data.

6- Implementation and results

In this section, some experimental results are demonstrated to show the effectiveness and the robustness of the proposed hiding algorithm; several different size test images are used for the hiding.

Figure (a) shows the original image that will be used for hiding data.

Figure a

To show how the algorithm works consider the following stream of bits as the data to be hidden (110011010100), Convert each three consecutive bits to decimal, the result will be (6, 3, 2, and 4).

The first hiding position (pixel) is determined using a random function.

From this pixel on, determine the sub image G1(4*4), now find B1(4*4) from G1. Use the first letter of the key word "samule" which is "s" to get the key matrix K1(4*4):

2	7	3	0
4	6	4	5
0	2	5	5
4	4	6	5

B1

1	1	1	0
0	1	1	1
1	1	0	0
1	1	1	1

K1

Now, multiply each $B1_{(i,j)}$ by $K1_{(i,j)}$ and add the results to obtain the summation (Sum= 50), therefore, M is equal to (2).

The first number to hide is (D=6), so:

V= |2-6| = 4

S= negative

Since the sign is negative, add the value of V to the sub image so that when the recipient calculates M, he gets 6 which is the data embedded in this sub image; To add V, first, find the proper pixel $P_{(r,c)}$ where, the addition of V to $B1_{(i,j)}$ won't be more than 7 and off course $K1_{(i,j)}$=1. the values of r,c is calculated depending on i,j. Then, the blue byte of P(r,c) will be increased by V. So, if the original value of $P_{(r,c)}$ = (121 77 74) in RGB, then after addition, $P_{(r,c)}$ will be changed to (121 77 78). Since the decimal 74 is equal in binary to (01001010), then after applying the algorithm it will be (01001110), whilst only one bit is changed, we actually hide three bits.

The same process will be applied to the next (4*4) sub image G2, to find B2. K2 will be created using the letter "a". by applying the algorithm to these matrices we'll get the following results:

Sum = 37, M = 5, V = 2,

S = positive, D = 3

Since S is positive then the pixel P(r,c) in G2 should be decremented by V, the position (r,c) is calculated depending on the position of B2(i,j) where:

$B2_{(i,j)} > V$ and $K2_{(i,j)}$=1

So, if the original value of $P_{(r,c)}$ = (127 74 68) in RGB, then after subtraction, $P_{(r,c)}$ will be changed to (127 74 66).

Continue this process until all the data are embedded. The resulting image won't be affected since we change one or two bits in each 16 pixels (i,e 16*24 = 384 bits).

The resulting image after embedding (250 characters) is shown in figure (b).

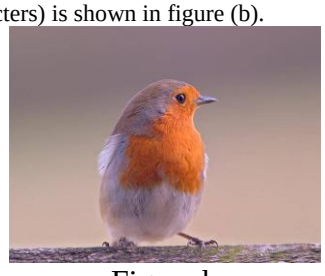
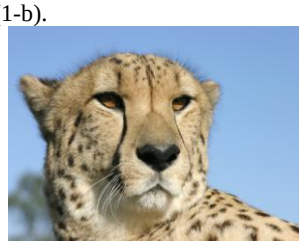


Figure b

7- Other experimental results

By applying the algorithm to other image with different size (figure 1-a), and by using n=5

for the matrices' size to embed (500 character), the resulting image was also not affected as it is shown in figure (1-b).



Figure(1-a): before hiding

Figure(1-b): after hiding

For the image in figure (2-a), a size of n=2 was used to embed (120 character), the result is shown in figure (2-b).

Figure(2-a): before hiding

Figure(2-b): after hiding

8- Conclusion

This research presents a new steganography technique for hiding ASCII bits in colored images (RGB). The algorithm is capable of hiding a large amount of data in a colored image without affecting the quality of the image; the algorithm uses the LSB's of blue byte (three or two LSBs) to embed the data. The algorithm uses a secrete key which makes the extracting of the embedded data impossible for the attackers. It also uses a random function to determine the starting position and to increase the difficulties of extracting the embedded data.

The size of the matrices used in the algorithm could be changed without affecting the security of the algorithm.

The number of the embedded bits within each sub image could be two or three bits, in both cases the result is a high quality image.

REFERENCES

[1] Cryptography and Network Security: Principles and Practice, 2nd edition, by William Stallings, Prentice Hall 2006

[2] J. Fridrich, M. Goljan, and R. Du, "Distortion-Free Data Embedding," to be published in Lecture Notes in Computer Science, vol. 2137, Springer-Verlag, Berlin, 2001.

[3] J. Anderson and F. A. P. Petitcolas, "On the Limits of Steganography," IEEE Journal on Selected Areas in Communications, Vol. 16, No. 4, May 1999, pp. 474–481.

[4] H.Kobayashi Y.Noguchi ,H.Kiya "A method of embedding binary data into JBEG bit streams ,"Trans. Of IEICE ,vol J83-D-II , pp.1469-1476 , 2000.

[5] Wayner, Peter (2002). Disappearing cryptography: information hiding: steganography & watermarking. Amsterdam: MK/Morgan Kaufmann Publishers. ISBN 1-55860-769-2. 4096000