A New Steganography Method for Gray Level Images using Parity Checker

Rajkumar
Assistant Professor
Deptt. Of Comp. Sc. & Engg.
UIET, MD University, Rohtak-124001 (INDIA)

Rahul Rishi
Associate Professor
Department of Comp. Sci. & Engg,
TITS, Bhiwani-127021 (INDIA)

Sudhir Batra
Professor
Department of Mathematics
DCRUST, Murthal-131039 (INDIA)

ABSTRACT

In the present study, we present a new steganography technique for hiding data in images using parity checker. This method uses the concept of odd and even parity for insertion and retrieval of message. This method is an improvement over earlier methods like least significant bit method and 6^{th}, 7^{th} bit method [1] for hiding information in images. Our method retains the advantages of above methods but discards the disadvantages of above methods and provides us the optimal results.

Keywords

Parity Checker, Even Parity, Odd Parity, Pseudo Random Number Generator, Steganalysis

1. INTRODUCTION

Steganography is an art and science of hiding information in some cover media. The term originated from Greek roots that literally means "covered writing" [2]. Steganography has been used since ancient times, for example people practiced it by etching messages in wooden tablets and covered with wax. They used tattooing a shaved messenger's head, letting his hair grow back and then saving it again when he arrived at his contact point to reveal the message. Different types of stenographic techniques have been used that employ invisible inks, microdots, character etc. [3].

Digital steganography uses the digital objects such as image, video, music or any other computer file for hiding the data. The idea was first given by Simmons in 1983[4]. Steganography is different from cryptography; the latter is about concealing the content of message whereas former is about concealing the existence of message itself [5].

A most popular and oldest technique for hiding data in digital image is the Least Significant Bit Method Technique [6]. One of the major disadvantages associated with LSB techniques is that the hidden message can be destroyed by the intruder by changing the LSB of all image pixels. In this way, hidden message can be destroyed but the change in image quality is in the range of +1 to -1 at each pixel position. The algorithm designed by Parvinder *et al.* [7] removes the disadvantage of least significant bit method technique by using 6^{th} and 7^{th} bit of pixel value for message insertion. But the chance of message insertion at pseudo random location at first instance using this algorithm (as described by Parvinder *et al.*) is only 49% which in itself is a disadvantage. In spite of all, one common disadvantage of above mentioned algorithms for insertion and retrieval of message is the stress on specific bits (like 6^{th} and 7^{th} bit etc.) which makes the steganalysis very easy. In the present study, we used the concept of even and odd parity and stressed on whole pixel rather than specific bits by using the parity checker. In our algorithm, the chance of message insertion at the pseudo random location at first instance is 98.82% as compared with *Parvinder et al.* that provides only 49% chance. This is near about optimal solution and an improvement over previous methods.

2. DESCRIPTION OF PROPOSED METHOD

In our method, we used the concept of even and odd parity by using the parity checker. As we already know that even parity means that the pixel value contains even number of 1's and odd parity means that the pixel value contains odd number of 1's. We inserted '0' at a pixel value where pixel value had odd parity and if odd parity is not present over there than we made the odd parity by adding or subtracting '1' to the pixel value. Similarly, we inserted '1' at a pixel value if it had even parity. In case, if even parity is not present at that location then we made even parity over that location by adding or subtracting '1'. In this way we can insert '0' or '1' at any location. The insertion process is shown in figure 2 (a_1 & a_2).

For Retrieval of message, again we used the parity checker. If odd parity is present at the selected location then '0' is message bit, else message bit is '1'. Retrieval process was repeated for all locations where message bits were hidden. In this way, we retrieved the message bits from all the locations where the message bit were inserted. The retrieval process is shown in Figure 2 (b)

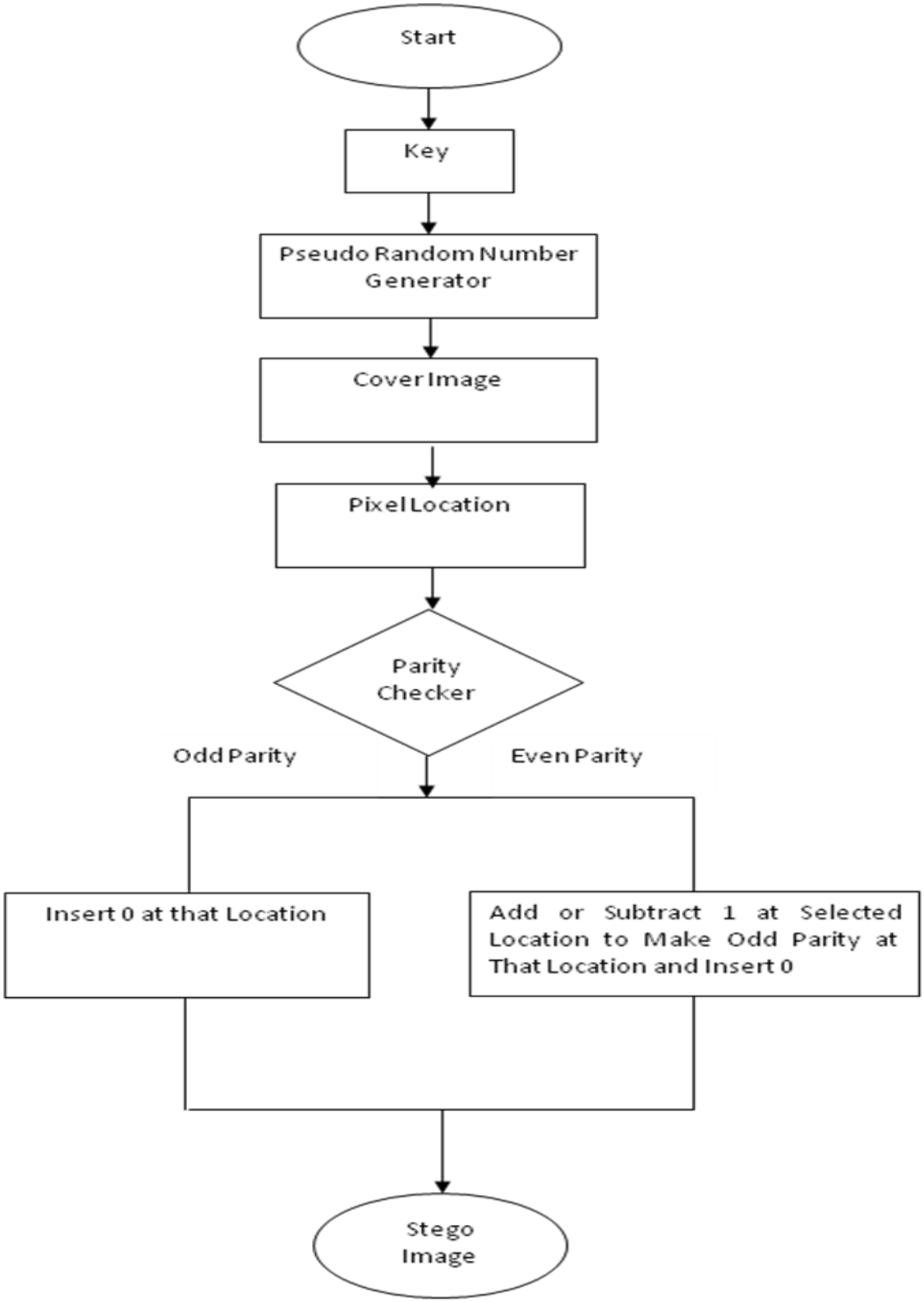

Figure 2(a_1): Flow chart depicting the Insertion of '0'

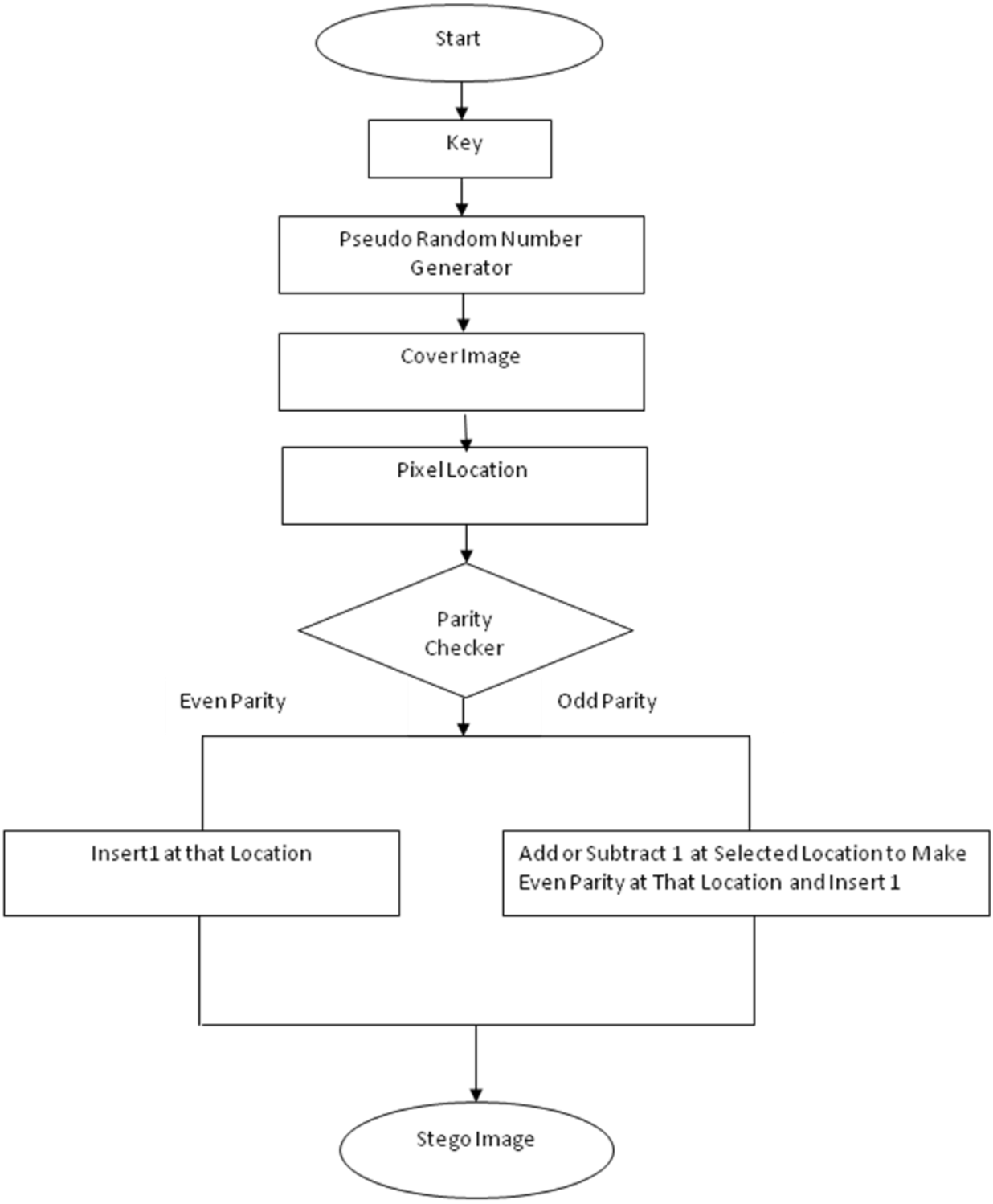

Figure 2(a_2): Flow chart depicting the Insertion of '1'

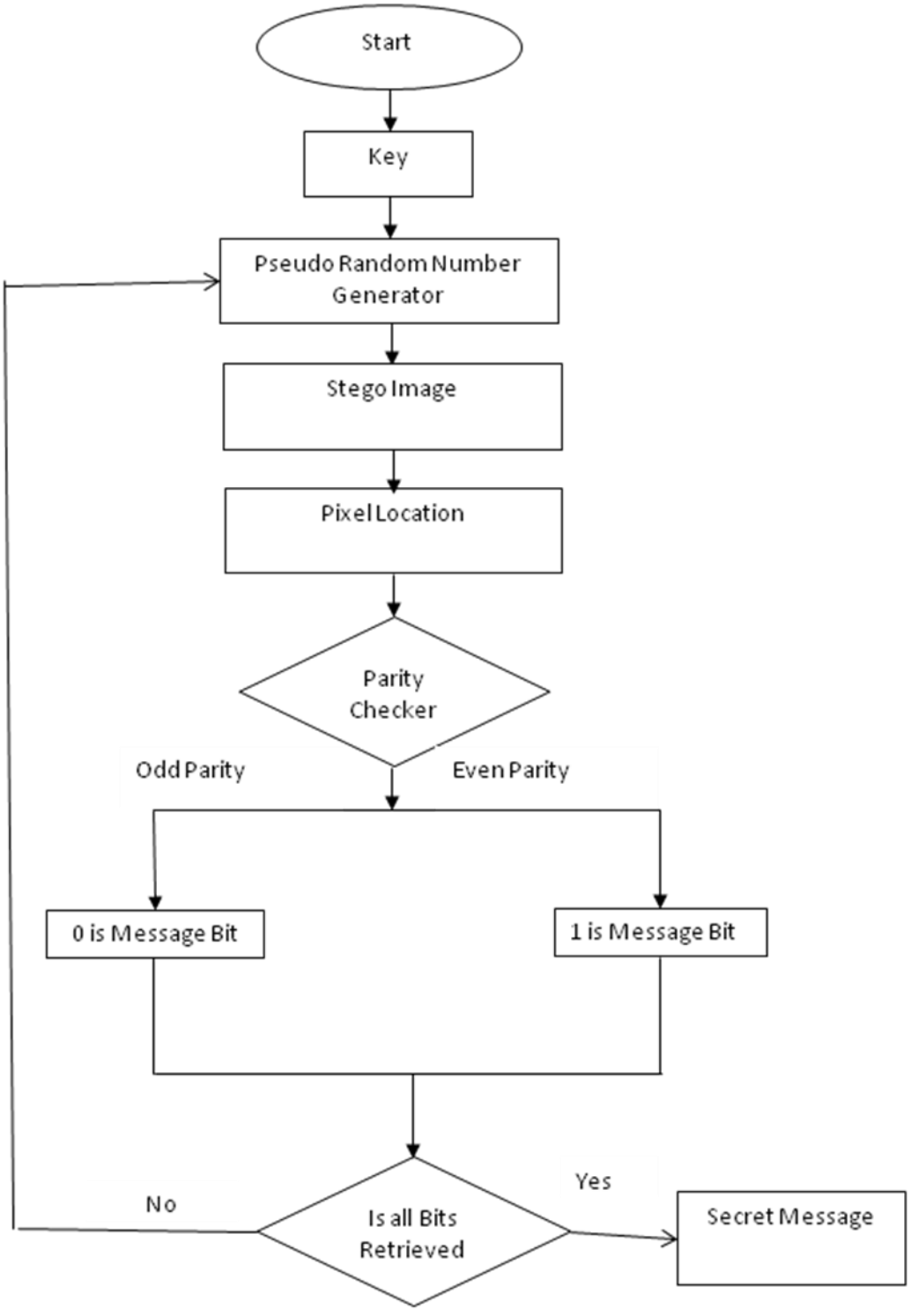

Figure 2(b): Flow chart depicting the retrieval process

3. ALGORITHM

3.1 Assumption

(i) Sender and recipient agree on the cover object in which message is supposed to be hidden.

(ii) Both sender and recipient agree on the same pseudo–random key to decide the random locations where message is to be inserted.

3.2 Insertion Algorithm

(i) Find the pseudo-random location (L) in cover image from secret key to insert the message bit. (For detail see [7] and [8]).

(ii) Check whether at location (L); pixel value is 00000000 OR 11111111. If yes, ignore this location and go to step (i). Here, we are ignoring these boundary values because the change may be +2 or -2 in pixel values which is to be avoided.

(iii) If we want to insert 0 then go to step (iv) else go to step (v).

(iv) (a) Check whether at location (L) there exists odd parity. If yes, insert 0 at location 'L', go to End. If No, go to step (b)

(b) Make the odd parity by adding or subtracting 1 and insert 0. Go to END

(v) (a) Check whether at location 'L' there exists even parity. If yes, insert 1 at location (L) and go to END. If No, go to step (b).

(b) Make the even parity by adding or subtracting 1, insert 1 and go to END.

(vi) END

3.3 Retrieval Algorithm

(i) Trace out the location (L) from the same secret key as used for insertion of message.

(ii) Pixel value is equal to 00000000 OR 11111111? If yes, then it is invalid address. Go to step (i)

(iii) Check whether at location (L):

(a) If there is odd parity then 0 is message bit.

(b) If there is even parity then 1 is message bit

(iv) END

4. CHANGE IN PIXEL DURING INSERTION OF MESSAGE

It is obvious from the above described algorithm that how various pixel values changed during insertion process. The corresponding change in pixel values while inserting 0 and 1 is shown in Table 1 and 2 respectively.

5. RESULTS AND CONCLUSION

5.1 The following results obtained from Table (I) and Table (II) tell us how our method is better than previous methods.

(i) The message bit will be inserted at the pseudo random location at first

chance = 506/512*100= 98.82%

(ii) Chance when message is inserted, no change in pixel value is required

= 254/506*100= 50.19%

5.2 The comparison of our method with 6^{th}, 7^{th} bit method for hiding the messages in images is shown in Table 3.

Table 1: Change in pixel value after insertion of '0'

Decimal Value	Pixel Value Before Insertion	Comment By Parity Checker	Pixel Value After Insertion an of '0'	Change in Pixel Value & Comment for Insertion of '0'
0	00000000	BV	00000000	NC, Invalid Location
1	00000001	Odd Parity	00000001	NC ,Insert
2	00000010	Odd Parity	00000010	NC, Insert
3	00000011	Even Parity	00000010	-1, Insert
4	00000100	Odd Parity	00000100	NC, Insert
5	00000101	Even Parity	00000100	-1, Insert
6	00000110	Even Parity	00000111	+1 Insert
7	00000111	Odd Parity	00000111	NC, Insert
8	00001000	Odd Parity	00001000	NC, Insert
9	00001001	Even Parity	00001000	-1, Insert
10	00001010	Even Parity	00001011	+1, Insert
11	00001011	Odd Parity	00001011	NC, Insert
12	00001100	Even Parity	00001011	-1, Insert

13	00001101	Odd Parity	00001101	NC, Insert
14	00001110	Odd Parity	00001110	NC, Insert
15	00001111	Even Parity	00001000	+1, Insert
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-
127	01111111	Odd Parity	01111111	NC, Insert
128	10000000	Odd Parity	10000000	NC, Insert
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-
254	11111110	Odd Parity	11111110	NC, Insert
255	111111111	Even Parity	11111111	NC, Invalid Location

BV= Boundary values

NC= No Change

Table 2: Change in pixel value after insertion of '1'

Decimal Value	Pixel Value Before Insertion	Comment by Parity checker	Pixel Value After Insertion of '1'	Change in Pixel value & Comment for insertion of
0	00000000	BV	00000000	NC,Invalid Location
1	00000001	Odd Parity	00000000	-1, Invalid Location
2	00000010	Odd Parity	00000011	+1, Insert
3	00000011	Even Parity	00000011	NC, Insert
4	00000100	Odd Parity	00000101	+1, Insert
5	00000101	Even Parity	00000101	NC, Insert
6	00000110	Even Parity	00000110	NC, Insert
7	00000111	Odd Parity	00000110	-1, Insert
8	00001000	Odd Parity	00001000	+1, Insert
9	00001001	Even Parity	00001000	NC, Insert
10	00001010	Even Parity	00001100	NC, Insert
11	00001011	Odd Parity	00001100	+1, Insert
12	00001100	Even Parity	00001111	NC, Insert
13	00001101	Odd Parity	0000111	-1, Insert
			1	
14	00001110	Odd Parity	00001110	+1, Insert
15	00001111	Even Parity	00001000	NC, Insert
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-
127	01111111	Odd Parity	01111110	-1,Insert
128	10000000	Odd Parity	10000001	+1,Insert
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-
254	111111110	Odd Parity	11111111	+1,Invalid location
255	111111111	Even Parity	11111111	NC,Invalid location

Table 3: Comparison between methods described by Rajkumar *et al.* and Parvinder *et al.*

Method	Message Bit Insertion at pseudorandom location at First Chance	No Change in Pixel Value when Message bit is inserted
Parvinder *et al.*	49%	50%
Rajkumar *et al.*	98.82%	50.19%

So, from the above discussion, we can say that our method is better than previous methods like 6th, 7th bit method for the following reasons:

(a) It provides greater chances i.e. 98.82% for message insertion.
(b) Change in image required is less than previous methods.
(c) Steganylysis is difficult in our method because our stress is a whole pixel rather than specific bits as used by previous methods.

6. REFERENCE

[1] Parvinder, Sudhir Batra and H R Sharma, "Evaluating the Performance of Message Hidden in First and Second Bit Plane, W SEAS Transaction on Information

Science and Technology, Vol. 2, No. 89, PP 1220-1222, Aug. 2005.

[2] A. Gutub, M. Fattani, "A Novel Arabic Text Stegnography Method Using Letter Points and Extension", WASET International Conference on Computer, Information and System Science and Engineering (ICCISSE), Vienna, Austria, May 25-27, 2007.

[3]. RJ Anderson, "Stretching the Limit of Stegnography", In Information Hiding, Springer Lecture Notes in Computer Science, Vol. 1174, pp 39-48, 1996.

[4]. G J Simmons, "The Prisoners Problem and the Subliminal Chaunell", Proceedings of cypto' 83, Plenum Press, pp 51-67, 1983.

[5] RJ Anderson, FAP Petitcolas, "On the Limits of Stegnography" IEE Journal on Selected Areas in Communications, Vol. 16 No. 4, PP 474-481, May, 1998.

[6]. Neil F Johnson, Sushil Jajodia, "Exploring Stenography: Seeing the Unseen", IEEE Computer, pp 26-34, Feb 1998.

[7]. E Franz, A Jerichow, S Moller, A Pfitznaun, I Stierand, "Computer Based Stegnography", Information Hiding, Springer Lecture Notes in Computer Science vol. 1174, pp 7-21, 1996.

[8] Yeuan-Kuen Lee, Ling – Hwei chen, "A secure Robust Image Stegnography Model", 10th National Conference on Information Security, Hualien, Taiwan, pp 275-284, May, 2000.