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

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.
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. $\text{Arb}, \text{Agb}, \text{Abb} \in (0,1)^{N \times 8}$ 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 $F_x, F_y, F_z$ respectively. Likewise, $x_1, y_1, z_1$ are generated by the Lorenz system, $x_2, y_2, z_2$ are generated by the Henon chaotic system. Also note that each sorting sequence defines a permutation over the set of integers $\{1, 2, \ldots, N\}$. The horizontal shuffling of bits within the $i$th pixel of $\text{Aerb}$ uses the sorting sequence that orders the numbers $F_x(i), F_y(i), F_z(i)$, respectively. After the horizontal shuffling, we end up with $\text{Aerb}, \text{Agb}$ and $\text{Abb}$.

2.3. Encryption

2.3.1. Vertical shuffling

Each column of the matrix $\text{Arb}$ is shuffled using one of the sequences $F_x, F_y, F_z$ to generate the intermediate matrix $\text{Aerb}$. The choice of the shuffling sequence is determined as follows:

$$\text{Aerb}(i,j) = \begin{cases} \text{Arb}(F_x(i), j), & j = 1, 2, \\ \text{Arb}(F_y(i), j), & j = 3, 4, \\ \text{Arb}(F_z(i), j), & j = 5, 6, \\ \text{Arb}(F_y(i), j), & j = 7, 8. \end{cases}$$

(1)

The horizontal shuffling is applied to the green and blue components $\text{Agb}$ and $\text{Abb}$ with the shuffling sequences $F_y, F_x, F_z$ and $F_z, F_x, F_y$, respectively. After the vertical shuffling, we use $\text{Aerb}, \text{Agb}$ and $\text{Abb}$.

2.3.2. Horizontal shuffling

In this step, the rows of $\text{Aerb}, \text{Agb}$ and $\text{Abb}$ are shuffled horizontally. This corresponds to the shuffling of pixel bits among themselves.

The red component $\text{Br}$ of the ciphered image $B$ is obtained by first concatenating the bits in each row of $\text{Brb}$ 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 $\text{Aegb}$ and $\text{Aebb}$ with the sorting sequences derived from $F_y$ and $F_z$ respectively.

The final ciphered image $B$ is obtained by joining all three color components $\text{Br}, \text{Bg}$ and $\text{Bb}$.

3. Cryptanalysis

In this section, we give chosen-plaintext and known-plaintext attacks against the cryptosystem. Both of the attacks yield the sorting sequences $F_x, F_y$ and $F_z$ 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 $F_x, F_y$ and $F_z$, 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 $(N!)^4$ 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 $i$th pixel of $\text{Aerb}$ uses the sorting sequence that orders the numbers $F_x(i), F_y(i), F_z(i)$, respectively. Let us denote this sequence by $h_i$. When the sequences $F_x$ are known, $h_i$ can be trivially constructed. Even when the attacker does not know $F_x$, if he can devise a method to reveal the sorting sequences $h_i$, $1 \leq i < N$, 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 $F_x, F_y, F_z$ used in the vertical shuffling and the set of $N$ sorting sequences $h_i$, $1 \leq i < N$, 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 $F_x, F_y, F_z$ and $F_{z}$, respectively. Namely, the first column of $P$ is shuffled using the permutation $\pi_1$ and so on.

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

$$M(\pi_i(i,j)) = P(i,j), \quad 1 \leq i < N, \quad 1 \leq j \leq 4.$$  (3)

Let us also define the horizontal permutation $\sigma_i$ corresponding to the sorting sequence $h_i$. Namely, the $i$th row of the intermediate image $M$ is shuffled using the permutation $\sigma_i$. Thus, we have

$$C(i, \sigma_i(j)) = M(i,j), \quad 1 \leq i < N, \quad 1 \leq j \leq 4.$$  (4)

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

$$C(\pi_i(i), \sigma_i(j)) = P(i,j), \quad 1 \leq i < N, \quad 1 \leq j \leq 4.$$  (5)
3.2. Revealing the horizontal sorting sequences

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 \([0, 1, 0, 1, 1, 1, 0]\) 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. \( P \in \mathbb{Z}^{N \times 4}_{4} \).

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

\[
P(i, 1) = 0, \quad P(i, 2) = 1, \quad P(i, 3) = 2, \quad P(i, 4) = 3, \quad 1 \leq i \leq N.
\]

Namely, all the entries in the \( j \)th 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 \( \sigma_i \) permutes the \( i \)th row of \( P \). Since each entry in the row is distinct, the location of the entries in the \( i \)th row of \( C \) reveals the permutation \( \sigma_i \).

In order to better see how this choice of plaintext reveals the secret permutation \( \sigma_i \), 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 \( \sigma_i, 1 \leq i \leq N \).

3.2.2. Revealing the vertical sorting sequences

Now that the attacker knows \( \sigma_i, 1 \leq i \leq N \), 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 \( \pi(i) \) such that

\[
P(i) = 0, 0, 0, 0, \quad P(i + 1) = 1, 1, 1, 1, \quad P(i + 2) = 2, 2, 2, 2,
\]

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 \( i \), \( C(i_1, 1) = 0 \).

Then, the attacker concludes that \( \pi_1(i) = i_1 \).

Likewise, if the attacker observes that for some \( i \), \( C(i_2, 4) = 1 \), 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 \( \pi_v(i + v) = i_0 \).

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 \( \lfloor N/3 \rfloor \) chosen plain images to reveal the vertical shuffling parameters. Considering the single plain image used in revealing \( \sigma_i \)’s, the attack takes a total of \( \lfloor N/3 \rfloor + 1 \) 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 \( \sigma_i, 1 \leq i \leq N \).

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 \( (P_k, C_k) \),

\[
P_k(i, j) = C_k(i_1, j_1) = C_k(i_2, j_2) = \cdots = C_k(i_{t-1}, j_{t-1}).
\]

Namely, the value of the plaintext at \( t \) ciphered text locations \( (i_1, j_1), (i_2, j_2), \ldots, (i_{t-1}, j_{t-1}) \). So, the permutation \( \pi_j \) must have mapped \( i \) to one of \( i_1, i_2, \ldots, i_{t-1} \). Hence,

\[
\pi_j(i) \in R_k = \{i_1, i_2, \ldots, i_{t-1}\}.
\]

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 \( i, j, 1 \leq i \leq N, 1 \leq j \leq 4 \), 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 \( \pi_1, \pi_2, \pi_3, \pi_4 \), he goes on to reveal the horizontal permutation functions \( \pi_1, \pi_2, \pi_3, \pi_4 \).

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 \( M_1, M_2, \ldots, M_4 \). The relation between \( M_k \) and \( C_k \) is given as...
\[ C_i(i, \sigma(j)) = M_k(i, j), \quad 1 \leq i \leq N, \quad 1 \leq j \leq 4. \]

For each \( i \), the attacker constructs the sets \( S_k = \{j_1, j_2, \ldots, j_k\} \) which satisfy

\[ M_k(i, j) = C_k(i, j_1) = C_k(i, j_2) = \cdots = C_k(i, j_k). \]

Note that \( S_k \) has at most 4 elements.

Thus the attacker knows that

\[ \sigma(j) \in \{j_1, j_2, \ldots, j_k\}. \]

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

\[ \sigma(j) \in S = \bigcap_{k=1}^{4} S_k. \]

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 \( 3 \times 2 \) 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 \( 3 \times 2 \) 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 \( \sigma_i \), 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

\[
\begin{array}{cccc}
27 & 27 & 27 & 120 \ 156 \ 135 \\
27 & 27 & 27 & 156 \ 75 \ 39 \\
\end{array}
\]

(a) (b)

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
0 & 1 & 2 & 3 \\
0 & 1 & 2 & 3 \\
0 & 1 & 2 & 3 \\
0 & 1 & 2 & 3 \\
0 & 1 & 2 & 3 \\
\end{array}
\]

(c) (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.

4.2. Known-plaintext attack

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

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

The distribution of the number of intersected sets is as follows. Out of 262,144 unknown values for the permutations \( \pi_1, \pi_2, \pi_3, \pi_4 \), and \( \pi_5, \pi_6, \pi_{204,828} \) 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 \( \sigma_i \) 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