Enhanced Chaotic Block Cipher Algorithm for Image Cryptosystems

Vishnu G. Kamat
M Tech in Information Security and Management
Department of Computer Science, DIT University
Dehradun, India
kamatvishnu14@gmail.com

Madhu Sharma
Assistant Professor
Department of Computer Science, DIT University
Dehradun, India
madhuashishsharma@gmail.com

Abstract— ***In this paper, we provide possible vulnerabilities of the algorithm set forth by Mohamed Amin et al. [Commun. Nonlinear Sci. Numer. Simulat. 15, 3484-97, (2010)]. They proposed a block cipher algorithm using chaotic Tent map. For the Feistel structure used there are a certain vulnerabilities that can be exploited to find out the rotations the plaintext goes through. For a small number of rounds this can be used to retrieve the plaintext or the key. Hence we have proposed an enhanced algorithm which shows favorable results.***

Keywords-Encryption; Chaotic Maps; Block Cipher; Security Analysis

I. Introduction

There has been a lot of research done recently in the field of chaotic cryptography. In this field, for encryption we use chaotic maps, which generate good pseudo-random numbers. Cryptographic properties of these maps such as, sensitive dependence on initial parameters, ergodicity and random like behavior, make them ideal for use in designing secure cryptographic algorithms. Many scholars have proposed various chaos-based encryption schemes in recent years [1-7]. A scheme proposed by Mohamed Amin et al. [8] uses Tent map for chaotic key generation and follows a Feistel-like structure. In this paper, we discuss certain possible vulnerabilities that can be exploited in the algorithm proposed in [8]. We then propose an algorithm which enhances the scheme and provides better results.

II. Overview of Current Scheme

The Encryption scheme proposed in [8] is as shown below in Fig 1.

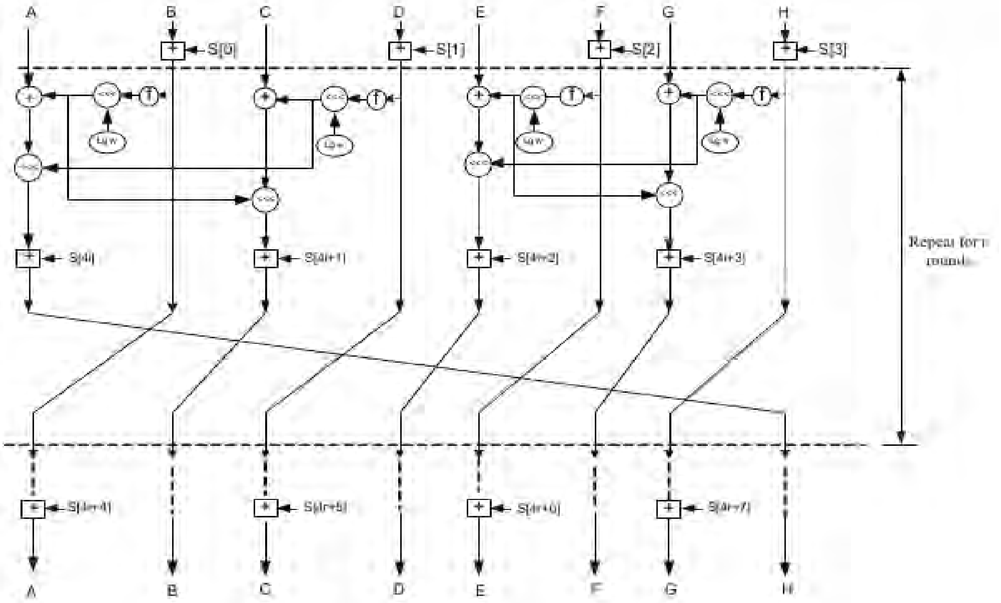
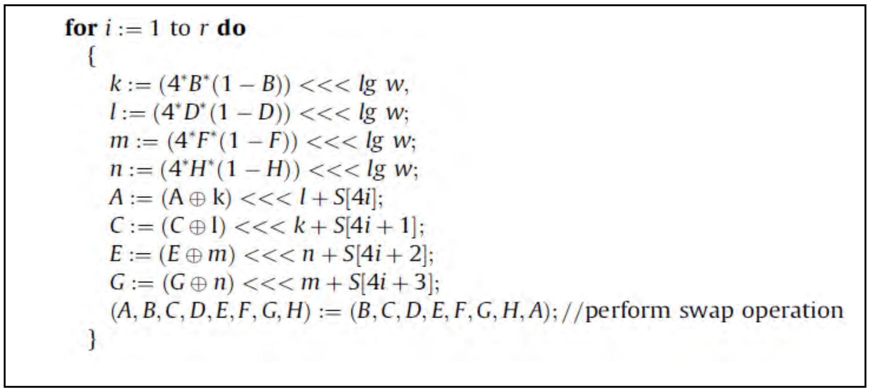


Figure 1. Current Encryption Scheme

In this scheme the plaintext (image) is taken as blocks of bits rather than block of pixels. The block size is 8w, where ‘w’ is the word size which can be 16, 32 or 64 bits. Each block of data is divided and stored into 8 w-bit registers and operations are performed on them. The key length depends on the number of rounds ‘r’ ie. Key length is 4r+8.

In the first step 4 bytes of the key are added to alternate registers. 2’s compliment addition is performed. Then for ‘r’ rounds following steps are performed as indicted in Fig. 2 as proposed in [8]. It uses a function ‘f’, the output of which is rotated depending on ‘w’ and is used as the number of rotations to be performed on another block of data. The number of rounds can vary from 0-255. After the swapping operation of the last round, the last 4 key bytes are added.



```
for $i := 1$ to $r$ do
  {
    $k := (4^*B^*(1 - B)) <<< lg\ w,$
    $l := (4^*D^*(1 - D)) <<< lg\ w;$
    $m := (4^*F^*(1 - F)) <<< lg\ w;$
    $n := (4^*H^*(1 - H)) <<< lg\ w;$
    $A := (A \oplus k) <<< l + S[4i];$
    $C := (C \oplus l) <<< k + S[4i + 1];$
    $E := (E \oplus m) <<< n + S[4i + 2];$
    $G := (G \oplus n) <<< m + S[4i + 3];$
    $(A, B, C, D, E, F, G, H) := (B, C, D, E, F, G, H, A);$ //perform swap operation
  }
```

Figure 2. Current Encryption Algorithm

III. Vulnerability Details

To attack a cryptosystem an attacker tries to decrypt an encoded message. But decrypting a single message may not be sufficient. Hence the attacker tries to decipher the key, which will allow the person to decrypt all the subsequent messages. To find out the key attacker can try out all possible combinations of the key, known as brute force method. Since brute force method is lengthy more recent cryptanalytic techniques are used. Cryptanalysis is the art of deciphering encrypted communications without knowing the proper keys. Given some encrypted data, the goal of the cryptanalyst is to gain as much information as possible about the original, unencrypted data. Based on the amount of information available to the cryptanalyst, the cryptanalytic techniques used differ. Some important ones are described below.

- Ciphertext-only: the cryptanalyst has access only to a collection of ciphertexts. In this situation the attacker does not know anything about the plaintext, and must work from ciphertext only.
- Known-plaintext: Here the attacker has a set of encrypted texts to which he knows the corresponding plaintexts. He then uses this information to decrypt future ciphertexts or to retrieve the key.
- Chosen-plaintext: In this case the attacker obtains the ciphertext for a plaintext of his choice and then tries to use this information to retrieve the key.
- Chosen-ciphertext: In this case the attacker obtains the plaintext for a ciphertext of his choice and then tries to use this information to retrieve the key.

We use the chosen plaintext method for finding out the vulnerabilities. For the purpose of cryptanalysis we first encrypted the plaintext of all zeros ie. All the plaintext bits are zero. The size of the registers used is 32 bits ie. w=32 and a single round of encryption is performed. To find out the number of rotations performed we now introduce a single 1 in our plaintext of all zeros. The position of the 1 after one round will enable us to find out the number of rotations performed.

We have used different key bytes to check the amount of rotation performed on the registers after initial key addition. After testing we found that the rotation performed on the register data is the same for various key bytes, as shown in Table I. The rotations shown are the values of effective rotations performed ie. the values of k, l, m and n in Fig. 2. The table also shows that the number of rotations change at each step after the modifications to the function.

For subsequent rounds the rotations change, but since the algorithm rounds can vary from 1-255, if a single round is performed the rotations tend to not provide any extra strength. The rotation performed based on the value of ‘w’ also does not provide much strength as the value of w can be 16, 32, or 64 and the rotations will be log of either of the three.

Now, since we know the plaintext bits and the number of rotations performed on it, we can reverse the rotation process. This value will be addition of two key bytes. Trying many such known plaintexts we can derive the key. If the key is found, then the whole cryptosystem is not secure.

For a single round of encryption if any key byte is zero, then we can directly get the value of a key byte. Eg. Based on Fig. 1 and the above method of finding the number of rotations performed, if the first key byte is zero, then we get the value of the 8th key byte. If the value of 5th key byte is zero we get the value of the 2nd key byte. Similarly we can get hold of the entire key.

IV. PROPOSED SCHEME

Now we propose enhancements to the scheme described above. This will keep the merits of the previous algorithm while fixing the vulnerabilities.

We designed our algorithm for color(RGB) images. We have not used any user defined key. The key is generated by the 3D chaotic Rossler map as shown in (1). Rossler map is a system of 3 non-linear ordinary differential equations [9]. This system was put forth by Otto Rossler in 1976.

$$\begin{aligned} X_{n+1} &= -Y_n - Z_n \\ Y_{n+1} &= X_n + \alpha Y_n \\ Z_{n+1} &= \beta + Z_n (X_n - \gamma) \end{aligned} \tag{1}$$

α, β and γ are real parameters. Rossler system is chaotic for the values of $\alpha=0.432$, $\beta=2$ and $\gamma=4$. The number of key bytes 't' depends on the rounds i.e. t=4r+8; where r is the number of rounds. We have introduced a way to use the 3 dimensions of the equations separately. Each dimension of the map is used separately as a key during the encryption process of the red, gren and blue channel respectively. The key generation concept is as shown below. The steps repeat 't' times to generate necessary key bytes.

a. iterate equation (1) 'r' times where 'r' is the number of rounds.

b. use the decimal part of the X, Y, Z values to generate the key byte.

X_n = abs(X_n - integer part); // decimal part of x

Y_n = abs(Y_n - integer part); // decimal part of y

Z_n = abs(Z_n - integer part); // decimal part of z

c. key byte for each dimension (RGB) is taken as X, Y, Z values respectively by mapping it to a value between 0-255.

d. for the next set of key bytes the number of iterations is changed to a value obtained by performing exclusive-or on the current set of key bytes.

Iterations for next key byte = XOR (X_n , Y_n, Z_n);

The general structure followed is similar to the one shown in Fig. 1. The processing is done on 256 bits (32 bytes) of data at a time using eight 32-bit registers. To fix the vulnerability based on number of rotations we propose the following changes to the function 'f'. The changed function will perform the following on a data in registers depicted as B in equation (2).

$$f = B*((1-B) \div 4); \tag{2}$$

Table I shows the effect of change of function 'f' on the rotations after changing it to (2). This remove the vulnerability yet keeping the structural benefits of the algorithm.

TABLE I. SINGLE ROUND ROTATIONS PERFORMED BEFORE AND AFTER FUNCTION CHANGE

First 4 Key Bytes Added	Rotation Before Function Change	Rotation After Function Change
45, 67, 115, 181	31, 31, 31, 31	16, 17, 9, 12
22, 216, 172, 2	31, 31, 31, 31	16, 18, 4, 18
4, 137, 206, 160	31, 31, 31, 31	14, 28, 16, 22
45, 79, 154, 229	31, 31, 31, 31	3, 17, 29, 4
68, 85, 89, 14	31, 31, 31, 31	26, 11, 10, 7

The rotation based on ‘w’, the word size is removed as it did not add extra advantage. The diffusion function that it performed was limited to a small block of pixels in a register. For the actual diffusion to propagate it would be advantageous to add a diffusion step for each channel after the encryption of that channel. The diffusion process described in [10] is a good way to ensure that the key value spreads over the entire image. We have added the horizontal diffusion for each channel after its encryption, propagating the changes for a single channel at a time. The vertical diffusion can be added before and/or after the entire encryption process. For our experimentation purpose we have used vertical diffusion both before and after the entire encryption process.

In the next section we show the experimentation results of the new improved scheme incorporating the changes described in this section.

V. Experimentation Results

For demonstrating the security of the enhanced algorithm we show the analysis performed on six 256 x 256 color(RGB) images as shown in Fig. 3(a-f). The results of the statistical and differential analysis tests performed are shown in this section.

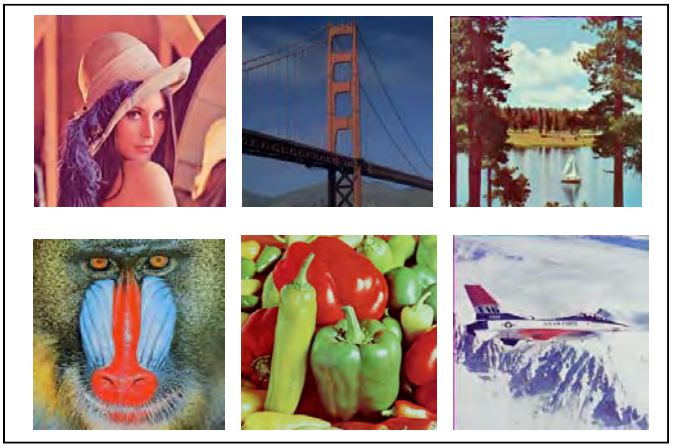


Figure 3. Plain images (clockwise from top left): (a) Lena (b) Bridge (c)Lake (d) Plane (e) Peppers (f) Mandrill.

A. Statistical Analysis

Statistical analysis of the cipher image is very important to ascertain that the cipher image does not bear any resemblance to the plain image and that the cipher image pixels do not have much correlation among them. In this section we provide the histogram and correlation analysis.

1) Histogram Analysis:

For information to be secure from leakage, the encrypted image should bear very little resemblance to the plain image. Histograms plot number of pixels at each intensity level. This shows how pixels are distributed.

Fig. 4 depicts the histogram of the plain image ‘lena’ for the red, green and blue channels on the left side (from top to down) and the histograms of the ‘lena’ image after encryption for the 3 channels respectively on the right side. They depict that the encryption does not leave any concentration of a single pixel value.

2) Correlation Analysis:

In a plain image the adjacent pixels are highly correlated in the vertical or horizontal direction. The cipher image should not bear much correlation among adjacent pixels. To calculate the correlation coefficient, 1000 random pairs of pixels are selected from the image and following formula is applied.

$$corr_{xy} = \frac{C(x,y)}{\sqrt{D(x)D(y)}} \tag{3}$$

where,

$$C(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))(y_i - E(y)) \tag{4}$$

$$D(x) = \frac{1}{N}\sum_{i=1}^{N}\left(x_i - E(x)\right)^2 \tag{5}$$

$$E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{6}$$

Here x_i and y_i form the i^{th} pair of adjacent pixels and N is the total number of pairs.

Table II shows the correlation between horizontal, vertical and diagonal adjacent pixels of the 6 plain images shown in fig. 1. It can be noted that there is significant resemblance among the adjacent pixels.

Table III shows the correlation coefficients (horizontal, vertical and diagonal), for the Red, Green and Blue channel, of the cipher images formed by encrypting the plain images with the proposed encryption algorithm. This displays that the cipher images bear very little resemblance to the original images and that the adjacent pixels do not resemble as well.

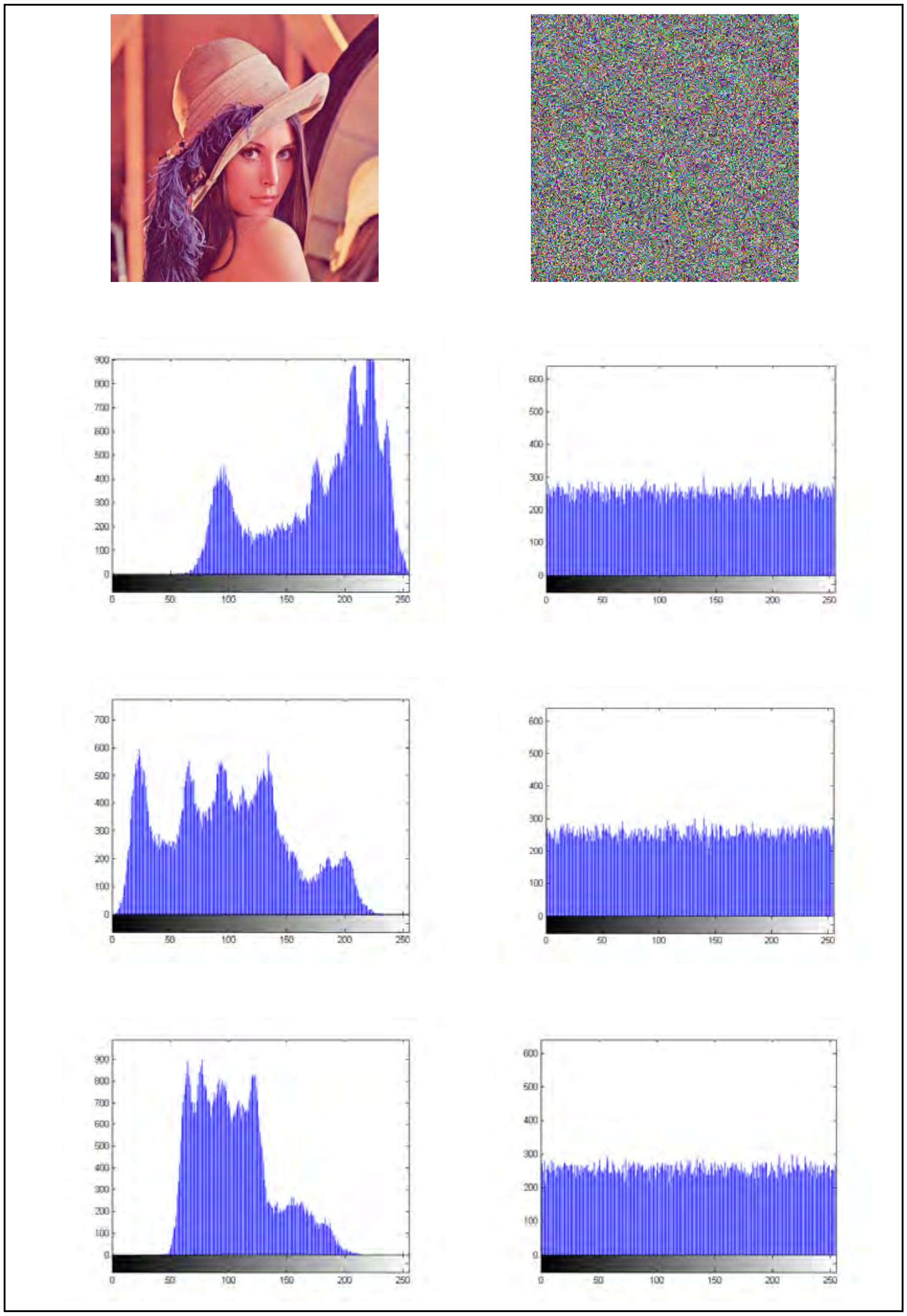


Figure 4. Left Side: Histogram of 'lena' plain image for red, grren and blue channels (top to down). Right Side: Histogram of encrypted 'lena' image for red, green and blue channels (top to down).

B. *Differential Analysis*

To perform differential analysis, an adversary makes a small change (like changing a pixel value of an image) in the plaintext and tries to find out meaningful relationship between plaintext and ciphertext, by comparing the 2 different ciphertext of similar plaintext.

Two common measures used for differential analysis are: NPCR (net pixel change rate) and UACI (unified average changing intensity). NPCR shows the percentage rates at which pixels change in the cipher image when pixels of plain image are changed. UACI measures average intensity of difference between plain image and cipher image.

Let us consider 2 cipher images X_1 and X_2, obtained by plain images I_1 and I_2, where I_1 and I_2 have a single pixel difference. The pixel values at the grid position (i,j) for the two cipher images are denoted as $X_1(i,j)$ and $X_2(i,j)$. A bipolar array B is defined as follows

$$B(i,j) = \begin{cases} 0, \ if \ X_1(i,j) = X_2(i,j) \\ 1, \ if \ X_1(i,j) \neq X_2(i,j) \end{cases} \quad (7)$$

NPCR and UACI values are calculated as given in equation (8) and (9), where W and H denote width and height of the cipher images, T denotes the largest supported pixel value in the cipher images (255 in our case) and abs() computes the absolute value. The results of our analysis are provided in Table IV.

$$\mathrm{NPCR} = \frac{\sum_{i,j} B(i,j)}{\mathrm{W \; x \; H}} \text{ x } 100\% \quad (8)$$

$$\mathrm{UACI} = \frac{1}{\mathrm{W \; x \; H}} \left[\sum_{i,j} \frac{abs(\mathrm{x}_1(i,j) - \mathrm{x}_2(i,j))}{T} \right] \text{ x } 100\% \quad (9)$$

TABLE II. CORRELATION VALUES OF **PLAIN-IMAGES**

Channels	**Plain Images**	**Horizontal**	**Vertical**	**Diagonal**
RED	Lena	0.9558	0.9781	0.9336
	Bridge	0.8680	0.9070	0.8287
	Lake	0.9234	0.9201	0.8886
	Mandrill	0.8474	0.8032	0.7944
	Peppers	0.9371	0.9392	0.9077
	Plane	0.9205	0.9092	0.8546
GREEN	Lena	0.9401	0.9695	0.9180
	Bridge	0.9055	0.9131	0.8700
	Lake	0.9354	0.9272	0.8943
	Mandrill	0.7285	0.6674	0.6487
	Peppers	0.9657	0.9673	0.9451
	Plane	0.8938	0.9174	0.8419
BLUE	Lena	0.9189	0.9495	0.8948
	Bridge	0.9354	0.9411	0.9138
	Lake	0.9377	0.9401	0.9099
	Mandrill	0.8030	0.7914	0.7625
	Peppers	0.9259	0.9330	0.8928
	Plane	0.9179	0.8912	0.8563

TABLE III. CORRELATION VALUES OF **CIPHER-IMAGES**

Channels	Plain Images	Horizontal	Vertical	Diagonal
RED	Lena	-0.0014	-0.0012	0.0004
	Bridge	-0.0040	-0.0066	-0.0010
	Lake	-0.0052	-0.0011	0.0018
	Mandrill	0.0034	0.0001	0.0033
	Peppers	-0.0014	-0.0034	-0.0016
	Plane	-0.0024	-0.0043	0.0088
GREEN	Lena	0.0004	0.0067	-0.0026
	Bridge	-0.0053	-0.0017	0.0008
	Lake	0.0044	-0.0025	0.0068
	Mandrill	-0.0031	-0.0041	0.0029
	Peppers	0.0008	0.0027	0.0029
	Plane	0.0026	-0.0003	0.0014
BLUE	Lena	-0.0049	0.0014	-0.0005
	Bridge	0.0023	0.0001	0.0037
	Lake	-0.0010	-0.0044	0.0002
	Mandrill	0.0023	0.0001	-0.0014
	Peppers	-0.0016	-0.0006	0.0013
	Plane	0.0040	-0.0007	0.0041

TABLE IV. NPCR AND UACI VALUES OBTAINED FOR ENCRYPTION OF 6 PLAIN IMAGES AND SAME IMAGES WITH 1 PIXEL CHANGED

Plain Images	NPCR	UACI
Lena	99.6333	33.4706
Bridge	99.5722	33.4403
Lake	99.5900	33.5313
Mandrill	99.6089	33.4595
Peppers	99.6185	33.4657
Plane	99.6206	33.4539

VI. CONCLUSION

The algorithm proposed in [8] had a few weakness and vulnerabilities. Key bytes could be determined by cryptanalysis as the function performed fixed rotations. We proposed an enhanced algorithm which fixed the issues and provided better performance. The enhanced algorithm is implemented for 3D images using Rossler system. Diffusion for each step increased the strength by spreading the key across the channel. Experimentation results depict that the enhanced algorithm is stronger and secure.

REFERENCES

[1] Chen G, Mao Y, and Chui CK. “A symmetric image encryption based on 3D chaotic cat maps”, Chaos Solitons and Fractals, vol. 21, pp. 749-61, 2004.

[2] Mao Y, Chen G, and Lian S. “A novel fast image encryption scheme based on 3D chaotic Baker maps”, Int J Bifurc Chaos, vol. 14(10), pp. 3613-24, 2004.

[3] Guan Z-H, Huang F, and Guan W. “Chaos based image encryption algorithm”, Phys Lett A, vol. 346, pp. 153-7, 2005.

[4] Zhang L, Liao X, and Wang X. “An image encryption approach based on chaotic maps”, Chaos Solitons and Fractals, vol. 24, pp. 759-65, 2005.

[5] Gao H, Zhang Y, Liag S, and Li D. “A new chaotic algorithm for image encryption”, Choas Solitons and Fractals, vol. 29, pp. 393-399, 2006.

[6] Pareek NK, Patidar V, and Sud KK. “Image encryption using chaotic logistic map”, Image Vision Comput, vol. 24, pp. 926-34, 2006.

[7] Wong K-W, Kwok BS-H, and Law W-S. “A fast image encryption scheme based on chaotic standard map”, Phys Lett A, vol. 372, pp. 2645-52, 2008.

[8] Amin M, Faragallah OS, and Abd El-Latif AA. “A chaotic block cipher algorithm for image cryptosystems”, Commun Nonlinear Sci Numer Simulat, vol. 15, pp. 3484-3497, 2010.

[9] Rössler OE, “An equation for continuous chaos”, Physics Letters A, vol. 57 (5), pp. 397-398, 1976.

[10] Patidar V, Pareek NK and Sud KK. “A new substitution-diffusion based image cipher using chaotic standard and logistic maps”, Commun Nonlinear Sci Numer Simulat, vol. 14, pp. 3056-3075, 2009