Enhancing the Security and Quality of LSB based Image Steganography

Nadeem Akhtar, Pragati Johri, Shahbaaz Khan
Department of Computer Engg, Zakir Husain College of Engg & Tech
Aligarh Muslim University
Aligarh, India
{nadeemalakhtar,johri.7405,saksplanet}@gmail.com

***Abstract*—This work is concerned with implementing Steganography for images, with an improvement in both security and image quality. The one that is implemented here is a variation of plain LSB (Least Significant Bit) algorithm. The stego-image quality is improved by using bit-inversion technique. In this technique, certain least significant bits of cover image are inverted after LSB steganography that co-occur with some pattern of other bits and that reduces the number of modified LSBs. Thus, less number of least significant bits of cover image is altered in comparison to plain LSB method, improving the PSNR of stego-image. By storing the bit patterns for which LSBs are inverted, message image can be obtained correctly. To improve the robustness of steganography, RC4 algorithm has been used to achieve the randomization in hiding message image bits into cover image pixels instead of storing them sequentially. This process randomly disperses the bits of the message in the cover image and thus, making it harder for unauthorized people to extract the original message. The proposed method shows good enhancement to Least Significant Bit technique in consideration to security as well as image quality. (Abstract)**

Keywords-steganography; RC4; LSB; security; image quality (key words)

I. INTRODUCTION

Steganography provides secrecy of text or images to prevent them from attackers. Steganography embed the message image in a cover image and changes its properties. Steganography provides secret communication so that intended hacker or attacker unable to detect the presence of message. To prevent the detection of secret messages is the major art of steganography. Steganography, derived from Greek, literally means "covered writing." It includes a vast array of secret communications methods that conceal the message's very existence. These methods include invisible inks, microdots, character arrangement, digital signatures, covert channels, and spread spectrum [1].

The basic concept is that it has a cover object that is used to cover the original message image, a host object that is the message or main image which is to be transmitted, a stego-key which is used to hide the message image into cover image, and the steganography algorithm to carry out the required object. The output is an image called stego-image which has the message image inside it, hidden. This stego image is then sent to the receiver where the receiver retrieves the message image by applying the de-steganography (Fig. 1 and Fig. 2).

Figure 1. Steganography at senders side

Figure 2. Steganography at receiver side

The advantages of Least-Significant-Bit (LSB) steganographic data embedding are that it is simple to understand, easy to implement, and it produces stego-image that is almost similar to cover image and its visual infidelity cannot be judged by naked eyes. Several steganography methods based on LSB have been proposed and implemented [2][3][4].

A good technique of image steganography aims at three aspects. First one is capacity (the maximum data that can be stored inside cover image). Second one is the imperceptibility (the visual quality of stego-image after data hiding) and the last is robustness [5]. The LSB based technique is good at imperceptibility but hidden data capacity is low because only one bit per pixel is used for data hiding. Simple LSB technique is also not robust because secret message can be retrieved very easily once it is detected that the image has some hidden secret data by retrieving the LSBs.

Several staganalytic methods have been developed to detect the hidden message from the image in communication. One of the earliest methods is chi-square test [6], which performs statistical analysis to identify the message. By reducing the size

978-0-7695-5069-5/13 $26.00 © 2013 IEEE
DOI 10.1109/CICN.2013.85

of message, detection risk in this attack can also be reduced. In [7], authors have proposed technique known as RS staganalysis which can estimate message size efficiently when the message is embedded randomly. In [8], a powerful staganalysis method is proposed, called SPA, which uses sample pair analysis to detect the message length.

In this paper, we present a LSB based steganography method which is more secure and robust than plain LSB method. Rather than storing the message bits sequentially, they are stored in a random order generated by RC4 algorithm which uses a stago-key shared by both sender and receiver. After that staganalysis is performed on the stago-image to analyze the bit patterns of second and third LSBs that co-occur with LSB. Based on this analysis, LSB of those bytes may be inverted which co-occurs with a specific bit pattern, which improves the PSNR of stago-image and also makes the task of staganalysis difficult.

Next section describes the method proposed. Section III describes the experiments and results. In section IV, conclusion is discussed.

II. IMPLEMENTATION

A. Classical LSB Algorithm:

Image represents various light intensities that represents pixels and pixels represents a number, so image is an array utilizing 256 colors(common images) having different pixels values at different locations.

Digital images are typically stored in either 24-bit or 8-bit per pixel. 24-bit images are sometimes known as true color images. Obviously, a 24-bit image provides more space for hiding information; however, 24-bit images are generally large and not that common. A 24-bit image 1024 pixels wide by 768 pixels high would have a size in excess of 2 megabytes. As such, large files would attract attention were they to be transmitted across a network or the Internet.

Generally 8 bits images are used to hide information such as GIF files represented as a single byte for each pixel. Now, each pixel can correspond to 256 colors. It can be said that pixel value ranges from 0 to 255 and the selected pixels indicates certain colors on the screen.

The technique for classical least significant bit implies the replacement of LSB's of cover images with message image in order to hide information by altering cover image. Since LSB is replaced there is no affect on cover image and hence intended user will not get the idea that some message is hidden behind the image.

Here given a example to show LSB replacement to hide letter 'A' behind cover image.

Cover Image:

00100111 11101001 11001000 00100111 11001000
11101001 11001000 00100111

Message Image:

10000001

Steganographed Image:

00100111 11101000 11001000 00100110 11001000
11101000 11001000 00100111

The bold bits represent the changed bits. LSB insertion requires on average that only half the bits in an image be changed.

There are several variations in this approach as instead of replacing one bit, two or more bits of cover image can be replaced so that large amount of message can be hidden in a single image but it can deteriorate quality of cover image as the number of replaced LSB's are increased [9] [10].

The LSB replacement allows simply replacing the information behind cover image directly and changing a single bit of a pixel does not cause perceptible difference in image quality [11]. So, the change in amplitude is very small and this allows high perceptual transparency of LSB.

The disadvantages included in the LSB approach is the size of cover image required for a particular message image that is for a certain capacity of message cover image required is 8 times thus increasing the bandwidth to send the image [12]. Another big disadvantage is that if an attacker suspects that some information is hidden behind the cover image, He can easily extract information by just collecting LSBs of stego image. For these criteria, this method is not successful.

B. Randomization using RC4 algorithm:

This classical LSB implementation is most easy and direct method to hide information in a cover image. The message is embedded with sequence-mapping technique in the pixels of a cover-image. Although LSB hides the message in such way that the humans do not perceive it, it is still possible for the opponent to retrieve the message due to the simplicity of the technique. Therefore, malicious people can easily try to extract the message from the beginning of the image if they are suspicious that there exists secret information that was embedded in the image.

Therefore, this method is proposed to improve the security of LSB scheme. This method overcomes the sequence-mapping problem by embedding the message and stego key into a set of random pixels, which are scattered over the entire cover image. The bits of the secret message are embedded in pixels of the cover image that are generated by RC4 algorithm.

The system enhanced the LSB technique by randomly dispersing the bits of the message in the image and thus making it harder for unauthorized people to extract the original message. The proposed algorithm provides a stego-key that will be used during the embedding and extracting of the message.

RC4 algorithm is invented by Ron Rivest ."RC" refers to "Ron's Code" or "Rivest Cipher". It depends on a stream cipher

that generates key stream byte at a step. It uses an initial Key Stream Generator and Pseudo Random Generator algorithms. These are specified below.

RC4 Initialization (Key Stream Initialization):

```
Array key contains N bytes of key
Array S always has a permutation of 0, 1,…size
    for i = 0 to size of message image
          S[i] = i
    K[i] = key [i (mod N)]
    Next i
     j = 0
      for i = 0 to 255
          j = (j + S[i] + K[i]) (mod size)
            Swap(S[i], S[j])
              Next i
         i = j = 0
```

RC4 Key stream:
For each key stream byte, swap elements of array S and select a byte from the array:
i = (i + 1) (mod 256)
j = (j + S[i]) (mod 256)
Swap(S[i], S[j])
t = (S[i] + S[j]) (mod 256)
Key_Stream_Byte = S[t]

Advantages:
RC4 is efficient in software. It is simple and elegant. RC4 is used in lots of places: SSL, WEP, etc. it is the most popular stream cipher in existence. It is used mainly for random stream generation.

C. RC4 algorithm with LSB replacement

Following steps describes how RC4 algorithm is used to generate random cover image pixels. To generate the random sequence, a stego-key is used, which can be stored into the cover image or supplied to the receiver separately.

1) Step 1: Calculate the size of message Image.
2) Step 2: Select the cover image at least 8 times of the size message image.
3) Step 3: Pixilate both the images and convert to binary form.
4) Step 4: Generate array of cover image locations for selected stego-key using RC4 algorithm.
5) Step 5: Replace the LSBs of cover image pixels in the sequence generated in step 4, with message image bits

It is very hard to know the sequence of cover image pixels which is used to hide the message bits for an observer in absence of stego-key. Thus this method improves the steganography security in comparison to the plain LSB method, in which cover image pixels are replaced sequentially with message bits.

D. PSNR Improvement:

We have employed a novel bit-inversion technique to improve the stego-image quality. To understand the technique, we consider the following example.

Four message bits 1011 are to be hidden into four cover image pixels 10001100, 10101101, 10101011 and 10101101.

After plain LSB steganography, stego-image pixels are 10001101, 10101100, 10101011 and 10101101. Two pixels i.e. first and second of cover image have changed. Now, we see that the second and third LSB of three cover image pixels are 0 and 1 respectively. For two of these three pixels, LSB has changed. If we invert the LSB of these three pixels, cover image pixels will be 10001100, 10101101, 10101011 and 10101100. Now, there is only one pixel of stego-image which differs from cover image i.e. the last one. Thus, the PSNR would increase improving the quality of stego-image. For correct de-steganography, we need to store the fact that we have inverted the LSBs of those pixels in which second and third LSB are 0 and 1 respectively.

If we consider two bits, there are four (00, 10, 10, 11) possible combinations. For each of the combination, stego-image is analyzed to find the number of pixels of first type i.e. whose LSB has changed and second type i.e. whose has not changed. If the number of pixels of first type is greater than the number of second type pixels, we invert the LSB of first type pixels. In this way, less number of pixels of cover image would be modified. The totals pixel benefit would be equal to the difference between the number of first and second type pixels.

III. RESULTS AND ANALYSIS

We analyze the temporary stego-image generated by simple LSB method, considering two bits (2^{nd} and 3^{rd} LSB). We used a two 1024*1024 cover images baboon Fig. 4 (a) and JuliaSet Fig. 4 (b); and six 256*256 message images Fig.5 (a)-(f). Table I, II and III show the analysis and bit inversion for the message images Crods, TestPat and Laser respectively [13].

TABLE I. STATISTICS AND DECISION FOR CRODS IMAGE

Bit pattern (3^{rd}, 2^{nd} LSB)	Changed Bits	Not Changed Bits	Invert	Changed Bits in Final Image
Cover Image: Baboon				
0 0	110787	73293	YES	73293
0 1	64053	139729	NO	64053
1 0	35126	32772	YES	32772
1 1	33325	35203	NO	33325
Cover Image: JuliaSet				
0 0	122158	73095	YES	73095
0 1	31354	53239	NO	31354
1 0	80392	54662	YES	54662
1 1	49793	59595	NO	49793

TABLE II. STATISTICS AND DECISION FOR TESTPAT IMAGE

Bit pattern (3rd, 2nd LSB)	Changed Bits	Not Changed Bits	Invert	Changed Bits in Final Image
Cover Image: Baboon				
0 0	62249	121831	NO	62249
0 1	161778	42004	YES	42004
1 0	32123	35775	NO	32123
1 1	35915	32613	YES	32613
Cover Image: JuliaSet				
0 0	48943	146310	NO	48943
0 1	58448	26145	YES	26145
1 0	46006	89048	NO	46006
1 1	62348	47040	YES	47040

For the crods image when embedded into Baboon, in Table I, number of unchanged bits is much less than the number of changed bits for bit pattern 00, so inversion is performed. Number of unchanged bits is also less than the number of changed bits for bit pattern 10; inversion is also performed for this case too. Number of unchanged bits is not less than the number of changed bits for other bit patterns. Before inversion, the number of stego-pixels which differs from the cover image pixels is 243291. After inversion is performed, the number of stego-pixels which differs from cover image pixels is 203443. A pixel-benefit of 39848 pixels is achieved, increasing the PSNR by 0.78. When crods image is embedded into JuliaSet cover image, a large pixel benefit of 74793 pixels is achieved, increasing the PSNR by 1.33.

For the TestPat message image when embedded into Baboon cover image, in Table II, number of unchanged bits is much less than the number of changed bits for bit pattern 01, so inversion is performed. Number of unchanged bits is also less than the number of changed bits for bit pattern 11; inversion is also performed for this case too. Number of unchanged bits is not less than the number of changed bits for other bit patterns. Before inversion, the number of stego-pixels which differs from the cover image pixels is 292065. After inversion is performed, the number of stego-pixels which differs from cover image pixels is 168989. A large pixel-benefit of 123076 pixels is achieved, increasing the PSNR greatly by 2.37. When TestPat image is embedded into JuliaSet cover image, a large pixel benefit of 47611 pixels is achieved, increasing the PSNR by 1.09. Table III shows the results for Laser image.

TABLE III. STATISTICS AND DECISION FOR TESTPAT IMAGE

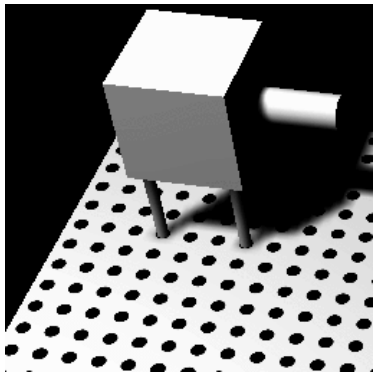
Bit pattern (3rd, 2nd LSB)	Changed Bits	Not Changed Bits	Invert	Changed Bits in Final Image
Cover Image: Baboon				
0 0	103397	80683	YES	80683
0 1	78443	125339	NO	78443
1 0	35211	32687	YES	32687
1 1	33523	35005	NO	33523
Cover Image: JuliaSet				
0 0	119372	75881	YES	75881
0 1	35815	48778	NO	35815
1 0	81257	53797	YES	53797
1 1	52330	57058	NO	52330

Table IV shows the improvement in PSNR for all the six message images when bit-inversion technique is used to embed them into baboon and JuliaSet cover images. It can be analyzed from the table III that the improvement in PSNR is small for some images and large for some other images. Actually, it depends on the distribution of bits in cover and message images which is totally random. For some message and cover image, the improvement in PSNR may be zero. For example, when crods and fishingboat images are embedded into lego cover image Fig. 6, No improvement is achievement whereas when TestPat is embedded into lego, improvement in PSNR is very large i.e. 4.82, getting a heavy pixel-benefit of 264306.

The choice of cover image for a message image depends on how much improvement in PSNR it provides. For example, JuliaSet is better choice for cover image for crods message image because it provides better improvement in PSNR in comparison to Baboon. For TestPat message image, Baboon is better choice for cover image.

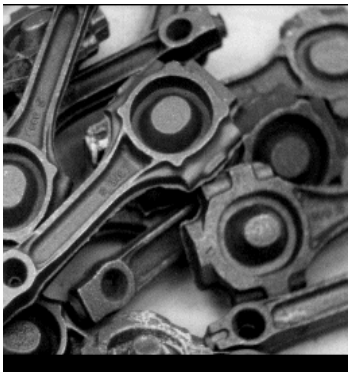
TABLE IV. IMPROVEMENT IN PSNR FOR THE TWO MESSAGE IMAGES

Message Images	PSNR			
	Cover Image: Baboon		Cover Image: JuliaSet	
	Before Inversion	After Inversion	Before Inversion	After Inversion
Laser	54.34	54.80	53.73	54.95
3things	54.11	54.25	54.15	54.22
Crods	54.47	55.25	53.80	55.13
Mandrill	54.20	54.32	54.07	54.30
Fishingboat	54.30	54.62	53.96	54.60
TestPat	53.68	56.05	54.99	56.08

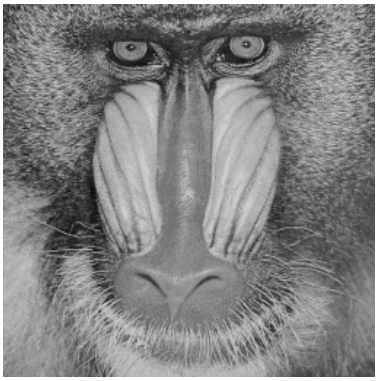


(a)

(b)

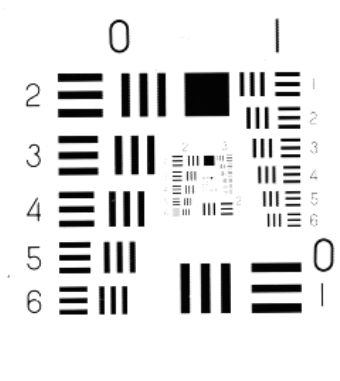


(c)



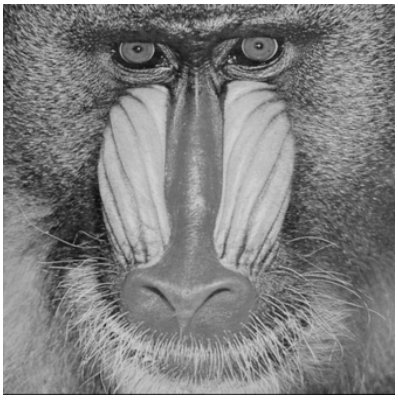
(d)

(e)

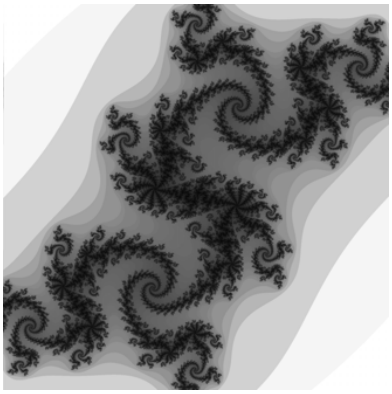


(f)

Figure 3. Message images (a) Laser (b) 3 things (c) Crods (d) Madrill (e) Fishing boat (f) TestPat



(a)



(b)

(c)

Figure 4. 1024*1024 Cover Images (a) baboon (b) JuliaSet (c) Lego

IV. CONCLUSION:

The proposed bit inversion method enhances the stego-image quality. The enhancement in PSNR is not bounded. The improvement in PSNR may be very large for some image as in the case of TestPat image and for some other image, it may be small as for 3things image. For given a message image, a set of cover image can be considered and that cover image is selected for which the improvement is largest.

For security enhancement, the order of selected cover-image-pixel for embedding the message bits depends on the random numbers generated by the RC4 using a shared key. Although the third party could determine if the message bits are embedded, he would have a difficulty to recover it because the message bits are embedded in a random order. Moreover, some of the LSBs have been inverted; it will misguide the staganalysis process and make the message recovery difficult.

The bit inversion method and use of random locations together makes the steganography better by improving its security, image quality and robustness. Classical LSB algorithm when used with RC4 and bit inversion provides user with multilayer of protection so that intended opponent find difficulty to trace hidden information in cover image.

In future work, other bit combinations of cover image pixels can be considered. There are 21 ($^7C_2$) bit combinations of two bits in a pixel leaving the LSB. Further, more bits of cover image pixels can be considered for analysis. For example, three bits can be considered which provides 8 different bit patterns improving the possibility of greater enhancement in PSNR.

REFERENCES

[1] Cheddad, J. Condell, K. Curran, & P. Kevitt, (2010). Digital image Steganography- survey and analysis of current methods. Signal Processing, 90, 727–752.

[2] C. K. Chan, L. M. Cheng, “Hiding data in image by simple LSB substitution”, pattern recognition, Vol. 37, No. 3, 2004, pp. 469-474.

[3] R. Z. Wang, C. F. Lin and I. C. Lin, “Image Hiding by LSB substitution and genetic algorithm”, Pattern Recognition, Vol. 34, No. 3, pp. 671-683, 2001.

[4] D. Sandipan, A. Ajith, S. Sugata, An LSB Data Hiding Technique Using Prime Numbers, The Third International Symposium on Information Assurance and Security, Manchester, UK, IEEE CS press, 2007

[5] C. Kessler. (2001). Steganography: Hiding Data within Data. An edited version of this paper with the title "Hiding Data in Data". Windows & .NET Magazine.http://www.garykessler.net/library/steganography.html

[6] Westfeld, A., Pfitzmann, A.: Attacks on Steganographic Systems. In: 3rd International Workshop on Information Hiding (IHW 99), pp. 61-76. (1999)

[7] Fridrich, J., Goljan, M., Du, R.: Detecting LSB Steganography in Color and Gray Images. Magazine of IEEE Multimedia (Special Issue on Security), October-November, pp. 22-28. (2001)

[8] Dumitrescu, S., X. Wu and Z. Wang, Detection of LSB steganography via sample pair analysis, Springer LNCS, vol.2578, pp.355-372, 2003.

[9] C.C. Thien, J.C. Lin, A simple and high-hiding capacity method for hiding digit-bydigit data in images based on modulus function, Pattern Recognition 36 (2003) 2875–2881.

[10] C.C. Chang, J.Y. Hsiao, C.S. Chen, Finding optimal Least-Significant-Bit substitution in image hiding by dynamic programming strategy, Pattern Recognition 36 (2003) 1583–1595.

[11] R. Chandramouli, N. Memon, “Analysis of LSB Based Image Steganography Techniques”, IEEE pp. 1019-1022, 2001.

[12] C. Cachin, “An Information-Theoretic Model for Steganography”, in proceeding 2nd Information Hiding Workshop, vol. 1525, pp. 306-318, 1998.

[13] http://sipi.usc.edu/database/