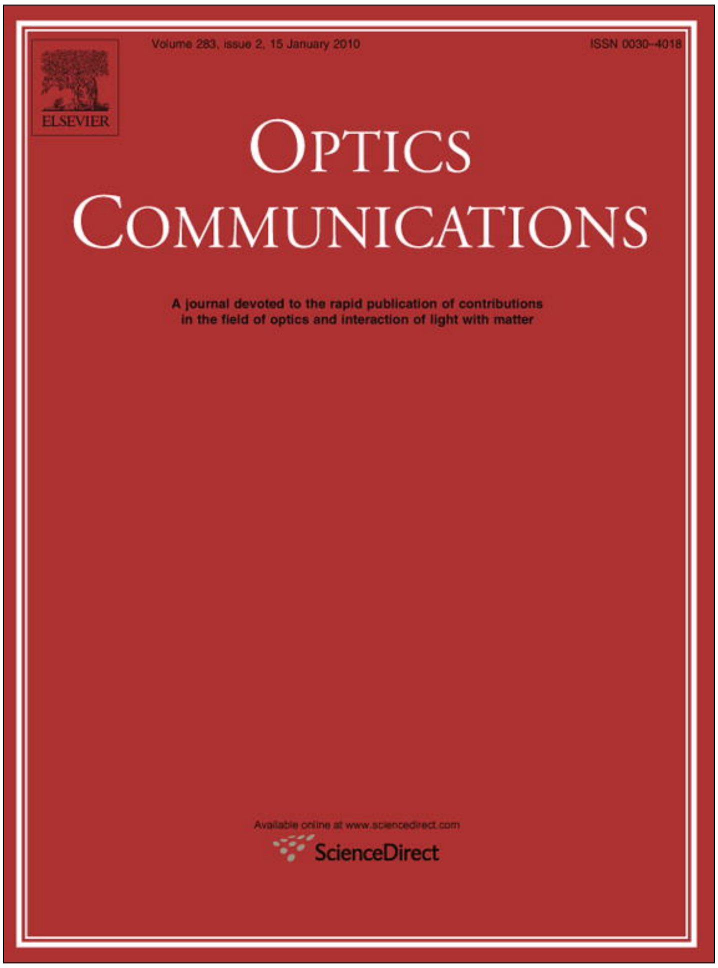

Volume 283, issue 2, 15 January 2010

ISSN 0030-4018

ELSEVIER

Optics Communications

A journal devoted to the rapid publication of contributions
in the field of optics and interaction of light with matter

Available online at www.sciencedirect.com

ScienceDirect

Optics Communications 283 (2010) 232–236

Contents lists available at ScienceDirect

Optics Communications

journal homepage: www.elsevier.com/locate/optcom

Cryptanalysis of a multi-chaotic systems based image cryptosystem

Ercan Solak [a,*], Rhouma Rhouma [b], Safya Belghith [b]

[a] *Department of Computer Science and Engineering, Isik University, 34980 Istanbul, Turkey*
[b] *Syscom Laboratory, Ecole Nationale d'Ingénieurs de Tunis, Tunisia*

ARTICLE INFO

Article history:
Received 11 August 2009
Accepted 29 September 2009

PACS:
05.45.Vx
05.45.−a
07.05.Pj

Keywords:
Chaos based cryptography
Shuffling
Chosen-plaintext attack
Known-plaintext attack

ABSTRACT

This paper is a cryptanalysis of a recently proposed multi-chaotic systems based image cryptosystem. The cryptosystem is composed of two shuffling stages parameterized by chaotically generated sequences. We propose and implement two different attacks which completely break this encryption scheme.

1. Introduction

Chaotic systems exhibit many suitable properties that make them applicable to the design of encryption schemes. Among the parallels between the properties of chaotic systems and those of cryptosystems are [1]:

1. Sensitivity to initial conditions → A small deviation in the plaintext results in a large change in the ciphertext.
2. Deterministic dynamic and pseudo-random aspect of a chaotic signal → A deterministic process (in a cryptosystem) can cause a pseudo-random behavior.
3. Ergodicity of a chaotic signal → The performance of an encryption algorithm has the same distribution for any plaintext.

As a result, many proposals dealing with both cryptography and chaos have been published in the last twenty years [2–9]. Some of them have been cryptanalysed [5–9] and have been found to be not secure [10–13].

In this paper, we cryptanalyze the image cryptosystem recently proposed in [2]. The cryptosystem uses the Henon, the Lorenz, the Chua and the Rössler chaotic systems to generate shuffling sequences. However, this was not enough to make the cryptosystem secure. Indeed, the use of these systems was only for generating pseudo-random sequences that are used to rearrange the pixel bits of the plain image. Hence, the cryptosystem under study can be viewed as a pure shuffling algorithm that can be broken using a method similar to the one developed in [14]. In the present paper, in order to break the image cryptosystem, we propose two different attacks specific to this scheme.

The rest of this paper is organized as follows: Section 2 gives a brief description of the cryptosystem proposed in [2]. Section 3 gives an equivalent description of the cryptosystem that makes it simpler to analyze. Using the equivalent description, a chosen ciphertext attack and a known-plaintext attack are given. Simulations results are given in Section 4. The paper concludes with final remarks in Section 5.

2. Description of the cryptosystem

The cryptosystem proposed in [2] shuffles plaintext image bits using chaotic systems. The shuffling parameters are generated by the iterations of four 3D chaotic systems. The key of the cryptosystem is the set of 12 initial conditions for the chaotic maps. The parameters of the chaotic systems are fixed and public.

The shuffling is performed in two stages. In the first stage, designated bits of all the pixels are shuffled. In the second stage, the bits of each pixel are shuffled among themselves.

We now give the detailed descriptions of each stage.

[*] Corresponding author. Tel.: +90 216 528 7149; fax: +90 216 710 2872.
E-mail address: ercan@isikun.edu.tr (E. Solak).

0030-4018/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.optcom.2009.09.070

2.1. Plaintext preparation

The original plaintext is a $m \times n$ RGB image with each pixel color represented as a byte. For the purpose of encryption, the plaintext is first vectorized using the usual row scan. The resulting vector is a $N \times 1$ vector of bytes, where $N = mn$. In order to manipulate the bits of pixels, the vector is further split into its bits, resulting in a $N \times 8$ plaintext matrix, where each entry takes values 0 or 1. In the cryptosystem proposed in [2], each color component is processed independently. $Arb, Agb, Abb \in \{0,1\}^{N\times 8}$ denote respectively the Red, Green and Blue components of the vectorized and binarized image.

2.2. Key preparation

The four chaotic systems are iterated N times to generate 12 sequences of length N each. Each of these sequences are sorted and the indices of the sorted numbers in the original sequences give the sorting sequences $Fx_1, Fy_1, Fz_1, Fx_2, Fy_2, Fz_2, Fx_3, Fy_3, Fz_3, Fx_4, Fy_4$ and Fz_4.

Here, x_1, y_1, z_1 are the sequences of real numbers generated by the Henon chaotic system. Likewise, x_2, y_2, z_2 are generated by the Lorenz system, x_3, y_3, z_3 are generated by the Chua system and finally x_4, y_4, z_4 are generated by the Rössler system.

For example, if $x_1 = \{2.7, -0.4, 1.7, 0.1\}$, then the sorting sequence is given as $Fx_1 = \{2, 4, 3, 1\}$ because that is the indices of the ordered elements.

Note that the sequences $Fx_i, Fy_i, Fz_i, 1 \leqslant i \leqslant 4$, depend nonlinearly on the secret initial conditions. Also note that each sorting sequence defines a permutation over the set of integers $\{1, 2, \ldots, N\}$.

2.3. Encryption

2.3.1. Vertical shuffling

Each column of the matrix Arb is shuffled using one of the sequences Fx_1, Fx_2, Fx_3 and Fx_4 to generate the intermediate matrix $Aerb$. The choice of the shuffling sequence is determined as follows;

$$Aerb(i,j) = \begin{cases} Arb(Fx_1(i),j), & j = 1,2, \\ Arb(Fx_2(i),j), & j = 3,4, \\ Arb(Fx_3(i),j), & j = 5,6, \\ Arb(Fx_4(i),j), & j = 7,8. \end{cases} \tag{1}$$

The same vertical shuffling is applied to the green and blue components Agb and Abb with the shuffling sequences Fy_1, Fy_2, Fy_3, Fy_4 and Fz_1, Fz_2, Fz_3, Fz_4, respectively. After the vertical shuffling, we end up with $Aerb, Aegb$ and $Aebb$.

2.3.2. Horizontal shuffling

In this step, the rows of $Aerb, Aegb$ and $Aebb$ are shuffled horizontally. This corresponds to the shuffling of pixel bits among themselves.

For the shuffling of the ith row of red component, the sorting sequence for 4 numbers $Fx_1(i), Fx_2(i), Fx_3(i), Fx_4(i)$ are used. If the sorting sequence is $\{j_1, j_2, j_3, j_4\}$, the ith row is shuffled into the row

$$\begin{aligned} Brb(i) = [&Aerb(i,2j_1), Aerb(i,2j_1-1), Aerb(i,2j_2), \\ &Aerb(i,2j_2-1)Aerb(i,2j_3), Aerb(i,2j_3-1), \\ &Aerb(i,2j_4), Aerb(i,2j_4-1)], \end{aligned} \tag{2}$$

where $Brb(i)$ denotes the ith row of the $N \times 8$ binary matrix of the ciphertext red component.

The red component Br of the ciphered image B is obtained by first concatanating the bits in each row of Brb to obtain the pixels in vector format and then reshaping the vector into a $m \times n$ ciphered image.

A similar shuffling of the pixel bits is applied to the green and blue components $Aegb$ and $Aebb$ with the sorting sequences derived from Fy and Fz, respectively.

The final ciphered image B is obtained by joining all three color components Br, Bg and Bb.

3. Cryptanalysis

In this section, we give chosen-plaintext and known-plaintext attacks against the cryptosystem. Both of the attacks yield the sorting sequences Fx, Fy and Fz with very little amount of computation. We first give an equivalent representation of the encryption algorithm using permutation functions.

3.1. Equivalent representation

As claimed in [2], the security of the cryptosystem relies on the secrecy of the initial conditions driving the four chaotic systems. A naive attack on the cryptosystem might try to reveal those initial conditions. However, if an attacker knows the sorting sequences Fx, Fy and Fz, he can decrypt the ciphered image. Thus, the sorting sequences are the equivalent keys of the cryptosystem. Instead of attacking the initial conditions, the attacker devises methods to reveal the sorting sequences.

At first glance, it might seem that, by using the equivalent representation, we have unnecessarily increased the number of secret parameters. Indeed, in the original proposal there are 12 secret keys. Since each sorting sequence defines a permutation over N elements, the new key space has $(N!)^3$ elements. Obviously, this is a lot larger than what is necessary to preclude a brute-force attack. However, as we show in the sequel, attacking the sorting sequences is very easy compared to attacking the secret initial conditions.

The horizontal shuffling of bits within the ith pixel of $Aerb$ uses the sorting sequence that orders the four numbers $Fx_1(i), Fx_2(i), Fx_3(i), Fx_4(i)$. Let us denote this sequence by h_i. When the sequences Fx are known, h_i can be trivially constructed. Even when the attacker does not know Fx, if he can devise a method to reveal the sorting sequences $h_i, 1 \leqslant i \leqslant N$, it will be enough to break the horizontal shuffling stage of the algorithm.

Therefore, we can treat the cryptosystem as if it has two sets of independent secret parameters; the set of 12 sorting sequences Fx, Fy, Fz used in the vertical shuffling and the set of N sorting sequences $h_i, 1 \leqslant i \leqslant N$, used in horizontal shuffling.

We note that the vertical and horizontal shuffles do not separate the bit pairs (1,2), (3,4), (5,6) and (7,8). Namely, the 1st and the 2nd bits of a pixel are shuffled together and so on. Hence, we can define a new $N \times 4$ plain image P, where each entry $P(i,j)$ takes values in $\{0,1,2,3\}$.

The encryption first shuffles each column of P within itself. Let us define four permutation functions $\pi_1, \pi_2, \pi_3, \pi_4$ corresponding to Fx_1, Fx_2, Fx_3 and Fx_4, respectively. Namely, the first column of P is shuffled using the permutation π_1 and so on.

Denote by M the vertically shuffled image. Hence, we have

$$M(\pi_j(i),j) = P(i,j), \quad 1 \leqslant i \leqslant N, \quad 1 \leqslant j \leqslant 4. \tag{3}$$

Let us also define the horizontal permutation σ_i corresponding to the sorting sequence h_i. Namely, the ith row of the intermediate image M is shuffled using the permutation σ_i. Thus, we have

$$C(i,\sigma_i(j)) = M(i,j), \quad 1 \leqslant i \leqslant N, \quad 1 \leqslant j \leqslant 4. \tag{4}$$

Since the horizontal shuffles permute the row $\pi_j(i)$, using (3) and (4), we obtain the overall expression for the encryption as

$$C(\pi_j(i), \sigma_{\pi_j(i)}(j)) = P(i,j), \quad 1 \leqslant i \leqslant N, \quad 1 \leqslant j \leqslant 4. \tag{5}$$

In this new representation, P is the $N \times 4$ plain image and C is the ciphered image. Comparing with expression of the algorithm in (1), P is the two-bit stuck version of *Arb*, e.g. if *Arb* has [0,1,1,0,1,1,1,0] in a row, the corresponding row in P is [1,2,3,2]. Note that this is the same as expressing binary matrix *Arb*, in base 4, i.e. $P \in Z_4^{N\times 4}$.

3.2. Chosen-plaintext attack

In the chosen-plaintext attack, the attacker chooses a plain image and somehow obtains the corresponding ciphered image. By analyzing the plain-ciphered image pair, he tries to reveal the secret parameters.

Since each color component of the plain image is processed independently, we give the attack only for the red component. In this case, the attack reveals the permutations $\pi_1, \pi_2, \pi_3, \pi_4$ and $\sigma_i, 1 \leqslant i \leqslant N$. The other color components are analyzed similarly to reveal the rest of the sorting sequences.

For the purpose of the attack, we can take the plain image as the $N \times 4$ matrix P and the ciphered image as the $N \times 4$ matrix C.

3.2.1. Revealing the horizontal sorting sequences

In order to bypass the vertical shuffling, the attacker chooses the original plain image A such that P satisfies

$$P(i,1) = 0, \quad P(i,2) = 1, \quad P(i,3) = 2, \quad P(i,4) = 3, \quad 1 \leqslant i \leqslant N. \tag{6}$$

Namely, all the entries in the jth column of P has the value $j - 1$. Note that this corresponds to choosing the original image A with each pixel of all three color components equal to 27.

Obviously, P remains unchanged under vertical column shuffling. Hence,

$$M = P.$$

The horizontal shuffling σ_i permutes the ith row of P. Since each entry in the row is distinct, the location of the entries in the ith row of C reveals the permutation σ_i.

In order to better see how this choice of plaintext reveals the secret permutation σ_i, assume that the attacker observes

$$C(i) = 3, 0, 2, 1,$$

at the ciphertext red component for a particular row i. Comparing this with (6), the attacker sees that

$$\sigma_i = (2, 4, 3, 1),$$

i.e. $\sigma_i(1) = 2, \sigma_i(2) = 4, \sigma_i(3) = 3, \sigma_i(4) = 1$.

Similarly, by comparing every row of C with (6), the attacker reveals $\sigma_i, 1 \leqslant i \leqslant N$.

3.2.2. Revealing the vertical sorting sequences

Now that the attacker knows $\sigma_i, 1 \leqslant i \leqslant N$, he can invert the horizontal shuffling. Hence, once the attacker observes C, he can obtain the output M of the vertical shuffling operation.

This time the attacker chooses the plain image P such that

$$\begin{aligned} &P(i) = 0, 0, 0, 0, \\ &P(i+1) = 1, 1, 1, 1, \\ &P(i+2) = 2, 2, 2, 2, \end{aligned} \tag{7}$$

and all the other entries of P are identically 3. Comparing P with M that he obtained by the inverse horizontal shuffling of C, the attacker reveals the permutations $\pi_1, \pi_2, \pi_3, \pi_4$. In order to see how this is done, assume that the attacker observes that for some i_1

$$C(i_1, 1) = 0.$$

Then, the attacker concludes that

$$\pi_1(i) = i_1.$$

Likewise, if the attacker observes that for some i_2

$$C(i_2, 4) = 1,$$

he concludes that

$$\pi_4(i+1) = i_2.$$

In general, for $v \in \{0, 1, 2\}$, if the attacker observes that $C(i_0, j) = v$, then he infers that

$$\pi_j(i + v) = i_0.$$

Continuing in the same fashion, the entries $i, i+1$ and $i+2$ of $\pi_1, \pi_2, \pi_3, \pi_4$ are revealed.

For each $i \in \{1, 4, 7, \ldots\}, i < N$, the attacker chooses a plain image using (7) and obtains the corresponding ciphered image. In each case, he reveals 12 entries (4 entries for each of the three color components) of the secret sorting sequences. In total, the attacker needs $\lceil N/3 \rceil$ chosen plain images to reveal the vertical shuffling parameters. Considering the single plain image used in revealing σ_i's, the attack takes a total of $\lceil N/3 \rceil + 1$ chosen plain images.

3.3. Known-plaintext attack

In some cases, it might be impossible for the attacker to choose the plain images but instead the attacker may know some pairs of (plain/ciphered) images. Extracting information about the secret parameters using known plaintexts is known as known-plaintext attack.

In this section, we assume that the attacker somehow obtains some (P, C) pairs. The aim of the attack is to reveal the secret parameters $\pi_1, \pi_2, \pi_3, \pi_4$ and $\sigma_i, 1 \leqslant i \leqslant N$.

3.3.1. Revealing the vertical shuffling sequences

Suppose the attacker knows t (plain/ciphered) images pairs $(P_1, C_1), (P_2, C_2), \ldots, (P_t, C_t)$. Further assume that the attacker observes that for a particular plain image coordinates (i,j) and a particular pair (P_k, C_k),

$$P_k(i,j) = C_k(i_1, j_1) = C_k(i_2, j_2) = \cdots = C_k(i_{r_k}, j_{r_k}).$$

Namely, the value of the plaintext at $P_k(i,j)$ appears in the r_k ciphertext locations $(i_1, j_1), (i_2, j_2), \ldots, (i_{r_k}, j_{r_k})$. So, the permutation π_j must have mapped i to one of $i_1, i_2, \ldots, i_{r_k}$. Hence,

$$\pi_j(i) \in R_k = \{i_1, i_2, \ldots, i_{r_k}\}.$$

Using similar observations for the other pairs, we obtain other sets that include $\pi_j(i)$. Intersecting these sets R_k, we have

$$\pi_j(i) \in R = \bigcap_{k=1}^{t} R_k.$$

If the set R contains a single point, then this means that the attacker pinned down $\pi_j(i)$. If not, he needs more pairs of plain and ciphered images.

Repeating this set intersection method for all $i, j, 1 \leqslant i \leqslant N$, $1 \leqslant j \leqslant 4$, the attacker reveals all the secret quantities $\pi_j(i)$.

3.3.2. Revealing the horizontal shuffling sequences

Once the attacker reveals the 4 permutations π_1, π_2, π_3 and π_4, he goes on to reveal the horizontal permutation functions $\sigma_1, \sigma_2, \ldots, \sigma_N$.

Note that each horizontal permutation is defined over the set {1,2,3,4}.

The attacker uses the vertical permutations to obtain the intermediate images $M_1, M_2, \ldots, M_t$. The relation between M_k and C_k is given as

$$C_k(i,\sigma_i(j)) = M_k(i,j), \quad 1 \leqslant i \leqslant N, \quad 1 \leqslant j \leqslant 4.$$

For each i, the attacker constructs the sets $S_k = \{j_1, j_2, \ldots, j_{s_k}\}$ which satisfy

$$M_k(i,j) = C_k(i,j_1) = C_k(i,j_2) = \cdots = C_k(i,j_{s_k}).$$

Note that S_k has at most 4 elements.

Thus the attacker knows that

$$\sigma_i(j) \in \{j_1, j_2, \ldots, j_{s_k}\}.$$

Intersecting these sets, the attacker pins down $\sigma_i(j)$ using

$$\sigma_i(j) \in S = \bigcap_{k=1}^{t} S_k.$$

4. Simulations

In this section we illustrate the success of our proposed attacks using numerical examples.

4.1. Chosen-plaintext attack

Since each color component is encrypted separately, we illustrate the attack on the red component of a 3 × 2 image. In this case, $m = 3, n = 2$ and so $N = 6$. Our attack aims to reveal the permutations $\pi_1, \pi_2, \pi_3, \pi_4$, and $\sigma_1, \sigma_2, \ldots, \sigma_6$. For the purpose of illustration, we generate the permutations randomly.

First, the attacker chooses the 3 × 2 image Ar given in Fig. 1a. He obtains the corresponding ciphered image in Fig. 1b. The plain and ciphered images P and C of the equivalent representation are given in Fig. 1c and d, respectively.

Note that the rows of P are shuffled in C. Inspecting the shuffling patterns, the attacker reveals that $\sigma_1 = (4, 1, 3, 2), \sigma_2 = (4, 2, 1, 3), \sigma_3 = (2, 3, 1, 4)$ and so on.

This time, the attacker chooses the plain image given in Fig. 2a. This plain image corresponds to P given in Fig. 2c. The attacker observes the corresponding ciphered image given in Fig. 2b. This image corresponds to the image C given in Fig. 2d. Since the attacker knows the horizontal permutations σ_i, by applying their inverses on C, he obtains the intermediate image M given in Fig. 2e.

By comparing P in Fig. 2c to M in Fig. 2e, the attacker concludes that $\pi_1(1) = 6$, $\pi_1(2) = 3$, $\pi_1(3) = 5$, $\pi_2(1) = 5$, $\pi_2(2) = 1$, $\pi_2(3) = 2$ and so on. By choosing another plaintext with the rows of the one in Fig. 2a swapped, the attacker reveals the rest of the vertical permutations.

27	27	27
27	27	27

(a)

120	156	135
156	75	39

(b)

0	1	2	3
0	1	2	3
0	1	2	3
0	1	2	3
0	1	2	3
0	1	2	3

(c)

1	3	2	0
2	1	3	0
2	0	1	3
2	1	3	0
1	0	2	3
0	2	1	3

(d)

Fig. 1. (a) The chosen plain image Ar. (b) Corresponding ciphered image Br. (c) The plain image P in equivalent representation. (d) The ciphered image C in equivalent representation.

0	85	170
255	255	255

(a)

111	175	221
51	39	63

(b)

0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
3	3	3	3
3	3	3	3

(c)

1	2	3	3
2	2	3	3
3	1	3	1
0	3	0	3
0	2	1	3
0	3	3	3

(d)

3	1	3	2
3	2	2	3
1	3	3	1
3	3	0	0
2	0	1	3
0	3	3	3

(e)

Fig. 2. (a) The chosen plain image Ar. (b) Corresponding ciphered image Br. (c) The plain image P in equivalent representation. (d) The ciphered image C in equivalent representation. (e) The intermediate image M obtained by inverse horizontal shuffling of C.

4.2. Known-plaintext attack

In this case, assume that $m = n = 256$ and that the attacker knows six 256 × 256 randomly generated plain images and their corresponding ciphered images. Thus, $t = 6$.

By constructing the candidate sets $R_k, 1 \leqslant k \leqslant 6$ and taking their intersections, all the values of $\pi_j(i), 1 \leqslant j \leqslant 4, 1 \leqslant i \leqslant 256^2$ are determined.

The distribution of the number of intersected sets is as follows. Out of 262,144 unknown values for the permutations π_1, π_2, π_3, and π_3, 204,828 are pinned down with only 3 intersections. This makes about 78% of all the unknowns. 56,301 are determined using 4 intersections. Thus, we see that for 99.8% of the permutation values, less than 4 intersections were enough. 993 of the values required 5 intersections. Only 22 values required 6 intersections.

Once the vertical permutations are known, determining the horizontal permutations σ_i is similar and it is done by using set intersections.

The attack takes less than 5 h under MATLAB running on Mac OS X 10.5.7 with Intel Core 2 Duo 2.33 GHz processor and 3.3 GB RAM.

5. Conclusion

In this paper, we have cryptanalysed a recently proposed image cryptosystem by two different attacks. The weakness of this cryptosystem arise from the use of the same shuffling process for every plain image. And that is a consequence of using the same sequences generated by the four chaotic systems.

Acknowledgment

Ercan Solak is supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under Project No. 106E143.

References

[1] G. Alvarez, S.J. Li, Int. J. Bifurcat. Chaos 16 (2006) 2129.

[2] C.K. Huang, H.H. Nien, Opt. Commun. 282 (2009) 2123.

[3] S. Mazloom, A.M. Eftekhari-Moghadam, Chaos Solitons Fractals 42 (2009) 1745.

[4] J.K. Hu, F.L. Han, J. Netw. Comput. Appl. 32 (2009) 788.
[5] M.R.K. Ariffin, M.S.M. Noorani, Phys. Lett. A 372 (2008) 5427.
[6] V. Patidar, N.K. Pareek, K.K. Sud, Commun. Nonlinear Sci. Numer. Simul. 14 (2009) 3056.
[7] A.N. Pisarchik, N.J. Flores-Carmona, M. Carpio-Valadez, Chaos 16 (2006) 033118.
[8] T.G. Gao, Z.Q. Chen, Chaos Solitons Fractals 38 (2008) 213.
[9] T.G. Gao, Z.Q. Chen, Phys. Lett. A 372 (2008) 394.
[10] R. Rhouma, E. Solak, D. Arroyo, S. Li, G. Alvarez, S. Belghith, Phys. Lett. A. doi:10.1016/j.physleta.2009.07.035.
[11] R. Rhouma, E. Solak, S. Belghith, Commun. Nonlinear Sci. Numer. Simul. doi:10.1016/j.cnsns.2009.07.007.
[12] E. Solak, C. Cokal, Chaos 18 (2008) 038101.
[13] D. Arroyo, R. Rhouma, G. Alvarez, S.J. Li, V. Fernandez, Chaos 18 (2008) 033112.
[14] S.J. Li, C.Q. Li, G.R. Chen, N.G. Bourbakis, K.T. Lo, Signal Process. Image Commun. 23 (2008) 212.