Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 85 (2016) 31 – 38

Procedia Computer Science

International Conference on Computational Modeling and Security (CMS 2016)

Digital image steganography using variable length group of bits substitution

Gandharba Swain*

Department of Computer Science & Engineering, KL University, Vaddeswaram-522502, India

Abstract

This paper presents two new steganography methods in spatial domain. The basic idea is the substitution of a group of bits in a pixel by another group of bits of same length to hide one or two bits of secret data. The number of bits selected for substitution in a pixel depends upon some pre-defined conditions. The first method hides one bit per pixel and the second method hides two bits per pixel. Although a group of bits are substituted in a pixel, but the maximum change in a pixel value is not more than 2. The security has been improved by making the group length variable for different pixels. In most of the cases the pixel value remains same, but it hides one or two bits of secret data. The experimental results are compared with other methods and found to be satisfactory. The security has been evaluated and found to be improved.

Keywords: Steganography, LSB substitution, group of bits substitution, pixel value substitution, data hiding;

1. Introduction

Steganography is a technique for covert communication. It can be done with image, audio, and video carriers [1]. Image steganography techniques can be classified into two major categories such as spatial domain techniques and frequency domain techniques. The image in which secret message is hidden is called as the stego-image. By adding the unnatural message inside a natural image, there is a change in statistics, but if this change is so small, then it can not raise any suspicion [2]. When hiding information inside images usually LSB substitution method is used. But it is vulnerable to simple attacks. Some improvements and modifications have been proposed to strengthen this technique. Mathkour et al. [3] proposed a spiral based LSB substitution approach for hiding message in image. This approach is to divide the image into many segments and apply a different processing on each segment. A technique has been proposed to embed in LSBs of darkest and brightest pixels of an image to improve the security [4]. The LSBs of all the pixels can be supposed as an array and the secret message can be embedded at a maximum matching portion of the array, so that there will be minimum distortion [5,6]. Furthermore, by mapping the words on the LSB array and embedding at maximum matched locations can boost the security to a greater level [7]. To enhance the security and capacity one can hide adaptive number of bits in different pixels by measuring the embedding depth using some statistical analysis [8]. Jain and Ahirwal proposed an adaptive embedding approach by using a private stego-key [9]. The private stego-key consists of five gray level ranges that are selected randomly in the range from 0 to 255. The selected key shows the five ranges and each range substitute different number of bits at LSBs. This technique can embed upto 4 bits in a pixel. Swain and Lenka [10] proposed message bit dependent embedding, wherein the embedding locations in a pixel are selected depending on the bit pattern of the secret message.

Corresponding author Tel: +91-9573975571
E-mail address: gswain1234@gmail.com

doi:10.1016/j.procs.2016.05.173

A new track in image steganography called pixel value differencing (PVD) has been proposed by Wu and Tsai [11] in 2003. As per this technique a difference value d is calculated from every non-overlapping block of two consecutive pixels and it is substituted by a new difference value with the bits of secret message being embedded in it. A PVD technique using tri-way pixel value differencing have been proposed in [12]. Like these pixel-block techniques, other techniques based on correlation of a pixel with its neighboring pixels have also been evolved. Chang and Tseng [13] proposed two-sided, three-sided and four-sided side match methods wherein the correlation of a target pixel with its neighboring pixels is exploited to take embedding decision in the target pixel. Improved versions of side match methods have been proposed in [14]. Tseng and Leng proposed the 2-pixel block PVD with a range table based on perfect square [15]. Shen and Huang [18] proposed a steganography technique based on PVD and exploiting modification directions. They achieved higher hiding capacity and improved imperceptibility.

In this paper a new approach hitherto known as group of bits substitution (GBS) has been proposed. There are two schemes. The first one, called 1-bit GBS scheme hides one bit per pixel and the second one, called as 2-bit GBS scheme hides two bits per pixel. Here one pixel means one byte of the image. The embedding is done by replacing a group of bits in a pixel by another group of bits of same length. The rest of the paper is organized as follows. In section 2 & 3 these schemes are narrated. In section 4 the results are discussed and compared with other methods. Finally, the conclusions are given in section 5.

2. The 1-Bit GBS Method

The embedding procedure is as follows. Convert the image to binary and convert the secret message to binary. Convert the length of secret message to binary with 20 bit length. Append the length at the beginning of the binary message. Now simply you call the binary length plus binary message as *message*. One bit of message should be embedded in one byte of the cover image. Suppose the image length is N bytes and the *message* length is n bits. Then the message can be embedded if $n \leq N$. The embedding procedure comprises of the following steps.

Step1- The binary image, A comprises of bytes A_k, for k= 1 to N, where each A_k is 8 bits in length. Each byte is considered to be one pixel. The message, b comprises of bits b_i , for i= 1 to n, where each b_i is a bit 0 or 1.

Step2- If n>N, then display "The image is smaller and can not be embedded", otherwise initialize k=1 and i=1 and go to step3.

Step3-Suppose the eight bits of A_k are denoted as $d_1d_2d_3d_4d_5d_6d_7d_8$ and assume that, $D_1=d_1d_2d_3d_4d_5d_6d_7d_8$, $D_2=d_2d_3d_4d_5d_6d_7d_8$, $D_3=d_3d_4d_5d_6d_7d_8$, $D_4=d_4d_5d_6d_7d_8$, $D_5= d_5d_6d_7d_8$, $D_6= d_6d_7d_8$, and $D_7= d_7d_8$.

If the byte D_1 is 11111111 or 00000000; then apply simply one bit LSB substitution. After embedding we get D_1 = 11111111 or 11111110 or 00000000 or 00000001.

Else if D_1 is 01111111 or 10000000;
  if b_i is 0, then after embedding set D_1= 01111111, Otherwise, if b_i is 1, then after embedding set D_1=10000000.

Else if D_2 is 0111111 or 1000000;
  if b_i is 0, then after embedding set D_2= 0111111, otherwise, if b_i is 1, then after embedding set D_2=1000000.

Else if D_3 is 011111 or 100000;
  if b_i is 0, then after embedding set D_3= 011111, otherwise, if b_i is 1, then after embedding set D_3=100000.

Else if D_4 is 01111 or 10000;
  if b_i is 0, then after embedding set D_4= 01111, otherwise, if b_i is 1, then after embedding set D_4=10000.

Else if D_5 is 0111 or 1000;
  if b_i is 0, then after embedding set D_5= 0111, otherwise, if b_i is 1, then after embedding set D_5=1000.

Else if D_6 is 011 or 100;
  if b_i is 0, then after embedding set D_6= 011, otherwise, if b_i is 1, then after embedding set D_6=100.

Elseif D_7 is 01 or 10;
  if b_i is 0, then after embedding set D_7= 01, otherwise, if b_i is 1, then after embedding set D_7=10.

Step4- Increment k by 1 and i by 1.

Step5- If $i \leq n$ then go to step3, else go to step6.

Step6- Display "Embedded successfully".

The extraction procedure is very simple and is the reverse of embedding, described by the following steps. Suppose the binary stego-image is B and the message to be extracted bit by bit from it is m.

Step1- Initialize m to blank and initialize the counter, c=1.

Step2- The binary stego-image B comprises of bytes B_k , for k= 1 to N, where each B_k is one byte. Initialize k=1.

Step3- Suppose the eight bits of B_k are denoted as $d_1d_2d_3d_4d_5d_6d_7d_8$ and assume that, $D_1=d_1d_2d_3d_4d_5d_6d_7d_8$, $D_2=d_2d_3d_4d_5d_6d_7d_8$, $D_3=d_3d_4d_5d_6d_7d_8$, $D_4=d_4d_5d_6d_7d_8$, $D_5= d_5d_6d_7d_8$, $D_6= d_6d_7d_8$, and $D_7= d_7d_8$.

If the D_1 is 11111111 or 00000000 or 11111110 or 00000001, then the extracted bit d_8 is appended to m.

Else if D_1 is 01111111, then the extracted bit 0 is appended m. Else if D_1 is 10000000, then the extracted bit 1 is appended to m.

Else if D_2 is 0111111, then the extracted bit 0 is appended to m. Else if D_2 is 1000000, then the extracted bit 1 is appended to m.

Else if D_3 is 011111, then the extracted bit 0 is appended to m. Else if D_3 is 100000, then the extracted bit 1 is appended to m.

Else if D_4 is 01111, then the extracted bit 0 is appended to m. Else if D_4 is 10000, then the extracted bit 1 is appended to m.
Else if D_5 is 0111, then the extracted bit 0 is appended to m. Else if D_5 is 1000, then the extracted bit 1 is appended to m.
Else if D_6 is 011, then the extracted bit 0 is appended to m. Else if D_6 is 100, then the extracted bit 1 is appended to m.
Else if D_7 is 01, then the extracted bit 0 is appended to m. Else if D_7 is 10, then the extracted bit 1 is appended to m.

Step4- Set c=c+1. If $c \leq 20$ then, increment k by 1and then go to step3, otherwise go to step5.

Step5- Convert the 20 bits in m to decimal and then multiply by 7, which is the length of the embedded message in bits, say it is n.

Step6- Reinitialize m to blank, and for k= 21 to n repeat step3. Then we get m with n-20 bits length. Now go to step7.

Step7- Convert the binary message m to characters to formulate the secret message and then display “The message is extracted successfully”.

3. The 2-Bit GBS Method

The embedding procedure is as follows. Convert the image to binary and convert the secret message to binary. Convert the length of secret message to binary with 20 bit length. Append the length at the beginning of the binary message. Now simply you call the binary length plus binary message as *message*. Two bits of message should be embedded in one byte of the cover image. Suppose the image length is N bytes and the *message* length is n bits. Then the message can be embedded if $n/2 \leq N$. The embedding procedure comprises of the following steps.

Step1- The binary image, A comprises of bytes A_k, for k= 1 to N, where each A_k is 8 bits in length. Each byte is considered to be one pixel. The message, b comprises of bits b_i, for i= 1 to n, where each b_i is a bit 0 or 1.

Step2- If n>N, then display “The image is smaller and can not be embedded”, otherwise initialize k=1, i=1 and go to step3.

Step3-Suppose the eight bits of A_k are denoted $d_1d_2d_3d_4d_5d_6d_7d_8$ and assume that, $D_1=d_1d_2d_3d_4d_5d_6d_7d_8$, $D_2=d_2d_3d_4d_5d_6d_7d_8$, $D_3=d_3d_4d_5d_6d_7d_8$, $D_4=d_4d_5d_6d_7d_8$, $D_5= d_5d_6d_7d_8$, $D_6= d_6d_7d_8$, and $D_7= d_7d_8$.

If the byte D_1 is 11111111 or 00000000; then apply simply two bit LSB substitution. After embedding we get D_1= 11111100 or 11111101 or 11111110 or 11111111 or 00000000 or 00000001 or 00000010 or 00000011.

Else if D_1 is 01111111 or 10000000;
 if b_ib_{i+1} is 00, then after embedding set D_1= 01111111, if b_ib_{i+1} is 01, then after embedding set D_1= 01111110,
 if b_ib_{i+1} is 10, then after embedding set D_1= 10000001, if b_ib_{i+1} is 11, then after embedding set D_1= 10000000.

Else if D_2 is 0111111 or 1000000;
 if b_ib_{i+1} is 00, then after embedding set D_2= 0111111, if b_ib_{i+1} is 01, then after embedding set D_2= 0111110,
 if b_ib_{i+1} is 10, then after embedding set D_2= 1000001, if b_ib_{i+1} is 11, then after embedding set D_2= 1000000.

Else if D_3 is 011111 or 100000;
 if b_ib_{i+1} is 00, then after embedding set D_3= 011111, if b_ib_{i+1} is 01, then after embedding set D_3= 011110,
 if b_ib_{i+1} is 10, then after embedding set D_3= 100001, if b_ib_{i+1} is 11, then after embedding set D_3= 100000.

Else if D_4 is 01111 or 10000;
 if b_ib_{i+1} is 00, then after embedding set D_4= 01111, if b_ib_{i+1} is 01, then after embedding set D_4= 01110,
 if b_ib_{i+1} is 10, then after embedding set D_4= 10001, if b_ib_{i+1} is 11, then after embedding set D_4= 10000.

Else if D_5 is 0111 or 1000;
 if b_ib_{i+1} is 00, then after embedding set D_5= 0111, if b_ib_{i+1} is 01, then after embedding set D_5= 0110,
 if b_ib_{i+1} is 10, then after embedding set D_5= 1001, if b_ib_{i+1} is 11, then after embedding set D_5= 1000.

Else if D_6 is 011 or 100;
 if b_ib_{i+1} is 00, then after embedding set D_6= 011, if b_ib_{i+1} is 01, then after embedding set D_6= 010,
 if b_ib_{i+1} is 10, then after embedding set D_6= 101, if b_ib_{i+1} is 11, then after embedding set D_6= 100.

Else if D_7 is 01 or 10;
 if b_ib_{i+1} is 00, then after embedding set D_7= 11, if b_ib_{i+1} is 01, then after embedding set D_7= 10,
 if b_ib_{i+1} is 10, then after embedding set D_7= 01, if b_ib_{i+1} is 11, then after embedding set D_7= 00.

Step4 - Increment k by 1 and i by 2.

Step5- If $i \leq n$ then go to step3, else goto step6.

Step6- Display “Embedded successfully”.

The extraction procedure is very simple and is the reverse of embedding. It is described by the following steps. Suppose the binary stego-image is B and the message to be extracted bit by bit from it is m.

Step1- Initialize m to blank, and initialize counter, c=1.

Step2- The binary stego-image B comprises of bytes B_k, for k= 1 to N, where each B_k is one byte. Initialize k=1.

Step3- Suppose the eight bits of B_k are denoted as $d_1d_2d_3d_4d_5d_6d_7d_8$ and assume that, $D_1=d_1d_2d_3d_4d_5d_6d_7d_8$, $D_2=d_2d_3d_4d_5d_6d_7d_8$, $D_3=d_3d_4d_5d_6d_7d_8$, $D_4=d_4d_5d_6d_7d_8$, $D_5= d_5d_6d_7d_8$, $D_6= d_6d_7d_8$, and $D_7= d_7d_8$.

If the D_1 is 11111100 or 11111101 or 11111110 or 11111111or
 00000000 or 00000001 or 00000010 or 00000011, then the extracted bits d_7d_8 are appended to m.

```
Else if $D_1$ is 01111111, then the extracted bits are 00. Append it to m.
    if $D_1$ is 01111110, then the extracted bits are 01. Append it to m.
    if $D_1$ is 10000001, then the extracted bits are 10. Append it to m.
    if $D_1$ is 10000000, then the extracted bits are 11. Append it to m.
Else if $D_2$ is 0111111, then the extracted bits are 00. Append it to m.
    if $D_2$ is 0111110 , then the extracted bits are 01. Append it to m.
    if $D_2$ is 1000001, then the extracted bits are 10. Append it to m.
    if $D_2$ is 1000000, then the extracted bits are 11. Append it to m.
Else if $D_3$ is 011111, then the extracted bits are 00. Append it to m.
    if $D_3$ is 011110 , then the extracted bits are 01. Append it to m.
    if $D_3$ is 100001, then the extracted bits are 10. Append it to m.
    if $D_3$ is 100000, then the extracted bits are 11. Append it to m.
Else if $D_4$ is 01111, then the extracted bits are 00. Append it to m.
    if $D_4$ is 01110, then the extracted bits are 01. Append it to m.
    if $D_4$ is 10001, then the extracted bits are 10. Append it to m.
    if $D_4$ is 10000, then the extracted bits are 11. Append it to m.
Else if $D_5$ is 0111, then the extracted bits are 00. Append it to m.
    if $D_5$ is 0110, then the extracted bits are 01. Append it to m.
    if $D_5$ is 1001, then the extracted bits are 10. Append it to m.
    if $D_5$ is 1000, then the extracted bits are 11. Append it to m.
Else if $D_6$ is 011, then the extracted bits are 00. Append it to m.
    if $D_6$ is 010, then the extracted bits are 01. Append it to m.
    if $D_6$ is 101, then the extracted bits are 10. Append it to m.
    if $D_6$ is 100, then the extracted bits are 11. Append it to m.
Else if $D_7$ is 11, then the extracted bits are 00. Append it to m.
    if $D_7$ is 10, then the extracted bits are 01. Append it to m.
    if $D_7$ is 01, then the extracted bits are 10. Append it to m.
    if $D_7$ is 00, then the extracted bits are 11. Append it to m.
Step4- Set c=c+2. If c $\le$20 then, increment k by 1 and go to step3, otherwise go to step5.
Step5- Convert the 20 bits in m to decimal and then multiply by 7, which is the length of the embedded message in bits, say it is n.
Step6- Reinitialize m to blank, and for k=11 to n/2 repeat step3. Then we get m with n-20 bits length. Now go to step7.
Step7- Convert the binary message m to characters to formulate the secret message and then display "The message is extracted
       successfully".
```

4. Results and Discussion

4.1. Results

The proposed methods are implemented using MATLAB and the results are compared with other methods. Fig.1 represents two original images. Fig.2 and Fig.3 represents the stego-images of 1-bit GBS and 2-bit GBS schemes respectively with 7,00,000 bits of data hidden in each image. It can be observed that the stego-images do not show any identifiable marks.

(a) Lena (b) Baboon

Fig.1 (a),(b) Original Images

In Table 1 and Table 2 the proposed GBS techniques are compared with the other schemes. The comparison parameters are peak signal-to-noise ratio (PSNR), hiding capacity, and quality index (Q). The PSNR [16] is a measure of distortion. The quality of the stego-images are evaluated by using the universal image quality index (Q) [17]. The hiding capacity refers to the maximum amount of information that can be embedded in the cover image. It is represented in bits. The embedding bit rate is calculated as the ratio of the embedding capacity to the size of the image. It can be represented as bits per pixel, or bits per byte.

The PSNR, capacity, quality index and bit rate of the proposed 1-bit GBS scheme are same as 1-bit LSB substitution scheme. The PSNR and quality index of proposed 1-bit GBS scheme are higher, but capacity and bit rate are lesser than that of Wu & Tsai’s and Tseng & Leng’s schemes. The step effects in pixel difference histograms of Wu & Tsai’s and Tseng & Leng’s schemes are very high as compared to the proposed 1-bit GBS scheme. Although the capacity and bit rate of proposed 2-bit GBS scheme is same as that of 2-bit LSB substitution scheme, but PSNR is better than that of 2-bit LSB substitution scheme. The PSNR, capacity and bit rate of proposed 2-bit GBS scheme is better than that of Wu & Tsai’s scheme and Tseng & leng’s scheme. Furthermore, the step effects in the pixel difference histogram in Wu & Tsai’s scheme and Tseng & Leng’s scheme are very high, but in the proposed 2-bit GBS scheme it is very less.

Fig.2 Stego-images in 1-bit GBS scheme

Fig.3 Stego-images in 2-bit GBS scheme

Table 1. Comparison of the results

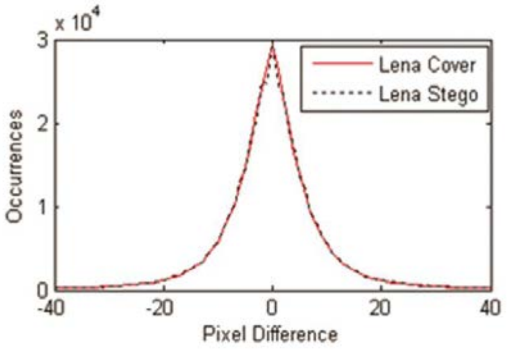
Images 512×512	Wu & Tsai’s PVD scheme				1-bit LSB substitution scheme				Proposed 1-bit GBS scheme			
(color)	PSNR	Capacity	Q	Bit-rate	PSNR	Capacity	Q	Bit rate	PSNR	Capacity	Q	Bit rate
Lena	43.63	1232606	0.9995	1.56	51.65	786432	0.9999	1.0	51.63	786432	0.9999	1.0
Baboon	38.33	1403491	0.9984	1.78	51.64	786432	0.9999	1.0	51.64	786432	0.9999	1.0
Tiffany	44.07	954070	0.9992	1.21	51.70	786432	0.9998	1.0	51.71	786432	0.9998	1.0
Peppers	43.07	1174751	0.9996	1.49	51.64	786432	0.9999	1.0	51.62	786432	0.9999	1.0
Jet	43.95	1220544	0.9993	1.55	51.64	786432	0.9998	1.0	51.65	786432	0.9998	1.0
Boat	41.30	1278971	0.9994	1.62	51.64	786432	0.9999	1.0	51.64	786432	0.9999	1.0
House	41.22	1256404	0.9991	1.59	51.64	786432	0.9999	1.0	51.64	786432	0.9999	1.0
Pot	43.95	1163700	0.9997	1.47	51.64	786432	0.9999	1.0	51.63	786432	0.9999	1.0
Average	42.44	1210567	0.9992	1.53	51.65	786432	0.9999	1.0	51.64	786432	0.9999	1.0

Table 2. Comparison of the results

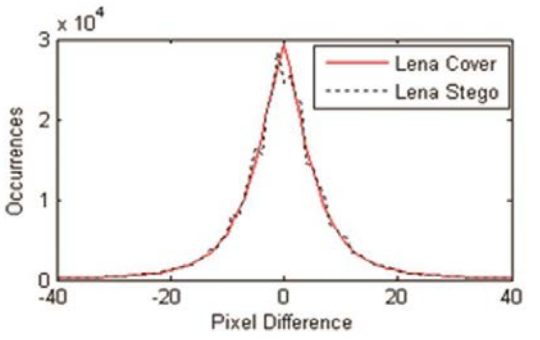
Images 512×512	2-bit LSB substitution scheme				Proposed 2-bit GBS scheme				Tseng & Leng’s scheme			
(color)	PSNR	Capacity	Q	Bit rate	PSNR	Capacity	Q	Bit rate	PSNR	Capacity	Q	Bit rate
Lena	47.57	1572864	0.9998	2.0	49.90	1572864	0.9999	2.0	47.51	970065	0.9998	1.23
Baboon	47.57	1572864	0.9998	2.0	49.88	1572864	0.9998	2.0	45.13	1157084	0.9996	1.47
Tiffany	47.44	1572864	0.9996	2.0	49.33	1572864	0.9997	2.0	48.03	900196	0.9997	1.14
Peppers	47.50	1572864	0.9998	2.0	49.70	1572864	0.9999	2.0	47.32	992058	0.9998	1.26
Jet	47.56	1572864	0.9997	2.0	49.86	1572864	0.9998	2.0	47.84	892178	0.9997	1.13
Boat	47.53	1572864	0.9998	2.0	49.85	1572864	0.9999	2.0	46.42	1046956	0.9998	1.33
House	47.59	1572864	0.9998	2.0	49.87	1572864	0.9998	2.0	46.77	938943	0.9997	1.19
Pot	47.51	1572864	0.9998	2.0	49.71	1572864	0.9999	2.0	48.87	843497	0.9999	1.07
Average	47.53	1572864	0.9998	2.0	49.76	1572864	0.9998	2.0	47.23	967622	0.9997	1.23

4.2. Security Analysis

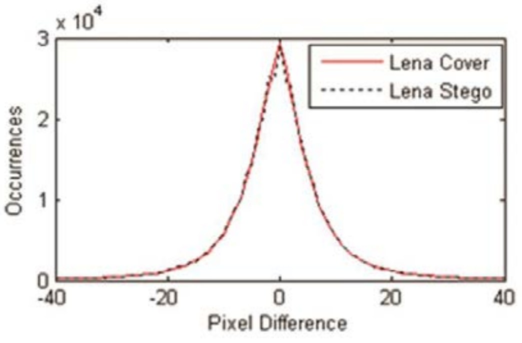
For any steganographic system the security analysis is a major concern. The pixel difference histograms of the different schemes for Lena and Baboon images are shown in Fig.4 and Fig.5 respectively. The step effects in the proposed 1-bit GBS and 2-bit GBS schemes are lesser as compared to Wu & Tsai and Tseng & Leng's PVD schemes. In 2-bit LSB substitution scheme the step effect in pixel difference histogram is more as compared to the 2-bit GBS scheme. This proves that the security in the said 2-bit GBS scheme is improved. As per the security is concerned, the proposed schemes do well as compared to LSB substitution, Wu & Tsai's and Tseng & Leng's schemes.



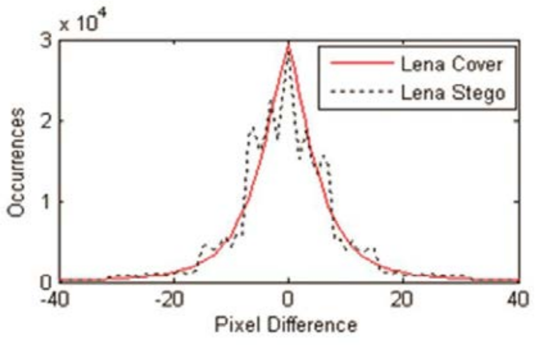
(a) 1-bit LSB



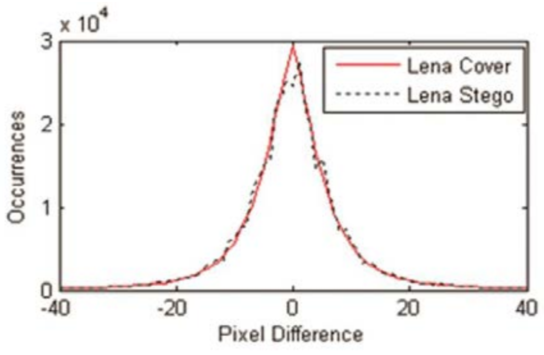
(d) 2-bit GBS



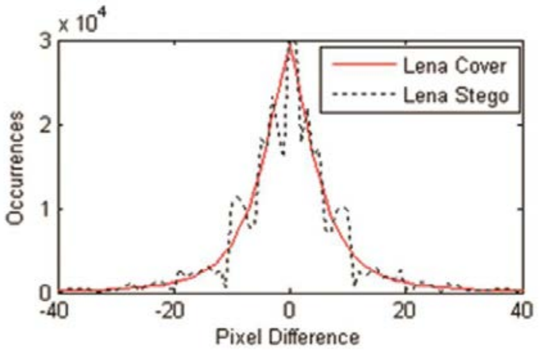
(b) 1-bit GBS



(e) Wu & Tsai's PVD

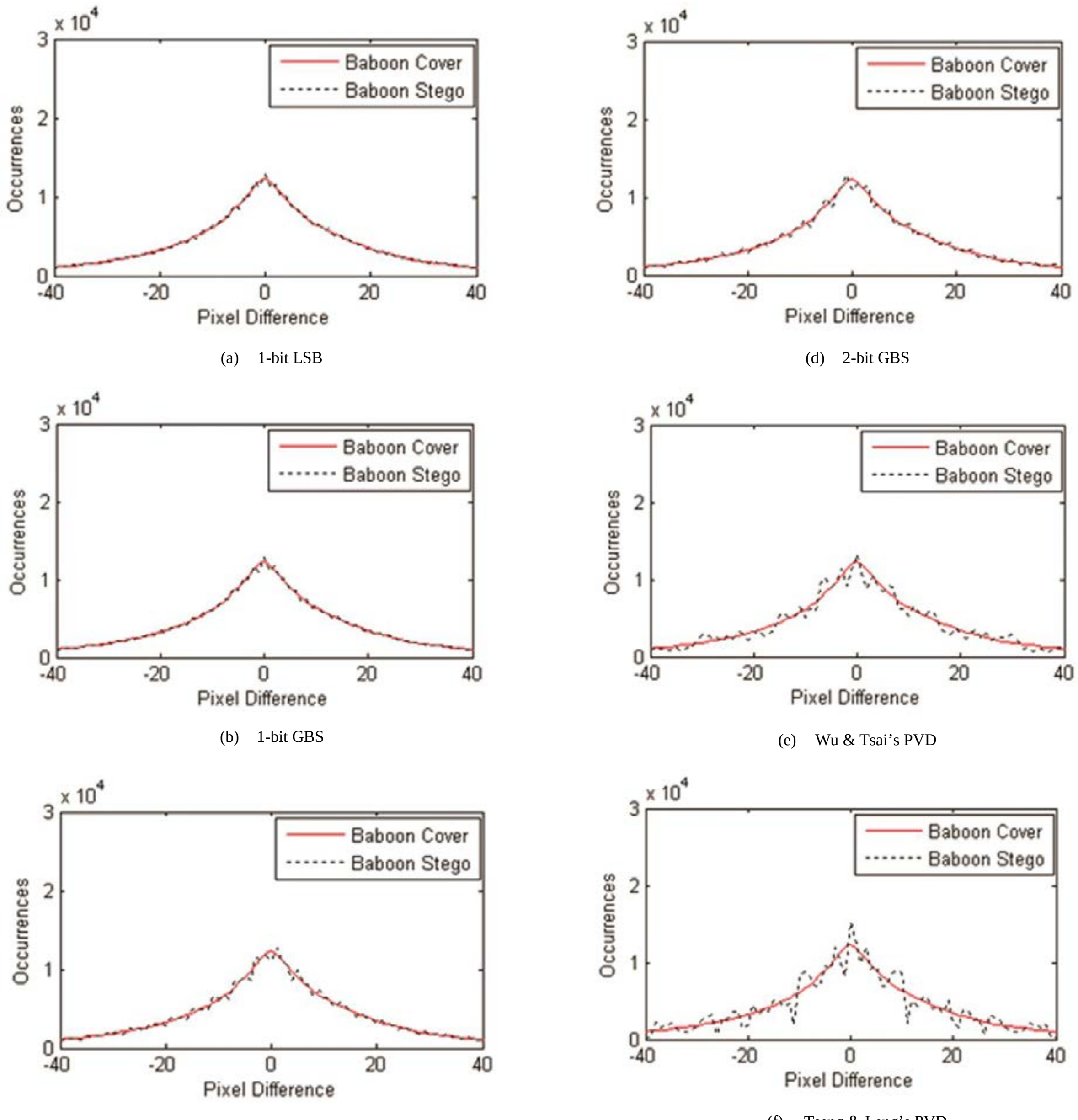


(c) 2-bit LSB



(f) Tseng & Leng's PVD

Fig.4 Pixel difference histograms for Lena image in different schemes



(a) 1-bit LSB

(b) 1-bit GBS

(c) 2-bit LSB

(d) 2-bit GBS

(e) Wu & Tsai’s PVD

(f) Tseng & Leng’s PVD

Fig.5 Pixel difference histograms for Baboon image in different schemes

5. Conclusion

The steganography methods presented in this paper are not direct LSB substitution. This type of group of bits substitution is seldom used for data hiding. The observed peak signal-to-noise ratio values are acceptable and the stego-images do not show any identifiable marks. The results are compared with traditional LSB substitution and the PVD schemes. The various conditions used in embedding procedure indicate the group of bits to be substituted and is variable for different pixels. This raises the security to a greater level. Although in 1-bit GBS scheme the PSNR is almost same as that of 1-bit LSB substitution scheme, but the PSNR of 2-bit GBS is highly improved compared to that of 2-bit LSB substitution scheme, Wu & Tsai's PVD scheme and Tseng & Leng's PVD scheme. For data extraction, no other extra information is needed and original image is also not required.

References

1. Cheddad A, Condell J, Curran K, Kevitt PM. Digital Image Steganography: Survey and Analysis of Current Methods. Signal Processing. 2010, 90:727-752.
2. Martin A, Sapiro G, Seroussi G. Is image steganography Natural?. IEEE Transactions on Image Processing. 2005, 14(12):2040-2050.
3. Mathkour H, Assassa GMR, Muharib AA, Kiady I. A novel approach for hiding messages in images. In: Proceedings of International Conference on Signal Acquisition and Processing. 2009, p.89-93.
4. Swain G, Lenka SK. A hybrid approach to steganography-embedding at darkest and brightest pixels. In: Proceedings of International Conference on Communication and Computational Intelligence. 2010, p.529-534.
5. Juneja M, Sandhu PS. Designing of robust image steganography technique based on LSB insertion and encryption. In: Proceedings of International Conference on Advances in Recent Technologies in Communication and Computing. 2009, p.302-305.
6. Swain G, Lenka SK. LSB array based image steganography technique by exploring the four least significant bits. Communications in Computer and Information Science. 2012, 270(2):479-488.
7. Swain G, Lenka SK. A novel steganography technique by mapping words with LSB array. International Journal of Signal and Imaging Systems Engineering. 2015, 8(1/2):115-122.
8. He J, Tang S, Wu T. An adaptive steganography based on depth-varying embedding. In: Proceedings of Congress on Image and Signal Processing. 2008, p.660-663.
9. Jain YK, Ahirwal RR. A novel image steganography method with adaptive number of least significant bits modification based on private stego-keys. International Journal of Computer Science and Security. 2010, 4(1):40-49.
10. Swain G, Lenka SK. A Technique for secret communication by using a new block cipher with dynamic steganography. International Journal of Security and Its Applications. 2012, 6(2):1-12.
11. Wu DC, Tsai WH. A steganographic method for images by pixel-value differencing. Pattern Recognition Letters. 2003, 24:1613-1626.
12. Lee YP, Lee JC, Chen WK, Chang KC, Su IJ, Chang CP. High-payload image hiding with quality recovery using tri-way pixel-value differencing. Information Sciences. 2012, 191:214-225.
13. Chang CC, Tseng HW. A steganographic method for digital images using side match. Pattern Recognition Letters. 2004, 25:1431-1437.
14. Swain G, Lenka SK. Steganography using two sided, three sided and four sided side match methods. CSI Transactions on ICT. 2013, 1(2):127-133.
15. Tseng HW, Leng HS. A steganographic method based on pixel-value differencing and the perfect square number. Journal of Applied Mathematics. 2013, Article ID 189706.
16. Swain G, Lenka SK. Classification of spatial domain image steganography techniques: a study. International Journal of Computer Science & Engineering Technology. 2014, 5(3):219-232.
17. Khodaei M, Faez K. New adaptive steganographic method using least-significant-bit substitution and pixel-value differencing. IET Image processing. 2012, 6(6):677-686.
18. Shen SY, Huang LH. A data hiding scheme using pixel value differencing and improving exploiting modification directions. Computers & Security. 2015, 48:131-141.