A Robust Method for Image Steganography based on Chaos Theory

Anoop Kumar Tiwari
Research Scholar
Dept of Computer Science
Banaras Hindu University
Varanasi, India

Ajay Rajpoot,
Ex-M.Sc. Student
Dept of Computer Science BHU
Varanasi, India

K. K. Shukla, Ph.D.
Professor
Dept of Computer Science and Engineering
IIT (BHU),
Varanasi, India

S. Karthikeyan, Ph.D.
Associate Professor
Dept of Computer Science
BHU
Varanasi, India

ABSTRACT
The Internet provides very economical real-time communications between computers and delivers services/products almost instantly to the users. The sensitive data transmitted through the internet must be secured as it is growingly susceptible to security related problems such as eavesdropping, malicious interventions etc. The steganography is an alternative to cryptographic techniques for secured transmission of data over the internet, where the secret message is hidden in some other innocuous communication so that only the rightful recipient can able to detect the presence of the secret message and extract it. Recently, many data embedding schemes have been proposed for achieving the robustness of this technique. However, most of the schemes lack to strike a tradeoff between the embedding capacity and the visual quality. In this paper, we have proposed a new method for hiding messages in a color image in the spatial domain based on chaos theory, which uses chaotic maps to embed data. This method is robust as well as has the higher payload capacity and compression resistant.

Keywords
Steganography, chaotic map, chaotic pixel selection Bifurcation diagram, Histogram, Logistic map

1. INTRODUCTION
In recent years the information and communication technology is advancing at tremendous pace which result in an easy access to the internet resources/services very economically to common man. The internet is increasing becoming more powerful and convenient medium for communication and supports many applications that require securing transmission of sensitive data. So there is a need for securing the data transmitted over the internet exclusively while communicating secret messages.

The issues of security and privacy have traditionally been dealt with using tools from cryptography that uses encrypting/ decrypting techniques. Messages can be appended with a message authentication code and encrypted so that only the rightful recipient can read them and verify their integrity and authenticity. But when intercepted, the encrypted messages can readily be identified as the sender and recipient are communicating secretly. So there is a need for some different approaches so that the secret communication can be hidden from the eyes of eavesdroppers and it can be prevented from being intervened [04,05]. Steganography[03] becomes an alternative tool where the secret messages are hidden in other innocuous-looking objects so that their very presence is not discovered. There are many major differences between cryptography and steganography, but the major advantage of steganography over cryptography is that the presence of the secret message is hidden and hence the communication does not attract the attackers or eavesdroppers towards itself. The Steganography is classified into different types [01,02], namely:

(i) Steganography in text (Text Steganography),

(ii) Steganography in images (Image Steganography),

(iii) Steganography in audio (Audio Steganography),

(iv) Steganography in video (Video Steganography),

(v) Steganography in TCP/IP Protocols etc.

Each of these types of steganography uses different techniques for embedding the secret message into the covert communication.

2. PREVIOUS WORKS
Recently chaos theory has been widely used in both steganography and cryptography. Bhavana et al. [06] had proposed a chaos based method for hiding text in images. The method explained about how a secret message can be hidden into an image using least significant bit insertion method along with chaos. They used henon map as a chaotic map to generate sequences of chaotic numbers. The sequences are converted to binary by taking their average as threshold value. Each bit of the converted message is XORed with this chaotic sequence. Each XORed bit is again XORed with the least significant bit of the pixel of the image selected as the cover image to hide that secret message's bit.

***Simple Illustration*:**
Consider, the binary value of I (ASCII-73), 01001001

Step 1: 01001001 is XORed with the binary chaotic sequence, say 10010001000100 and the output will be 11011000.

Step 2: Each of these bits is again XORed with LSB of individual pixel of the cover image. Suppose the 1st pixel of the image is 56 (which have an equivalent binary value 00111000). The LSB is `0' and is XORed with `1', which is obtained in the above sequence.

Step 3: The above step is repeated until all the bits are embedded in the image.

The desteganography is done to get back the secret message by following the reverse process followed for steganography.

Tayel, M et al [07] had proposed a new chaos based method for hiding multimedia data into cover image. The proposed steganography algorithm started with increasing the cover image's pixels from byte to word color capacity, and then distributes the secret-image's pixel randomly within the lower byte of the cover image's pixels using chaos distribution. The original image is separated from the received stego-image at the first stage of the receiver. The initial condition of the chaotic random sequence is used to collect the stego-image from the lower byte of the pixels. Then the hidden-image is reconstructed.

Bo Wang and JiuchaoFeng [08] had proposed a chaos based method for hiding a secret message in 'H.264' Standard Video Sequences. In this proposed method, the sender and the receiver initialize and synchronize the chaos generator 'C' with the same key, then two sequences can be obtained from C, one is S_i, (i == 1, 2, ...) for encryption, and another is G_k, (k == 1, 2,...) for embedding control. The plain text is encrypted into cryptograph c by using the algorithm EC ('), and c is embedded into the video sequence under the control of Gk by using the algorithm ES (.). Finally, d, containing secret message, can be obtained and is sent to the receiver through a channel. The receiver can recover m by the algorithm DEC(·) and DES(·) by using the same sequence S_i and G_k.

A Chaotic system is a deterministic, non-linear, dynamic system, but unpredictable. The unpredictability in chaotic system is due to insufficient knowledge about initial conditions. The chaotic systems are very sensitive to initial conditions. Any little differences in initial conditions yield widely diverging outcomes. In our proposed algorithm we also have exploited the critical dependency of chaotic systems towards their initial conditions.

3. PROPOSED METHODOLOGY

This method is primarily designed by keeping all the basic requirements of steganography (i.e. Capacity, Robustness and Imperceptibility) in mind and also hides the secret message in spatial domain so as to get larger capacity.

Our proposed methodology uses two chaotic systems (chaotic maps), one to chaotically select the image's data and the other to hide the message's bits. The chaotic system is used to select the positions of pixel to hide message's bits which leads to pixel's positions to hide message's bits becomes unpredictable and hence withstands against many steganalysis attacks. For better unpredictability, we use the first chaotic system to determine the initial condition of the second chaotic map and furthermore the initial condition of the first chaotic map is modified interactively at a fixed interval of 32 pixels, just to increase the chaocity. In this way the unpredictability and hence robustness of the methodology is increased.

The two chaotic maps used for selecting the position of pixels for embedding message's bits are Logistic map [9] and it is defined as follows:

$$X_{n+1} = 4\lambda x(1-x_n), \text{ such that } x_n \in (0, 1) \quad \text{and} \quad \lambda > 0 \qquad (1)$$

Where λ is a control parameter.

The figure 1 represents the bifurcation diagrams of the Logistic map with initial condition $x_0 = 0.5$ and $\lambda \in [.71, 1.0]$.

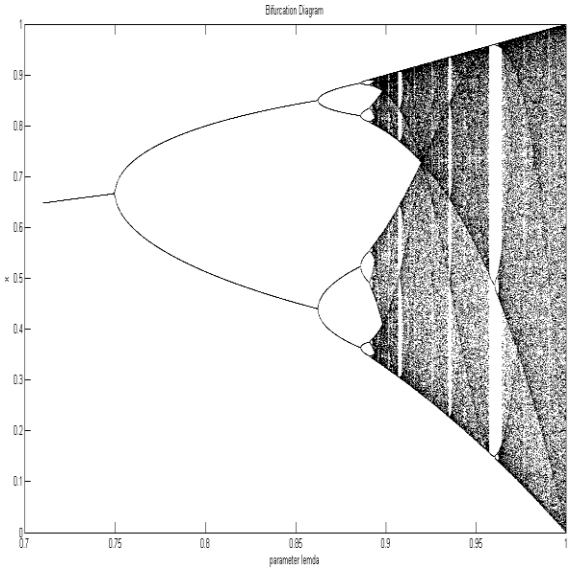

Fig 1: Bifurcation diagram

It is observed that the points on a line parallel to x-axis are very dense when parameter λ is in either the range [0.97675, 0.99000] or in the range [0.99075, 1.00000).

So, we have chosen λ = 0.9999 for the logistic maps. Accordingly the value of λ can be in any of the above intervals too, but the second interval is statistically found to be superior.

The two logistic maps, namely '*cm1*' and '*cm2*' generate two sequences, namely '*sequence1*' and '*sequence2*' of chaotic numbers. Two different 6_hexadecimal_digits keys '*key1*' and '*key2*' are used for these maps. Initially a 6-hexadecimal_digits value is assigned to the first key '*key1*' and the 'null' value is assigned to the second key '*key2*'. During, each iteration, we consider only 32 pixels and at the last iteration if there would not be a complete set of 32 pixels then we consider only the remaining pixels. These iterations repeat until there is no more bits of secret message remained to be embed. Here the first key 'key1' is modified by a predefined binary operation (here we choose bitwise XOR operation) performed on first and second key (i.e. 'key1' ←'key1'XOR 'key2'), then this modified key 'key1' is used to determine the initial condition of the map *'cm1*' that in turn used to generate a chaotic sequence 'sequence1' of 6 non-negative integers in [0, 15] which is assigned to the second key '*key2*'. The second key 'key2' is then used to determine the initial condition of the second chaotic map. The second chaotic map, then used to generate a sequence of n (where n = 'bits_to_be_hidden'<='24') non-negative integers in the interval [0, 31] that gives the positions of the pixels in the current segment of 32 pixels, and then 'bits_to_be_hidden' number of bits from the secret message are embed in the Luma-component of the pixels on these positions, the number 'bits_to_be_hidden' is determined for each iteration and it depends upon the image size and payload (size of the secret message). The method after successfully embedding the secret message within the image generates the stego-key to recover the message from the stego-image, only with this stego-key one can recover the embedded message.

Furthermore, the method is compression resistant i.e. withstands various compression techniques (e.g. JPEG, JPEG 2000, etc.) as it encodes the message's bits in the value of the Luma component of the selected pixel. Any compression method has to compromise with the characteristic of the

Human-visual-system (HVS). The HVS is much more sensitive to intensity than to color information; therefore any compression technique does not try to reduce the intensity information much. So the Luma (intensity) component of a pixel value is used to encode the message's bits in this method. In order to tolerate the changes made to the Luma components by the compression technique, we adjust the Luma value of the selected pixel according to certain compression tolerance coefficient that is also to be passed into the method.

Algorithm 1 Embedding the secret message

PROCEDURE Stegano_Hide

READ value of compression tolerance coefficient into 'l';

READ color cover image into image 'cover_img';

READ secret text message into array 'msg';

READ secret stego key into 'stego_key';

'original_key' ←'key1'←'stego_key';

'key2' ←'null';

'stego_img' ← 'cover_img';

'img_width'←width of the stego/cover image;

'img_height'←height of the stego/cover image;

'img_size'←'img_width' * 'img_height';

'nos'← FLOOR('img_size' / '32');

'rp' ←'img_size' mod '32';

'msg_size'←size of the secret message in Bytes;

'msg_bits'←'msg_size' * '8';

'bpp'←'msg_bits' / 'img_size';

IF 'bpp' >'0.75' THEN *ABORT THE PROCESS;*
'bit_count'←'0'; 'seg_count'←'0';

'bfps'←'0.0'; 'seg_len'←'32';

WHILE 'bit_count' <=('msg_bits'-'1') AND 'seg_count' <='nos' DO

IF 'seg_count='nos' THEN 'seg_len'←'rp';

'bits_at_hand' ←'seg_len'*'bpp' + 'bfps';

'abh' ←ROUND('bits_at_hand');

IF('abh' NOT= '0') THEN DO

'key1' ← 'key1' XOR 'key2';

'sequence1'←cm1.getChaoticSequence

('key1.numValue', 16,6); //'sequence1': 6 non-negative integers generated by CM 'cm1'.

'key2'←sequence1';
'sequence2'←cm2.getChaoticSequence('key2.numValue', 'seg_len', 'abh');

//'sequence2': 'abh' no. of non-negative integers generated by CM 'cm2'.

'next_bits' ←first 'abh' no. of bits of the remaining secret message;EMBED('next_bits', 'sequence2', 'actual_bits_to_be_hid');

END IF

'seg_count'←'seg_count'+'1';

'bfps'← 'bits_at_hand' - 'abh';

END WHILE

'rec_key_part2'← 'original_key' XOR 'msg_size';

'recovery_key' ←6-HexDigit 'original_key' + 6-HexDigit 'rec_key_part2'

END PROCEDURE

Algorithm 2 Embedding 'n' bits of secret message in the current segment given by 'seg_count'

FUNCTION EMBED('bits'[ ], 'pixl_pos'[ ], 'n')

GLOBAL VARIABLE: 'seg_count';

FOR 'i'='0' TO 'n'-'1' DO

'current_pixel_position'←'32'*'seg_count' + 'pixl_pos'['i'];'x'←'current_pixel_position' mod 'img_width';

'y'←'current_pixel_position' / 'img_width'

'Luma' ←Value of Luma component of the pixel of 'stego_img' at ('x', 'y');

IF 'bits'['i'] == '0' THEN //when bit to be hid is zero '0'

Adjust the value of 'Luma' such that

when divided by '4'*'l' yields remainder 'l';

ELSE //when bit to be hid is zero '1'

Adjust the value of 'Luma' such that

when divided by '4'*'l' yields remainder '3'*'l';

END IF

END FOR

END FUNCTION

Algorithm 3 Recovering the embedded message

PROCEDURE Stegano_Recover

READ value of compression tolerance coefficient into 'l';

READ color cover image into image 'stego_img';

READ secret stego key into 'recovery_key';

'key1' ←first 6 digits of 'recovery_key';

'key2' ← 'null';

'img_width'←width of the stego image;

'img_height'←height of the stego image;

'img_size'←'img_width' * 'img_height';

'nos'← FLOOR('img_size' / '32');

'rp'← 'img_size' mod '32';

'msg'['0'..'msg_size'-'1'] //byte array for holding the extracted message.

'msg_size'←decimalequivalant of last 6 digits of 'recovery_key';

'msg_bits'←'msg_size' * '8';

'bpp'←'msg_bits' / 'img_size';

IF 'bpp' >'0.75' THEN ABORT THE PROCESS;

'bit_count' ←'0'; 'seg_count'← '0';

'bfps'←'0.0'; 'seg_len'←'32';

WHILE 'bit_count' <=('msg_bits'-'1') AND 'seg_count' <='nos' DO

IF 'seg_count='nos' THEN 'seg_len'←'rp';

'bits_at_hand' ←'seg_len'*'bpp' + 'bfps';

'abh' ←ROUND('bits_at_hand');

IF('abh' NOT= '0') THEN DO

'key1'←'key1'XOR'key2';'

sequence1'←cm1.getChaoticSequence('key1.numValue', 16,6);//'sequence1': 6 non-negative integers generated by CM 'cm1'.

'key2' ←'sequence1';

'sequence2'←cm2.getChaoticSequence('key2.numValue', 'seg_len', 'abh');

//'sequence2': 'abh' no. of non-negative integers generated by CM 'cm2'.

'next_bits' ←EXTRACT('sequence2', 'actual_bits_to_be_hid');

//next 'abh' no. of bits extracted as the next bits of the remaining secret message;

APPEND 'next_bits' to 'msg';

END IF

'seg_count'←'seg_count'+'1';

'bfps'←'bits_at_hand' - 'abh';

END WHILE

SAVE the extracted message 'msg';

END PROCEDURE

Algorithm 4 Extracting next 'n' bits of secret message from the current segment given by 'seg_count'

FUNCTION EXTRACT('pixl_pos'[ ], 'n')

GLOBAL VARIABLE: 'seg_count';

bits['0'..'n-1']; // array for storing next 'n' bits of secret message.

FOR 'i'='0' TO 'n'-'1' DO

'current_pixel_position'←'32'*'seg_count'+ 'pixl_pos'['i'];

'x'←'current_pixel_position' mod 'img_width';

'y'←'current_pixel_position' / 'img_width'

'Luma' ←Value of Luma component of the pixel of 'stego_img' at ('x', 'y') ;

IF 'Luma' mod '4'*'l' = 'l' THEN 'bits'['i'] ←'0';

ELSE 'bits'['i'] ←'1';

END IF

END FOR

RETURN 'bits';

END FUNCTION

Description of the Input and Output for the above algorithms (For Embedding Process):

INPUT:

1. RGB Color Image 'cover_img';
2. Text message 'msg'['1'..'msg_size'] this is an array of characters representing each character as 8-bit (ASCII) integer (e.g. 'A' as 65 (8-bit int) or 01000001 (binary));
3. Value of compression tolerance coefficient 'l';
4. A 6-Hexadecimal-Digit secret key 'stego_key';

OUTPUT:

1. A Stego-image 'stego_img' (this will be the modified cover image with the secret text message embeded in it);
2. A 12-Hexadecimal_Digit secret key 'recovery_key' to recover the secret text message from the stego-image;

A small simulation of the embedding algorithm:

Here's a small simulation of the embedding process for the following parameters:

size of the input cover image, 'img_size' = '10*10' = '100'

number of 32-pixel segments 'nos' = QUOTIENT(100/32) = 3

number of remainig pixels 'rp' = REMAINDER(100/32) = 4

input message msg[] = "Rajpoot"

msg_size = 7 Byte

msg_bits = 7*8 = 56 bits

average bits to be hidden per pixel 'bpp' = 56/100 = 0.56 bits-per-pixel

initialize: 'bfps'=0.0, 'seg_count'=0

'bits_at_hand' is computed as:

'bits_at_hand' ←'seg_len'*'bpp' + 'bfps';

'abh' is computed as:

'abh' ←ROUND('bits_at_hand');

The chaotic sequence is generated with the stego-key "AB106F" and parameter $\lambda = 0.9999$.

Since the sequence is large to fit in the cell, therefore the sequences are listed here:

seq1: 29, 23, 25, 9, 18, 31, 21, 4, 14, 16, 6, 3, 20, 7, 8, 26, 0, 27

seq2: 23, 15, 25, 3, 27, 7, 18, 21, 16, 2, 19, 26, 24, 28, 11, 29, 14, 13

seq3: 30, 0, 19, 2, 8, 18, 16, 27, 22, 29, 4, 14, 25, 23, 5, 24, 15, 7

seq4: 1, 0

The actual pixel position in the image is computed as follows:

'pixel_position' ←'32'*'seg_count' + 'seq'['i'];

compueted as above the pixel positions are:

px_pos1: 29, 23, 25, 9, 18, 31, 21, 4, 14, 16, 6, 3, 20, 7, 8, 26, 0, 27

px_pos2: 55, 47, 57, 35, 59, 39, 50, 53, 48, 34, 51, 58, 56, 60, 43, 61, 46, 45

px_pos3: 94, 64, 83, 66, 72, 82, 80, 91, 86, 93, 68, 78, 89, 87, 69, 88, 79, 71

px_pos4: 97, 96

Table 1: Simulation Result of the Algorithm

'seg_count'	**0**	**1**	**2**	**3**
No. of pixels in segment (seg_len).	*32*	*32*	*32*	*4*
'bfps'	*0.0*	*-0.08*	*-0.16*	*-0.24*
'bits_at_hand'	*17.92*	*17.81*	*17.76*	*2*
'abh'	*18*	*18*	*18*	*2*
Chaotic Sequence	*seq1*	*seq2*	*seq3*	*seq4*
pixel positions for hiding next 'abh' no. of bits	*px_pos1*	*px_pos2*	*px_pos3*	*px_pos4*

Table 2: Hiding

'R'	***0***	***1***	***0***	***1***	***0***	***0***	***1***	***0***
px_pos	*29*	*23*	*25*	*9*	*18*	*31*	*21*	*4*
'a'	*0*	*1*	*1*	*0*	*0*	*0*	*0*	*1*
px_pos	*14*	*16*	*6*	*3*	*20*	*7*	*8*	*26*
'j'	*0*	*1*	*1*	*0*	*1*	*0*	*1*	*0*
px_pos	*0*	*27*	*55*	*47*	*57*	*35*	*59*	*39*
'p'	*0*	*1*	*1*	*1*	*0*	*0*	*0*	*0*
px_pos	*50*	*53*	*48*	*34*	*51*	*58*	*56*	*60*
'o'	*0*	*1*	*1*	*0*	*1*	*1*	*1*	*1*
px_pos	*43*	*61*	*46*	*45*	*94*	*64*	*83*	*66*
'o'	*0*	*1*	*1*	*0*	*1*	*1*	*1*	*1*
px_pos	*72*	*82*	*80*	*91*	*86*	*93*	*68*	*78*
't'	*0*	*1*	*1*	*1*	*0*	*1*	*0*	*0*
px_pos	*89*	*87*	*69*	*88*	*79*	*71*	*97*	*96*

Above 56 pixel positions are the positions where the subsequent bits of the 56 bits of the message are hidden. Selected pixel's Luma component is adjusted to encode the message bit.

4. EXPERIMENTAL RESULTS

The method is separately tested with two standard images 'lenna', 'babun' of resolution of '512*512' against various quality measures. The secret message is embedded in each image is with a compression tolerance coefficient of '15'. A 512 bytes secret text message is used to test a given combination of image and compression tolerance coefficient. For compression tolerance coefficients '01' the method shows very good results against all the three requirements of Steganogaphy i.e. Payload capacity, visual Imperceptibility and Robustness, but it could not tolerate compression. We analyze the results with the most common quality measures histogram and entropy only.

Fig 2: Original Image 'lenna'

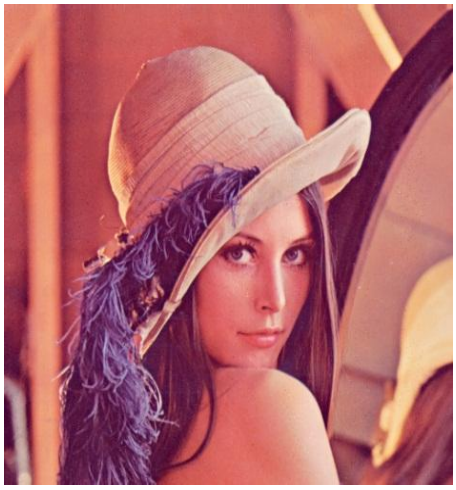

Fig. 3:Stego-Image 'lenna'

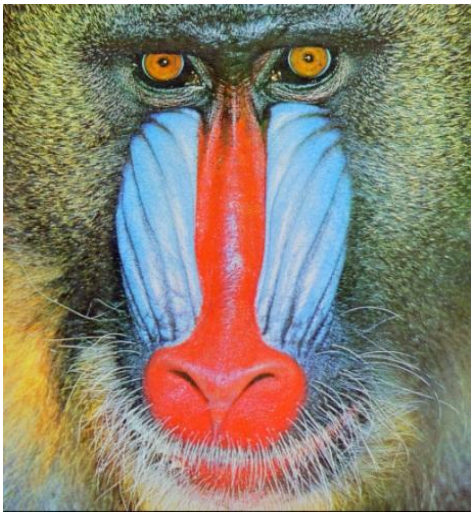

Fig 4: Original Image 'babun'

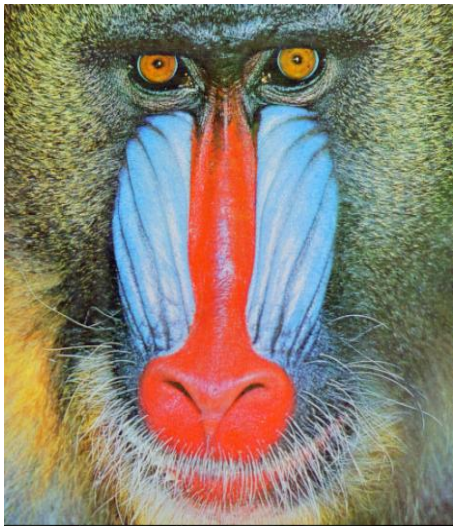

Fig 5:Stego-Image 'babun'

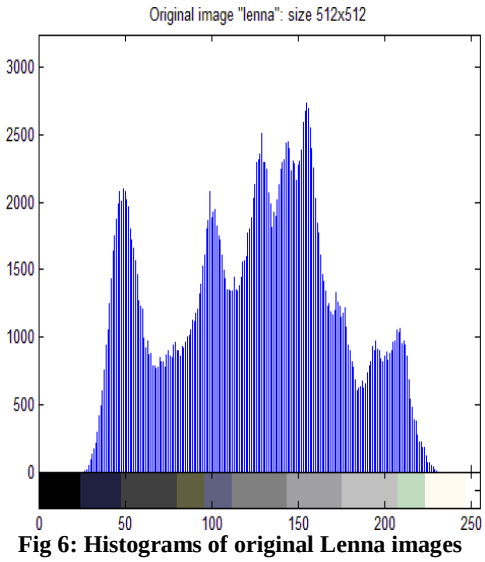

Fig 6: Histograms of original Lenna images

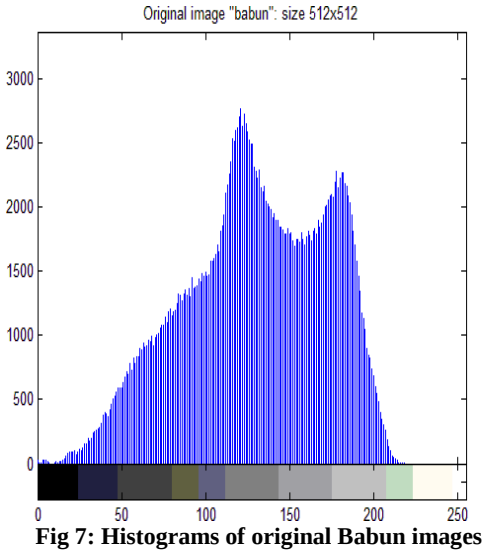

Fig 7: Histograms of original Babun images

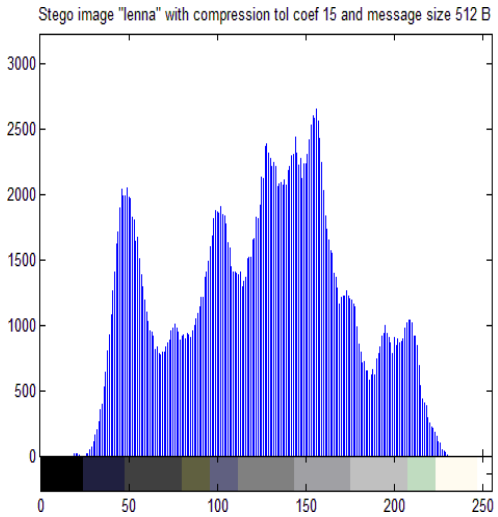

Fig 8: Histograms of stegol Lenna images

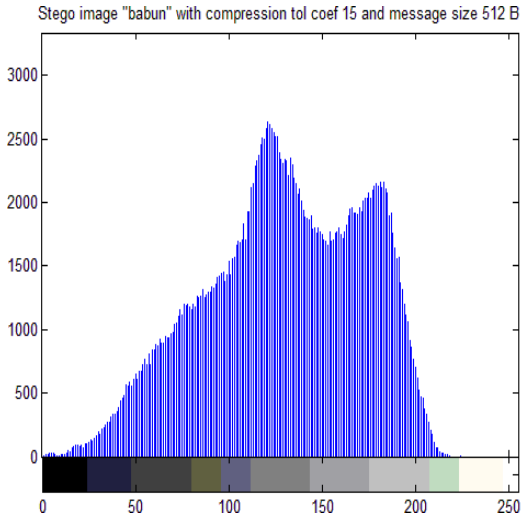

Fig 9: Histograms of stego Babun images

Entropies of luma component of original images 'lenna' and 'babun' are 7.3583 and 7.4451 respectively, and that of corresponding stego images are 7.3716 and 7.4596 respectively. This shows that the change in entropies is not much noticeable. If any attack would unfortunately guess or find the presence of some secret communication in the image, yet it is almost impossible to retrieve the secret message.

5. CONCLUSION

In this paper, we present a robust method that uses two chaotic maps for selecting pixels to be used for embedding the bits of secret message. Further, the initial conditions are changed during each iteration, with fixed number of pixels and the initial conditions are made chaotic to hide the secret message from the unauthorized persons or attackers. Hence the capability of the method to keep the secret message secret is very high and at the same time keeping the method robust. The salient feature of this method is compression resistant. The results show that this method can successfully embed the secret message by tolerating the changes upto'15' in Luma component, without affecting the image quality much.

The payload capacity of the method is also high as it embeds the secret message in spatial domain not in DCT coefficient unlike in transform domain in which one has to compromise with the payload capacity.

For further enhancing the robustness, this method can be preceded by an encryption technique that is fist encrypt the secret message and then embed into a cover-image by the proposed method.

The method can also be implemented upon segments of 32-pixels in parallel by using parallel programming; this would increase the speed of the process.

6. FUTURE WORK

The method can be improved by making the pixel selection depending upon the properties of the cover image, e.g. we can select only those segments which are noisier and then select the pixels chaotically using chaotic maps. To hide the secrete communication we can also select in the image the areas that marks some objects identified according to some pre-specified criteria. Although making the pixel selection in this way may decrease the payload capacity, but will definitely increase the imperceptibility of the method and also the robustness.

7. REFERENCES

[1] Herodotus , John M. Marincola and Aubrey De Selincourt, "Herodotus:T he Histories". Penguin Classics, London, 1996

[2] Shawn D. Dickman "An Overview of Steganography" James Madison University InfosecTechreport, JMU-INFOSEC-TR-2007-002, July 2007.

[3] Ingemar J. Cox, Matthew L. Miller, JeffreyA. Bloom, JessicaFridrich, TonKalker, "Digital Watermarking and steganography" Morgan Kaufmann, 2007.

[4] Pan ,H.K., Y.Y., Chen, and Y.C., Tseng, "A Secure Data Hiding Scheme for Two-Color Images", Proc. Fifth IEEE Symp.Computers and Comm., IEEE Press, Piscataway, N.J., 2000.

[5] Westfeld, A., "F5-A Steganographic Algorithm: High Capacity Despite Better Steganalysis", 4th International Workshop on Information Hiding, 2001.

[6] Bhavana.S, K.L.Sudha, "Text Steganography Using LSB Insertion Method Along With Chaos Theory", International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.2, No.2, April 2012.

[7] Tayel, M.; Shawky, H.; Hafez, A.E.S., "A new chaos steganography algorithm for hiding multimedia data," Advanced Communication Technology (ICACT), 2012 14th International Conference on , vol., no., pp.208,212, 19-22 Feb.

[8] Bo Wang; Jiuchao Feng, "A chaos-based steganography algorithm for H.264 standard video sequences," Communications, Circuits and Systems, 2008. ICCCAS 2008. International Conference on , vol., no., pp.750,753, 25-27 May 2008

[9] Nidhi Sethi and Deepika Sharma, (2012) "A novel method of image encryption using logistic mapping", in Proc. of International Journal of Computer Science Engineering (IJCSE), Vol. 1, No. 2, pp. 115-119.