Chapter 7
Cryptanalysis of Chaotic Ciphers

Ercan Solak

Işık University
Istanbul, Turkey
ercan@isikun.edu.tr

1 Introduction

Cryptanalysis is an integral part of any serious effort in designing secure encryption algorithms. Indeed, a cryptosystem is only as secure as the most powerful known attack that failed to break it. The situation is not different for chaos-based ciphers. Before attempting to design a new chaotic cipher, it is essential that the designers have a thorough grasp of the existing attacks and cryptanalysis tools.

There is a large variety of chaotic ciphers proposed in the literature. Consequently, their cryptanalyses come up with equally diverse attacks. Each attack tries to exploit weaknesses that are specific to the particular chaotic cipher. Thus, it is somewhat difficult to devise common non-trivial attacks that can be applied against a range of chaotic ciphers. On the other hand, such diversity of designs works against the security of the chaos-based ciphers. Rather than using well-analyzed and tested building blocks, there seems to be a general tendency to try novel and fancier structures, thus opening new venues for attacks.

If chaos cryptography is to make serious contributions to mainstream cryptography, we need to have more of analysis and less of design. Rather than trying to come up with new and interesting ways to incorporate chaos into encryption, the research effort should try to establish ground rules and primitive building blocks for the use of chaos in cryptography. This can only come through a rigorous cryptanalysis of existing proposals and by identifying the common weaknesses and pitfalls.

There have been a few noteworthy efforts in this direction. In particular, [Alvarez and Li, 2006] offer general observations about the flaws and weaknesses found in many chaotic encryption schemes. [Amigó et al., 2007, Masuda et al., 2006, Kocarev and Jakimoski, 2003, Dachselt and Schwarz, 2001] identify the building blocks that can be used in chaotic ciphers and random number generators. [Anstett et al., 2006] draws parallels between identifiability of dynamical systems and cryptanalysis. [Li et al., 2008]

L. Kocarev and S. Lian (Eds.): Chaos-Based Cryptography, SCI 354, pp. 227–256.
springerlink.com

establishes general attacks that can be launched against permutation-only chaotic image encryption algorithms.

In many proposals for chaotic ciphers, we observe a common tendency to use a subset of statistics in order to demonstrate the strength of the encryption. Although a necessary condition, good statistics are far from establishing good encryption. Indeed, any mildly sophisticated function produces good confusion and diffusion when applied in enough number of rounds. What statistics can not do, however, is hide the algebraic weaknesses inherent in the cipher. For example, even a linear block cipher will pass some easy statistical tests. Yet, a linear cipher can trivially be broken.

Therefore, it is crucially important to analyze the algebraic structure of a chaotic cipher and identify weak transformations.

A particular class of attacks against chaos based ciphers aims at bypassing the chaotic part of the cryptosystem. In this class, the encryption algorithm is expressed in an equivalent form in which the chaotic subsystems are replaced by a set of secret maps or parameters. In this way, the algebraic weaknesses in the rest of the algorithm are highlighted. This approach makes the whole system more amenable to cryptanalysis. In this chapter on the cryptanalysis of chaos-based ciphers, we illustrate the power of algebraic attacks on a number of different chaotic encryption algorithms.

In the next section, we examine the case of "inadvertently" linear ciphers. Such a cipher uses the nonlinear nature of chaos to generate some key parameters. However, the transformation from the plain image to the cipher image is linear.

The final part of the chapter illustrates the power of algebraic analysis in breaking chaotic ciphers.

2 Chaotic *Linear* Ciphers

Before we identify a few chaotic ciphers that turns out to be linear, we briefly show how a linear block cipher can be trivially broken.

Assume P and C are n-bit plaintext and ciphertext blocks, respectively. If the encryption transformation from P to C is linear, then it can be represented as a binary matrix multiplication

$$C = \mathbf{A}P, \tag{1}$$

where the matrix $\mathbf{A}$ is the secret mapping. For a known plaintext block P, an attacker can construct n linear equations

$$\begin{aligned} c_1 &= a_{11}p_1 \oplus a_{12}p_2 \oplus \cdots \oplus a_{1n}p_n, \\ c_2 &= a_{21}p_1 \oplus a_{22}p_2 \oplus \cdots \oplus a_{2n}p_n, \\ &\vdots \\ c_n &= a_{n1}p_1 \oplus a_{n2}p_2 \oplus \cdots \oplus a_{nn}p_n \end{aligned}$$

for the entries a_{ij} of $\mathbf{A}$.

Using a set of n distinct known plaintext-ciphertext pairs, an attacker can construct n^2 linear equations for n^2 secret entries of $\mathbf{A}$. Solving these linear equations, the attacker easily breaks the cipher.

A common weakness in many chaotic ciphers is to use a set of well-known chaotic systems with secret system parameters to generate a linear transformation $\mathbf{A}$, which is then used as in (1). This creates a complex relationship between the chaotic system parameters and the resulting linear transformation. However, the attacker bypasses this complexity by attacking the linear transformation rather than trying to reveal the secret system parameters. The situation is illustrated in Fig. 1.

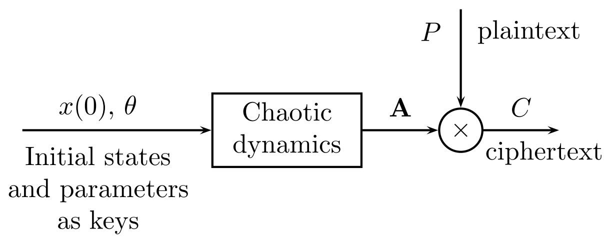

Fig. 1 A general structure of a chaotic *linear* block cipher.

Example 1. In the chaos-based image cipher proposed in [Guan et al., 2005], the encryption process consists of two parts. In the first part, the algorithm takes an image P and shuffles its pixels using Arnold Cat map. The second part of the algorithm changes the gray levels of the pixels using Chen's chaotic system.

Representing the image as a vector, the shuffling transformation can be represented as

$$S = \mathbf{A}P,$$

where $\mathbf{A}$ is a secret permutation matrix. For the second step of the encryption, Chen's chaotic system is used with secret parameters and initial values to generate a key vector, K. Thus, the encryption can be written as

$$C = \mathbf{A}P \oplus K. \tag{2}$$

Clearly, (2) is an affine linear equation. Assume that the attacker knows two plaintext-ciphertext image pairs (P_1, C_1) and (P_2, C_2). Let us define the differences as $\Delta P = P_1 \oplus P_2$ and $\Delta C = C_1 \oplus C_2$. Using (2), the attacker calculates

$$\Delta C = \mathbf{A}\Delta P$$

Going from ΔP to ΔC, there is only shuffling by the Arnold Cat map, which is a linear operation.

For a number of known plaintext-ciphertext differences, the attacker can find the secret $\mathbf{A}$. Once he reveals $\mathbf{A}$, he uses just one known pair (P, C) to calculate the secret K as

$$K = C \oplus \mathbf{A}P.$$

It is possible to improve the attack if one allows for chosen plaintexts. For more details, see [Çokal and Solak, 2009].

Although the attack is quite simple, it can be applied to a number of chaotic ciphers with only a few adaptations.

In [Patidar et al., 2009], a plaintext image P is encrypted in four steps. The first and the last steps involve adding chaotically generated key images K_1 and K_2. The second step linearly diffuses the pixel values in horizontal direction. The third step does the same in vertical direction. The two diffusions can be combined into one matrix multiplication. Thus, the whole encryption process becomes

$$C = \mathbf{A}(P \oplus K_1) \oplus K_2.$$

Clearly, this is a linear transformation. Moreover, the parameter A is not secret. This makes the whole scheme trivially weak. More details on the attack can be found in [Rhouma et al., 2010].

A general class of chaotic linear ciphers are shuffling-only image ciphers. In many cases, the shuffling parameters are generated by iterating one or more chaotic systems starting with secret initial conditions and parameters. In attacking these systems, the attacker aims to reveal the intermediate shuffling parameters rather than the chaotic system parameters. A recent example of such a cipher is proposed in [Huang and Nien, 2009], which is cryptanalyzed in [Solak et al., 2010b]. A general approach in attacking substitution-only image ciphers is given in [Li et al., 2008].

3 Algebraic Attacks

The mapping from the chaotic system parameters and initial conditions to its trajectories is highly nonlinear and complex. Still, when a chaotic system is used in encryption, the algebraic structures that it induces might be amenable to cryptanalysis. In the following discussion, we analyze three chaotic ciphers in order to illustrate the power of algebraic analysis in attacks.

3.1 Reconstructing Small Permutations

We first give a few facts about the powers of permutations over finite sets.

Definition 1. *[Fraleigh, 2002] An ordered orbit of a permutation π on a finite set is the ordered tuple $(a_0, a_1, \ldots, a_{n-1})$ such that $\pi(a_0) = a_1$, $\pi(a_1) = a_2, \cdots, \pi(a_{n-2}) = a_{n-1}$, $\pi(a_{n-1}) = a_0$. n is the length of the ordered-orbit.*

Theorem 1. *[Fraleigh, 2002] A permutation defined on a finite set partitions the set into disjoint ordered-orbits.*

Remark 1. Given a permutation π defined on a set V, determining its orbits is straightforward. We start from any element $a_0 \in V$ and form the orbit elements as $(a_0, \pi(a_0), \pi^2(a_0), \ldots, \pi^{n-1}(a_0))$ until $\pi^n(a_0) = a_0$. We then start over with an element not included in the orbits found so far. We continue forming orbits until we exhaust all the elements in the set V.

An example of a permutation over the set $\{a_0, a_1, \ldots, a_{10}\}$ is given in Fig. 2. Note that there are two orbits of lengths 5 and 6.

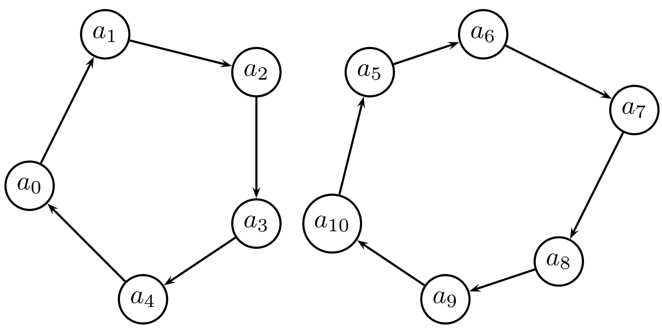

Fig. 2 A permutation with two orbits of lengths 5 and 6.

Note that if a_0 is an element in an orbit of length n in the permutation π, then, for all integers i,

$$\pi^i(a_0) = \pi^{i \bmod n}(a_0).$$

Lemma 1. *Let* $\alpha = (a_0, a_1, \ldots, a_{n-1})$ *be an orbit of length* n *in the permutation* π, *where* $\gcd(n, r) = v$. *Then,* α *is split into* v *equal length orbits in* π^r.

Lemma 2. *Let* $\beta = (b_0, b_1, \ldots, b_{t-1})$ *be the only orbit of length* t *in the permutation* π^r. *Then,*

$$\pi(b_j) = b_{(j+r^*) \bmod t}, \quad 0 \le j < t,$$

where r^* *is the multiplicative inverse of* r *in* $\bmod t$, *i.e.* $rr^* \equiv 1 \pmod t$.

Remark 2. An immediate result of Lemma 1 and Lemma 2 is that if we have an orbit $\alpha = (a_0, a_1, \ldots, a_{n-1})$ in π such that $\gcd(n, r) = 1$, then in π^r, α is not split but is rather shuffled as

$$\beta = (a_0, a_{r \bmod n}, a_{(2r) \bmod n}, \ldots, a_{((n-1)r) \bmod n}).$$

Lemma 3. *Let* $\beta = (b_0, b_1, \ldots, b_{t-1})$ *be one of the* q *orbits of length* t *in the permutation* π^r. *Let* v *be the least divisor of* r *larger than 1. Assume that* $q < v$. *Then,*

$$\pi(b_j) = b_{(j+r^*) \bmod t}, \ 0 \le j < t, \tag{3}$$

where r^* *is the multiplicative inverse of* r *in* $\bmod t$, *i.e.* $rr^* \equiv 1 \pmod t$.

Lemma 4. *Let* $\beta^{(1)} = (b_0^{(1)}, b_1^{(1)}, \ldots, b_{t-1}^{(1)})$ *and* $\beta^{(2)} = (b_0^{(2)}, b_1^{(2)}, \ldots, b_{t-1}^{(2)})$ *be two orbits of length* t *in* π^r. *If* $\pi(b_i^{(1)}) = b_j^{(2)}$ *for some* i, j *then*

$$\pi(b^{(1)}_{(i+k) \bmod t}) = b^{(2)}_{(j+k) \bmod t}, \ 1 \le k < t.$$

For the proofs of these lemmas, see [Solak and Çokal, 2009].

An illustration of how orbits are shuffled and split in powers of permutation is given in Fig. 3. The graph shows the orbit structure of π^2 of the permutation π given in Fig. 2. Note that the length 5 orbit of π is only shuffled while its length 6 orbit is split into two length 3 orbits.

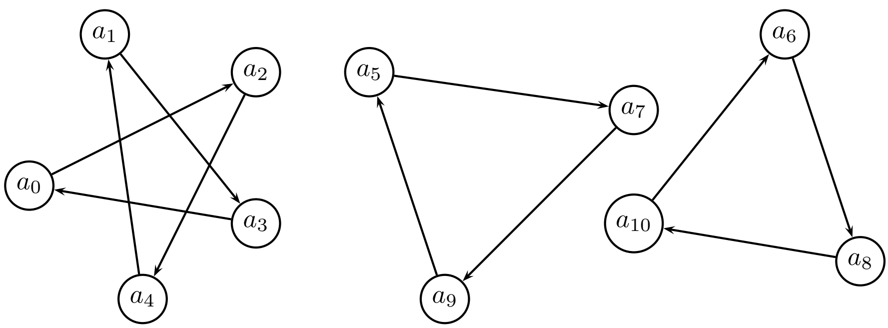

Fig. 3 Orbits of π^2 for the permutation π given in Fig. 2.

We know apply these properties of permutations to design algebraic attacks against two chaotic block ciphers.

3.2 Algebraic Attack on a Cryptosystem Based on Discretized Two-Dimensional Chaotic Maps

In the chaotic cipher proposed in [Xiang et al., 2007], plaintext and ciphertext sequences are partitioned into 16-bit blocks P_i, C_i, $1 \le i \le n$, as

$$\text{Plaintext}: \ P_1 P_2 \cdots P_n,$$
$$\text{Ciphertext}: \ C_1 C_2 \cdots C_n.$$

The key of the cryptosystem is the collection of the parameters (r, m, t, C_0, K_s, K_c). In [Xiang et al., 2007] this collection is defined as the master key. The master key is composed of the number of rounds r, the shift amount m, the number of iterations t, the initial value C_0, the subkey K_s and the collection of TDCM parameters K_c. Below, we explain how each part of the key is used in encryption.

A block key K_i is used in the encryption of plaintext block P_i. Initially, we assign

$$K_0 = K_s. \tag{4}$$

Before the encryption of block P_i, K_i is first updated as

$$K_i = \begin{cases} K_{i-1} \oplus C_{i-1} & \text{if } C_{i-1} \neq K_{i-1}, \\ K_{i-1} & \text{if } C_{i-1} = K_{i-1}. \end{cases} \tag{5}$$

The encryption of the i^{th} block is given as

$$C_i = E(K_i, P_i), \tag{6}$$

where the function E involves the following round operations.

$$\begin{aligned} v_0 &= P_i, \\ v_j &= \sigma(v_{j-1} \oplus \text{ROL}(K_i, jm)), \ 1 \leq j \leq r, \\ C_i &= v_r. \end{aligned} \tag{7}$$

Here, v_j is the output of round j. Thus, v_r becomes the ciphertext. $\text{ROL}(\cdot, jm)$ denotes the circular left rotation of its argument by jm bits. The amount of circular left shifts depends on the number of rounds r and is given as

$$m = \begin{cases} \lfloor 16/r \rfloor & r \leq 16, \\ 1 & \text{else.} \end{cases} \tag{8}$$

The round function σ is a composition of a number maps and is given as

$$\sigma = w \circ z^{-1} \circ \text{TDCM}^t_{K_c} \circ z \circ S. \tag{9}$$

In (9), S represents the S-box substitution. S invertibly maps between 16-bit quantities. The S-box is designed to have desirable nonlinear properties, and its value is fixed (not secret) for the algorithm.

The map z is an invertible function that maps from 16-bit quantities to 2D vectors of integers. It maps the unsigned integer values corresponding to each byte of its argument to one of the integer coordinates in 2D discrete state space. The aim of z is to prepare a 2D initial state out of a given 16-bit quantity.

$\text{TDCM}^t_{K_c}$ denotes the t-times iteration of TDCM. K_c denotes the collection of the chaotic system parameters. The choice of the chaotic map is part of the algorithm design. In [Xiang et al., 2007], the standard map, the generalized cat map, and the generalized baker map are considered. The chaotic map must be bijective in order to have an invertible encryption operation. The output of the chaotic system is passed through z^{-1} to map the final 2D state of TDCM to a 16-bit number.

The last mapping in w in (9) denotes the byte swap operation.

After the encryption of block i, the block key is once more updated as

$$K_i \leftarrow \text{ROL}(K_i, rm). \quad (10)$$

Since K_i is 16-bits, the effective amount of rotation on K_i in this step is rm mod 16.

We now give a detailed cryptanalysis of the cipher.

The relation (8) fixes m once r is known. This removes the freedom in the choice of m, and effectively reduces the key length by 8 bits. Therefore, the shift amount m must be treated not as a key but rather as an internal parameter that is derived from the key.

Another reduction in effective key length is due to the way the secret parameter C_0 is used. Before the encryption of the first 16-bit block, the subkey K_s is updated by using (5). Hence, the value of K_1 used in the encryption of P_1 is $K_s \oplus C_0$. Consequently, we can treat $K_s \oplus C_0$ as one secret parameter rather than two distinct parameters, K_s and C_0. Indeed, any pair of C_0 and K_s values that yields the same XOR value results in identical encryption functions. This fact reduces the effective key length by another 16 bits. In the subsequent sections, we assume without loss of generality that $C_0 = \texttt{0x0000}$.

After noting these reductions in the effective key space, we now give an algebraic break of the cipher. We first demonstrate how an attacker can reveal K_s without having access to the rest of the key parameters.

In out attacks, we assume that the attacker knows the number of rounds r. This is not a very restrictive assumption. Since r is represented with 8 bits, it can only take one of 255 possible nonzero values. The attacks that we develop in this and the next section have very low computational requirements. In the case when the attacker does not know the value of r, he tries all 255 possible values with the attacks described here.

Revealing K_s

To illustrate the method of the attack, we only analyze the case when $rm \equiv 0 \bmod 16$. For the details of the attack for the case $rm \not\equiv 0 \bmod 16$, see [Solak and Çokal, 2008].

We assume that the attacker does not know the TDCM parameters, so he does not know the function E in (6).

Assume that the first two ciphertext blocks are the same and given as

$$C_1 = C_2 = j. \quad (11)$$

If $j = K_s$, using (4), (5), (6) and (10), we have

$$j = E(K_s, P_1),\ j = E(K_s, P_2).$$

So, by the invertibility of E for fixed K_s, we have $P_1 = P_2$.

If $j \neq K_s$, we have

$$j = E(K_s, P_1),\ j = E(K_s \oplus j, P_2).$$

In this case, most probably $P_1 \neq P_2$. The difference in two cases indicates that the equality of P_1 and P_2 is a good test on whether $K_s = j$.

The attack on K_s proceeds as follows. The attacker chooses a 16-bit number j. He requests plaintexts for a two-block ciphertext C_1C_2 chosen as in (11). He compares these plaintext blocks P_1 and P_2. If they are equal, then j is a candidate for the secret K_s. The attacker repeats this for all the 16-bit j values and records candidates for K_s. A total of $2^{16} - 1$ trials are made.

It may happen that the attacker obtains $P_1 = P_2$ even when $j \neq K_s$. This is because we might have $E(K_1, P) = E(K_2, P)$ for some $K_1 \neq K_2$, and P. In order to eliminate the false keys, the attacker performs the following further tests.

Assume that the attacker has two candidates j_1 and j_2 for the subkey K_s. From his previous attempt at determining the keys, the attacker knows P_1 and P_2 which satisfy

$$j_1 = E(K_s, P_1),\ j_2 = E(K_s, P_2). \tag{12}$$

The attacker now chooses the new ciphertext blocks $\overline{C}_1$ and $\overline{C}_2$ as $\overline{C}_1 = j_1$ and $\overline{C}_2 = j_2$. He obtains the corresponding plaintext blocks $\overline{P}_1$ and $\overline{P}_2$. There are two cases for the validity of j_1. Let us see how $\overline{P}_1$ and $\overline{P}_2$ differ for each case.

Case 1: $j_1 = K_s$: Using (4), (5), (6) and (10), we find that

$$j_1 = E(K_s, \overline{P}_1),\ j_2 = E(K_s, \overline{P}_2).$$

Comparing this with (12), we obtain $\overline{P}_1 = P_1$ and $\overline{P}_2 = P_2$.

Case 2: $j_1 \neq K_s$: This time we find,

$$j_1 = E(K_s, \overline{P}_1),\ j_2 = E(K_s \oplus j_1, \overline{P}_2).$$

Comparing this with (12), we conclude $\overline{P}_1 = P_1$ and $\overline{P}_2$ is a random 16-bit number.

In both cases, $\overline{P}_1 = P_1$. However, only in the first case we are guaranteed to have $\overline{P}_2 = P_2$. In the second case, we might have $\overline{P}_2 = P_2$ even when $j_1 \neq K_s$. So, if $\overline{P}_2 \neq P_2$ the test is conclusive and $j_1 \neq K_s$. If $\overline{P}_2 = P_2$ the test is inconclusive.

This test gives the attacker a method to eliminate the false subkeys among the candidates. Assume that attacker has determined q candidates,

$\{j_1, j_2, \cdots, j_q\}$ for the subkey K_s. To eliminate the false subkeys, he chooses a pair of candidates j_{i_1} and j_{i_2} and applies the test as explained. In this way, he eliminates j_{i_1} if the test is conclusive. Otherwise, he chooses a different pair and repeats the test. The attack on K_s successfully terminates when there remains only one candidate for the subkey.

Once the attacker knows K_s, he proceeds to reveal the other parameters t and K_c. We assume that the attacker already knows the number of rounds r. Hence, by the relation (8), he also knows the shift amount m. The only secret parameters to be revealed are K_c, the collection of the TDCM parameters and t, the number of times the TDCM is iterated. When the block key K_i and r are fixed, the parameters K_c and t characterize the function E.

A brute force attack on K_c and t has to try all their values against a known plaintext-ciphertext pair. We now give a general attack that requires on the order of 2^{16} chosen ciphertext/plaintext blocks and very little amount of computation. Moreover, the computational complexity of our attack does not depend on the lengths of the keys K_c and t.

Sampling E

We first note that, for a fixed K_i of his choice, the attacker can choose either one of C or P in the relation

$$C = E(K_i, P), \tag{13}$$

and obtain the other. To see how this can be done, let us write the encryption equations for a sequence of two blocks of plaintext, $P_1 P_2$.

$$\begin{aligned} C_1 &= E(K_s, P_1), \\ C_2 &= E(\mathrm{ROL}(K_s, rm) \oplus C_1, P_2). \end{aligned} \tag{14}$$

Here, we assume that $\mathrm{ROL}(K_s, rm) \oplus C_1 \neq 0$.

If C_1 is chosen as $C_1 = K_i \oplus \mathrm{ROL}(K_s, rm)$, (14) becomes

$$C_2 = E(K_i, P_2).$$

So, the attacker first chooses a single block ciphertext with $C_1 = K_i \oplus \mathrm{ROL}(K_s, rm)$ and obtains the plaintext P_1. If he wants to choose C and obtain the corresponding P in (13), he next chooses the ciphertext sequence $C_1 C$ and obtains P_2 as his desired plaintext block P. If, instead, he wants to know C for a particular P in (13), he chooses the plaintext sequence $P_1 P$ and obtains C_2 as his desired ciphertext block C.

Thus, an attacker can freely choose K_i, and sample the function $C = E(K_i, P)$ at arbitrary points (P, C) of his choice. We will see that this ability lets the attacker determine the internal secret parameters of the encryption function E.

Since the functions w, z, S are fixed and the attacker already knows r, m, and K_i, revealing the secret parameters t, K_c is equivalent to revealing the function σ in (7). Namely, once the attacker knows σ, he can encrypt/decrypt any plaintext/ciphertext sequences as if he knew the parameters t and K_c. Below we describe three attacks that reveal the function σ.

We first note that σ is a permutation over the set $\{0, 1, \ldots, 2^{16} - 1\}$. We now show how particular choices of K_i lets an attacker reveal portions of σ.

Permutation orbit attack

Let us choose K_i such that

$$\mathrm{ROL}(K_i, m) = K_i. \tag{15}$$

Namely, K_i is m-bit rotation invariant.

When we use (15) in (7), we obtain

$$v_j = \sigma(v_{j-1} \oplus K_i),\ 1 \leq j \leq r.$$

Defining a new permutation π as

$$\pi(x) = \sigma(x \oplus K_i) \tag{16}$$

for $x \in \{0, 1, \ldots, 2^{16} - 1\}$, we can express the relation between P and C as

$$C = \underbrace{\pi \circ \pi \circ \cdots \circ \pi}_{r \text{ times}}(P) = \pi^r(P).$$

If the attacker reveals the value Y of π at P so that $Y = \pi(P)$, he reveals that the value of σ at $P \oplus K_i$ is Y, i.e. $Y = \sigma(P \oplus K_i)$.

To illustrate the choice of K_i that turns the function E into the $r-$power of a permutation, let us take $m = 2$. In this case, the nonzero K_i values that satisfy (15) are `0101010101010101` (`0x5555`), `1010101010101010` (`0xAAAA`) and `1111111111111111` (`0xFFFF`) . If $m = 1$, the only nonzero K_i that satisfies (15) is (`0xFFFF`). Note that by (5), K_i can never be zero.

Also note that for each value of K_i that satisfies (15), we obtain a different permutation π.

Using the sampling method given above, the attacker can obtain $\pi^r(P)$ for every P in $\{0, 1, \ldots, 2^{16} - 1\}$. Hence, he can reveal the permutation π^r.

For a given m, the attacker determines the keys K_i that satisfy (15). Assume that there are k such keys. For each such K_i^j, $1 \leq j \leq k$, the attacker finds $E(K_i^j, P)$ for all $P \in \{0, 1, \ldots, 2^{16} - 1\}$. This is in fact π_j^r that corresponds to K_i^j i.e. $\pi_j^r(x) = E(K_i, x)$, for every x. The attacker then determines the orbit structure of π_j^r. Then he starts partially revealing π_j. He performs the following two steps for each K_i^j.

1. Use lone orbits in π_j^r. If there is a lone orbit of length n_1 in π_j^r, use Lemma 2 to reveal n_1 points in π_j. From those, reveal n_1 points of σ using (16).

2. Look for a collection of the same length orbits in π_j^r. If the size q of the collection is less than the least divisor of r larger than 1, then use Lemma 3 to reveal qn_2 more points in π_j, where n_2 is the length of an orbit in the collection. Again, use (16) to reveal qn_2 points in σ.

Let $R_j \subset \{0, 1, \ldots, 2^{16} - 1\}$ be the points for which $\sigma(R_j)$ is revealed using Steps 1 and 2 above with K_i^j for $1 \leq j \leq k$. Let $R = R_1 \cup R_2 \cup \cdots \cup R_k$. Let $x \in R \backslash R_j$. Namely, the attacker knows $y = \sigma(x)$ but this point is revealed in either Step 1 or Step 2 for a key other than K_i^j. Then, π_j^r contains two same length orbits β^1 and β^2 such that $x \oplus K_i^j \in \beta^1$ and $y \in \beta^2$. These orbits were not used in the Step 2 for K_i^j above otherwise we would have $x \in R_j$. Hence, $\pi_j(x \oplus K_i^j) = y$. Then, the attacker uses Lemma 4 with β^1 and β^2 to reveal some more points on σ. This, in turn, adds points to R. The procedure is repeated until there are no points x satisfying $x \in R \backslash R_j$.

Furthermore, if the attacker uses any of the attacks explained below and somehow obtains the new knowledge of a sample point in π_j, and the point maps across two orbits of length n_3 in π_j^r, then he uses Lemma 4 to reveal $n_3 - 1$ more points in π_j.

Expansion attack

In the previous section we described an attack that partially reveals σ. We now describe an attack that works with a partially revealed permutation σ. This attack is applied together with the permutation orbit attack.

Assume that R and U are two disjoint subsets of $\{0, 1, \ldots, 2^{16} - 1\}$ such that $R \cup U = \{0, 1, \ldots, 2^{16} - 1\}$. Also assume that the attacker knows the value of $\sigma(x)$ for every $x \in R$ and he does not know the value of $\sigma(x)$ for any $x \in U$. In other words, R denotes the revealed portion of the domain of σ, and U denotes the unrevealed portion.

Assume that the attacker knows the triple (C, P, K_i) such that $C = E(K_i, P)$. Assume that $C \notin \sigma(R)$ i.e. he does not know the value which is mapped by σ to C. He now tries to carry out the calculation (7). He starts out with $v_0 = P$. He can calculate v_1 if $v_0 \oplus \mathrm{ROL}(K_i, m) \in R$. Once he knows v_1, he can calculate v_2 if $v_1 \oplus \mathrm{ROL}(K_i, 2m) \in R$. Assume that he continues in this fashion, reaches the penultimate step and calculates v_{r-1}. Obviously, $v_{r-1} \oplus \mathrm{ROL}(K_i, rm) \notin R$ because otherwise we would have $C = v_r = \sigma(v_{r-1} \oplus \mathrm{ROL}(K_i, rm)) \in \sigma(R)$ which contradicts the assumption. But this means that the attacker has just revealed the value of the map σ at a new point $v_{r-1} \oplus \mathrm{ROL}(K_i, rm)$ because he already knows C. Thus, if the attacker reaches the last step while staying in the partially revealed portion R, he expands R by one point and shrinks U by one point.

Every time the expansion attack succeeds and the attacker reveals a new point on the map σ, he uses Lemma 4 to check if this corresponds to mapping across two different same-length orbits in π^r. If so, the revealed portion R

is expanded even more. This, in turn, increases the probability that next application of the expansion attack succeeds.

Skipping attack

Using (8), we see that when $r \geq 9$, we have $m = 1$. So the attacker can use only K_i =`0xFFFF` in the permutation orbit attack. Moreover, when r is an even number, its smallest divisor is 2 and he can not use Lemma 3. This adversely affects the size of the revealed set R that can be used in the expansion attack. We now describe another attack that works with $r \geq 9$ and even. The attack relies on deriving a new permutation by skipping over odd rounds in the expression of E in (7).

Assume that a nonzero K_i satisfies

$$\mathrm{ROL}(K_i, 2) = K_i. \tag{17}$$

Using (17) with (7) and substituting odd round outputs into even round expressions, we obtain

$$\begin{aligned}
v_0 &= P, \\
v_2 &= \sigma(\sigma(v_0 \oplus \mathrm{ROL}(K_i, 1)) \oplus K_i), \\
v_4 &= \sigma(\sigma(v_2 \oplus \mathrm{ROL}(K_i, 1)) \oplus K_i), \\
&\vdots \\
v_r &= \sigma(\sigma(v_{r-2} \oplus \mathrm{ROL}(K_i, 1)) \oplus K_i), \\
C &= v_r.
\end{aligned}$$

Defining a new permutation γ as

$$\gamma(x) = \sigma(\sigma(x \oplus \mathrm{ROL}(K_i, 1)) \oplus K_i), \tag{18}$$

we can express the relation between C and P as

$$C = \gamma^{r/2}(P).$$

First, the attacker applies the permutation orbit attack with K_i =`0xFFFF`. In doing so, he obtains the permutation π^r, and using Lemma 2 and Lemma 3, he reveals a portion of σ. Let R denote the revealed portion of the map σ.

The skipping attack proceeds as follows. As in the permutation orbit attack, by choosing every $P \in \{0, 1, \ldots, 2^{16}-1\}$ and obtaining their corresponding ciphertext block with K_i^1 =`0x5555` and K_i^2 =`0xAAAA` satisfying (17), the attacker finds the permutations $\gamma_1^{r/2}$ and $\gamma_2^{r/2}$. For each $\gamma_j^{r/2}$, $j = 1, 2$, the attacker uses its orbit structure to reveal a portion of γ_j.

Assume the attacker has determined a pair (x, y) such that $y = \gamma_j(x)$ for some j. Hence, he knows that

$$y = \sigma(\sigma(x \oplus \mathrm{ROL}(K_i^j, 1)) \oplus K_i^j). \tag{19}$$

There are two ways the attacker can use (19) to reveal a new point on σ. If $x \oplus \mathrm{ROL}(K_i^j, 1) \in R$ and $y \notin \sigma(R)$, the attacker reveals the value of the map σ at $\sigma(x \oplus \mathrm{ROL}(K_i^j, 1)) \oplus K_i^j$ as y. On the other hand, if $y \in \sigma(R)$ and $x \oplus \mathrm{ROL}(K_i^j, 1) \notin R$, the attacker reveals the value of the map σ at $x \oplus \mathrm{ROL}(K_i^j, 1)$ as $\sigma^{-1}(y) \oplus K_i^j$.

Thus, with the skipping attack, the attacker reveals some new points on the map σ. He subsequently uses Lemma 4 to check if these new points correspond to mappings across two different orbits in π^r that were not used in the permutation orbit attack. If so, the revealed portion R is expanded even more.

Example 2. In the first example, we used the cryptosystem with secret parameters $r = 5$, $m = 3$, $t = 12$, C_0 =`0x4ED3`, K_s =`0x8F4C`. By the equivalence explained in above, this is equivalent to K_s =`0xC19F`=`0x4ED3`$\oplus$`0x8F4C` and C_0 =`0x0000`, We used the standard map as TDCM. The secret TDCM parameter is $K_c = 53246$.

Since $m = 3$, we can apply the permutation orbit attack only with K_i^1 =`0xFFFF`. We obtain the orbit structure of π_1^5 as $(1, 53712)$, $(1, 6432)$, $(5, 779)$, $(1, 699)$, $(1, 449)$, $(1, 252)$, $(1, 72)$, $(5, 5)$. Here a pair (q, n) means that there are q orbits of length n.

We apply Lemma 2 to lone orbits of length 53712, 6432, 699, 449, 252 and 72 in π_1^5 to reveal 61616 entries in σ. This corresponds to 94.02% of the map σ.

We saw that $1 \notin \sigma(R)$. So, we choose $C = 1$ in the expansion attack. We try $K_i = 1$ and find $P = 65082$. The expansion attack for these values indeed succeeds and we find $1 = \sigma(680)$.

Now, we go back to the result of permutation orbit attack. Searching for 680$\oplus$`0xFFFF` in the cycles of π_1^5, we see that it is mapped across two cycles of length 779. Using this sample point with Lemma 4, we reveal 779 new points in σ. Thus, the revealed set R gets bigger by 779 new points. Hence, a new expansion attack is even more likely to succeed. Repeating the attack with 9 more unrevealed ciphertext blocks with the same $K_i = 1$, we reveal the whole map σ.

3.3 Algebraic Cryptanalysis of a Chaotic Cipher Based on Chaotic Map Lattices

In the image encryption algorithm proposed in [Pisarchik et al., 2006], the plaintext is the vector $c \in \mathbf{Z}_{256}^m$ obtained by the usual row-scan of an $N \times M$ image, where m is the total number of pixels, i.e. $m = NM$. Here, $\mathbf{Z}_{256}$ denotes the set $\{0, 1, 2, \ldots, 255\}$ of integers which are represented by 8-bit pixels. The algorithm encrypts plaintext c in three steps; D/A conversion, chained chaotic iteration and A/D conversion.

1. D/A conversion: each integer pixel value c_i is mapped to one of 256 distinct real values x_i in the chaotic attractor $\Omega = (x_{\min}, x_{\max})$ for the logistic map

$$f(u) = au(1-u),$$

using

$$x_i = g_1(c_i) = x_{\min} + (x_{\max} - x_{\min})\frac{c_i}{255}, \quad 1 \le i \le m, \tag{20}$$

where $x_{\min} = (4a^2 - a^3)/16$ and $x_{\max} = a/4$.

2. Chained chaotic iteration: the real values x_i are transformed using repeated chaotic iteration as follows. We first initialize cycle 0 values as $y_i^{(0)} = x_i$, $1 \le i \le m$. The transformation for the j^{th} cycle is given as

$$\begin{aligned} y_1^{(j)} &= A(f^n(y_m^{(j-1)}) + y_1^{(j-1)}), \\ y_i^{(j)} &= A(f^n(y_{i-1}^{(j)}) + y_i^{(j-1)}), \quad i \ge 2,\ 1 \le j \le r, \end{aligned} \tag{21}$$

where the function $A : (2x_{\min}, 2x_{\max}) \rightarrow \Omega$ guarantees that the LHS of (21) falls within the attractor. The plot of A is given in Fig. 4.

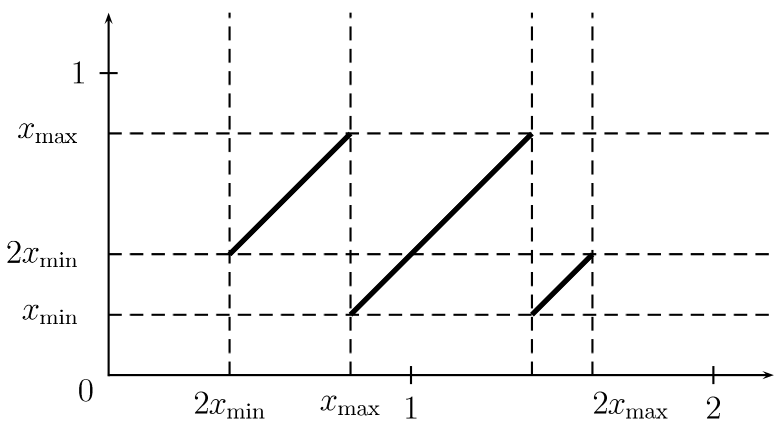

Fig. 4 The plot of the function $A : (2x_{\min}, 2x_{\max}) \rightarrow (x_{\min}, x_{\max})$. The function wraps around the attractor like modulus.

In (21), r denotes the number of cycles (rounds) in the encryption. Note that the logistic map f is iterated n times starting with the initial value $y_{i-1}^{(j)}$ for $i \ge 2$ and with $y_m^{(j-1)}$ for $i = 1$. The number of iterations n is part of the secret key.

3. A/D conversion: each $y_i^{(r)}$ is mapped back to an integer d_i in $\mathbf{Z}_{256}^m$ using

$$d_i = g_2(y_i^{(r)}) = \text{round}\left[(y_i^{(r)} - x_{\min})\frac{255}{x_{\max} - x_{\min}}\right]. \tag{22}$$

The vector $d \in \mathbf{Z}_{256}^m$ is the ciphertext.

In the subsequent discussion, we explain the attack given in [Solak and okal, 2011].

Equivalent representation

Here, we give the equivalent representation of the algorithm so that all the operations are done in $\mathbf{Z}_{256}$ and the secret quantities are some unknown permutations.

Note that g_1 and g_2 denote the D/A and A/D conversion functions in (20) and (22), respectively. For one round of encryption, we can write (21) as

$$\begin{aligned} d_i &= g_2(y_i) = g_2(A(f^n(y_{i-1}) + y_i)), \\ &= g_2(A(f^n(g_1(d_{i-1})) + g_1(c_i))), \quad 2 \leq i \leq m. \end{aligned} \tag{23}$$

Note that the mapping in (23) is from the pair $(d_{i-1}, c_i) \in \mathbf{Z}_{256} \times \mathbf{Z}_{256}$ to $d_i \in \mathbf{Z}_{256}$. Let us denote this map as $s : \mathbf{Z}_{256} \times \mathbf{Z}_{256} \to \mathbf{Z}_{256}$.

Given the secret quantities a and n, one can calculate the map s as

$$s(i,j) = g_2(A(f^n(g_1(i)) + g_1(j))), \; 0 \leq i,j \leq 255. \tag{24}$$

Now, we can write the single round encryption as

$$\begin{aligned} d_1 &= s(c_m, c_1), \\ d_i &= s(d_{i-1}, c_i), \; 2 \leq i \leq m. \end{aligned} \tag{25}$$

Similarly, we can trivially extend this expression for arbitrary number of rounds r. In this new expression of the algorithm, the equivalent secret quantities are the map s and the number of rounds r. However, the number of rounds is a small number in the range of 10. Thus, it can be safely assumed that the attacker knows r. Even when the attacker does not know r, he can try several values for r and apply the rest of the attack for the tried r. If the attack succeeds then the attacker has found the correct r.

In the next section, we give the attack that recovers the secret map s, assuming that r is known. The attack is first given in [Solak and okal, 2011].

Recovering s

Assume that the attacker chooses a two pixel image (c_1, c_2) as plaintext and obtains the corresponding ciphertext (d_1, d_2) for a single round. Using (25) with $m = 2$, we obtain

$$\begin{aligned} d_1 &= s(c_2, c_1), \quad (26) \\ d_2 &= s(d_1, c_2). \end{aligned}$$

Thus, (26) defines a function $\pi : \mathbf{Z}_{256} \times \mathbf{Z}_{256} \to \mathbf{Z}_{256} \times \mathbf{Z}_{256}$, $\pi((c_1, c_2)) = (d_1, d_2)$. Since the encryption is invertible, π is a permutation over the set $\mathbf{Z}_{256} \times \mathbf{Z}_{256}$.

Note that if attacker knows a point $(d_1, d_2) = \pi((c_1, c_2))$ on the permutation, then using (26), he can reveal the map s on two points (c_2, c_1) and (d_1, c_2).

If π is a single round encryption, then r round encryption becomes π^r. Hence, for his chosen plaintext image (c_1, c_2), the attacker observes $\pi^r((c_1, c_2))$. Choosing all of the 2^{16} possible 2-pixel plaintexts one by one and obtaining their corresponding ciphertexts, the attacker constructs the permutation π^r. Using the results given at the start of this section, the attacker reveals portions of π. Using the known points on π, the attacker finally recovers the secret map s.

We now give the details of the attack.

Permutation orbit attack

Once the attacker obtains π^r, he calculates its orbit structure using the procedure in Remark 1. Given the orbit structure of π^r, he starts by using the orbits that are shuffled going from π to π^r. The attacker uses such orbits in two distinct categories.

1. Look for lone orbits in π^r: If there is a lone orbit of length t_1 in π^r, use Lemma 2 to reveal t_1 points in π. From those, reveal at most $2t_1$ points of s using (26). Hence, if $\beta = (b_0, b_1, \ldots, b_{t_1-1})$ is a lone orbit of π^r, we can reveal some of the points on s for $0 \leq j < t_1$ as
$$s(b_{j,2}, b_{j,1}) = b_{(j+r^*) \bmod t_1, 1},$$
$$s(b_{(j+r^*) \bmod t_1, 1}, b_{j,2}) = b_{(j+r^*) \bmod t_1, 2}.$$
Note that each b_j is a pair $(b_{j,1}, b_{j,2})$, corresponding to a 2-pixel image.
2. Look for a collection of the same length orbits in π^r: If the size q of the collection is less than the least divisor of r larger than 1, then use Lemma 3 to reveal qt_2 more points in π, where t_2 is the length of an orbit in the collection. Again, use (26) to reveal at most $2qt_2$ new points in s.
Using the permutation attack, the attacker recovers a portion of the map s. If the portion is the whole, then the attack concludes successfully. If there are still unrevealed portions of s, the attacker performs the following consistency checks on the orbits not used in the permutation attack.

Consistency check

Suppose there are $q > 1$ orbits of length t_3 among the orbits of π^r and that none of these orbits were used in the permutation orbit attack. We now give consistency checks that can be applied to these orbits in order to reveal more points on the partially revealed map s.

Let β be one of those q orbits in π^r. There are two ways such a β might occur in π^r. One way is that β might have been obtained by the split of a larger orbit in π. The other possibility is that β was obtained by the shuffling of an orbit of the same length in π, see Remark 2.

We first test if latter is the case.

Assume that $\beta = (b_0, b_1, \ldots, b_{t_3-1})$ was obtained by the shuffling of an orbit of π. In this case, $\gcd(n_3, r) = 1$. Note that each b_j is a pair $(b_{j,1}, b_{j,2}) \in \mathbf{Z}_{256} \times \mathbf{Z}_{256}$. By Lemma 2, $\pi(b_j) = b_{(j+r^*) \bmod t_3}$, $0 \le j < n_3$. Thus, we conclude that, for $0 \le j < t_3$,

$$s(b_{j,2}, b_{j,1}) = b_{(j+r^*) \bmod t_3, 1},$$
$$s(b_{(j+r^*) \bmod t_3, 1}, b_{j,2}) = b_{(j+r^*) \bmod t_3, 2}.$$

Thus, on the assumption that β was obtained by shuffling, we reveal possibly $2t_3$ new points of the map s. However, if the assumption was wrong, then we expect to encounter inconsistencies. The newly revealed points might conflict with the already revealed points on s. Also, they might conflict among themselves.

To better see how two kinds of conflicting values might arise, let us assume that the attacker already knows $x, y, z \in \mathbf{Z}_{256}$ such that $s(x, y) = z$. If, for some j, $b_{j,2} = x$ and $b_{j,1} = y$ but $b_{(j+r^*) \bmod t_3, 1} \ne z$, then we have the conflict of the first kind, i.e. the newly revealed point conflicts with the already known point.

On the other hand, if we have j_1 and j_2 such that $b_{j_1,2} = b_{(j_2+r^*) \bmod t_3, 1}$ and $b_{j_1,1} = b_{j_2,2}$ but $b_{(j_1+r^*) \bmod t_3, 1} \ne b_{(j_2+r^*) \bmod t_3, 2}$, then we have newly revealed points conflicting among themselves.

The attacker can test both conflicts together. For every set of newly revealed points, he tries to add these to the map. If he fails due to a conflict with the already known portion, he concludes that β was not obtained by a simple shuffling, but instead was obtained by the split of a larger orbit in π.

By going through all the orbits not used in the permutation attack, and testing if they were obtained by simple shuffles, the attacker enlarges the revealed portion of s.

Now, the attacker is left with sets of orbits which are certainly obtained by the split of larger orbits in π. Let $\beta^{(1)}$ and $\beta^{(2)}$ be two such orbits of the same length t_4. We cannot directly use Lemma 1 because it is still possible that they were obtained by the split of different orbits in π.

In order to better see how this can happen, assume π has two orbits of length 10 and 15 and that $r = 6$. Then, by Lemma 1, in π^5, the length 10 orbit will be split into two length 5 orbits and lenth 15 orbit will be split into three length 5 orbits. Hence, in π^5, we see length 5 orbits coming from the split of different orbits.

Even if $\beta^{(1)}$ and $\beta^{(2)}$ come from the split of the same orbit in π, we may not directly use Lemma 4, because we lack a sample point mapping from one orbit to another.

Hence, the test for the second case proceeds as follows. The attacker chooses two same length orbits $\beta^{(1)}$ and $\beta^{(2)}$ in π^r. Let n_4 be the common length of these two orbits. He assumes that $\beta^{(1)}$ and $\beta^{(2)}$ come from the split of the same larger orbit in π and that there exist two integers $0 \leq i, j < t_4$ such that $b_i^{(1)} \in \beta^{(1)}$, $b_j^{(2)} \in \beta^{(2)}$ and $\pi(b_i^{(1)}) = b_j^{(2)}$. Fixing $i = 0$, he tries every j, $0 \leq j < t_4$, each time assuming that $\pi(b_0^{(1)}) = b_j^{(2)}$. If $\beta^{(1)}$ and $\beta^{(2)}$ are consecutive splits of a larger orbit in π, then there is such a j. If the attacker hits upon the correct j, he uses Lemma 4 and possibly reveals $2t_4$ new points on the map s as

$$s(b_{0,2}^{(1)}, b_{0,1}^{(1)}) = b_{j,1}^{(2)},\ 0 \leq j < t_4,$$
$$s(b_{j,1}^{(2)}, b_{0,2}^{(1)}) = b_{j,2}^{(2)},\ 0 \leq j < t_4.$$

On the other hand, if the attacker encounters an inconsistency with the already revealed portion of the map s, he discards j. If all the j's in $0 \leq j < t_4$ are discarded as such, then either $\beta^{(1)}$ and $\beta^{(2)}$ do not come from the same orbit π, or they come from the same orbit but their ordering was wrong, i.e. they are not consecutive splits.

By trying all the ordered pairs of orbits of the same length, the attacker is highly likely to eliminate the wrong assumptions with inconsistencies and reveal whole of the map s.

Complexity of the attack

Once the attacker obtains the permutation π^r, it takes only 2^{16} lookups to construct the orbit structure of π^r. The computational complexity of the rest of the attack depends on the orbit structure of the permutation π^r.

For a random permutation over the set $\{1, 2, \ldots, n\}$, the expected number of orbits is approximately $\log n$, [Lovasz, 2007, p. 227]. In our case $n = 2^{16}$, so we expect to have about 11 orbits in π. In the worst case, all orbits are split into r smaller orbits in π^r and we expect to have about $11r$ split orbits in π^r. If we were to check all pairs of orbits in π^r for consistency, we would perform about $121r^2$ consistency checks. Consistency checks can be done by a fixed number of lookups and comparisons. Let C denote the fixed cost of one consistency check for an orbit pair. For each pairing of two orbits of length L, we have to perform the consistency check for L shift amounts. The average orbit length for the original permutation π is $n/\log n$. Hence, for the average case with $n = 2^{16}$, we can take L as 5960. The split orbits in π^r will have an average length of $5960/r$.

Thus, the average complexity of the attack is $2^{16} + 121 \times rC \times 5960$ lookup or comparison operations. For a particular case of $r = 5$, $C = 20$, the attack takes about 10^8 basic operations on average.

3.4 Cryptanalysis of Fridrich's Image Cipher

Fridrich's cipher proposed in [Fridrich, 1998] is one of the earliest chaotic image encryption algorithms. The following discussion is a summary of the cryptanalysis given in [Solak et al., 2010a].

The plaintext P is an $M \times N$ grayscale image, where each pixel is represented using a byte. The image is first vectorized using the usual row-scan. Let $p \in S^n$ represent this vectorized image, where $S = \{0, 1, \ldots 255\}$ and $n = NM$. Thus, the plaintext is the vector $p = [p_1\ p_2\ \cdots p_n]$.

Each round consists of two steps. In the first step, p is shuffled using a secret permutation. Let b denote this secret permutation defined on the set $\{1, 2, \ldots, n\}$. Let us denote the shuffled vector by f. The relation between the shuffled vector f and the vectorized plaintext p can be expressed as

$$f_i = p_{b(i)},\ 1 \le i \le n. \tag{27}$$

Namely, the shuffled pixel at position i is obtained from the original pixel at position $b(i)$.

In the second step of the round, f is passed through a nonlinear function as

$$c_i = f_i + g(c_{i-1}) + h_i \bmod 256,\ 1 \le i \le n, \tag{28}$$

where $g : S \to S$ is a fixed nonlinear function and $h \in S^n$ is a fixed vector. In (28), c_0 is taken to be a fixed system parameter.

These two step are repeated for R rounds. In [Fridrich, 1998], $R = 10$ is suggested for good diffusion and confusion properties.

Combining (27) and (28), we obtain one round encryption as

$$c_i = p_{b(i)} + g(c_{i-1}) + h_i \mod 256,\ 1 \le i \le n. \tag{29}$$

The decryption for a single round is defined as follows. Let u be the inverse of b, so that

$$j = b(i) \ \Leftrightarrow\ i = u(j). \tag{30}$$

Using (30) in (29), we obtain

$$p_j = c_{u(j)} - g(c_{u(j)-1}) - h_{u(j)} \mod 256. \tag{31}$$

For $i = 1$, we have

$$c_1 = p_{b(1)} + g(c_0) + h_1 \mod 256.$$

The secret component of the algorithm is the permutation p. A set of secret keys are used in a chaotic system to generate this permutation. It is desirable that the permutation shows good diffusion properties in order to hide local correlations in an image. For example, in one of the schemes proposed in [Fridrich, 1998], the original image P is partitioned and Baker map applied to each partition to obtain the permutation. In this case, the set of keys are the boundaries where the image is partitioned. It is possible to use other schemes to generate a permutation. Our attack is general and applies to all of these cases.

A naive attack might try to reveal the keys that were used to generate the permutation b. However, anyone who knows the permutation p can decrypt the images. In our cryptanalysis, we develop methods to reveal the permutation b. Such an approach is more general as it easily covers cases where different chaotic maps are used to generate the permutation.

Inter-round dependencies in decryption

The function g in (29) forms a chain that relates consecutive ciphertext pixels. Hence, in encryption for a single round, a change in a plaintext pixel affects many ciphertext pixels. Indeed, if we change $p_{b(i)}$, by (29), c_i changes. Since we have

$$c_{i+1} = p_{b(i+1)} + g(c_i) + h_{i+1} \mod 256,$$

a change in c_i, in turn, changes c_{i+1}. Thus, for a single round, a change in $p_{b(i)}$ affects $c_i, c_{i+1}, \ldots, c_n$. As a result, a ciphertext pixel depends on many plaintext pixels.

However, the situation is quite different in decryption and there lies the weakest link in the algorithm. Using (31), we see that, for a single round, p_j is affected by only two ciphertext pixels, $c_{u(j)}$ and $c_{u(j)-1}$. Similarly, for two rounds, p_j is affected by at most four ciphertext pixels.

In order to see this more clearly, let us denote the output of the second round as $d_1 d_2 \cdots d_n$. Using (31) with c_k as the plaintext pixel that is input to second round, we obtain

$$c_k = d_{u(k)} - g(d_{u(k)-1}) - h_{u(k)} \mod 256, \ 1 \leq k \leq n. \tag{32}$$

Substituting $k = u(j)$ in (32), we find

$$c_{u(j)} = d_{u^2(j)} - g(d_{u^2(j)-1}) - h_{u^2(j)}. \tag{33}$$

Here, we denote by u^s, the s times composition of u with itself.

Similarly, for $k = u(j) - 1$, we have

$$c_{u(j)-1} = d_{u(u(j)-1)} - g(d_{u(u(j)-1)-1}) - h_{u(u(j)-1)}. \tag{34}$$

Thus, we see from (31), (33) and (34) that, for two rounds of decryption, p_j is affected only by the ciphertext pixels

$$d_{u^2(j)}, d_{u^2(j)-1}, d_{u(u(j)-1)}, d_{u(u(j)-1)-1}.$$

Obviously, depending on the particular permutation u, some of these four pixels might coincide.

Note that the plaintext pixel $p_{b(1)}$ is affected by only c_1 because c_0 is a fixed system parameter. Hence, for two rounds, $p_{b(1)}$ is affected by the ciphertext pixels

$$d_{u(1)}, d_{u(1)-1}.$$

Example 3. We illustrate the dependencies in the decryption for two rounds. Here, $n = 6$ and the permutation u is given as

$$u = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 1 & 5 & 6 & 3 \end{pmatrix}. \quad (35)$$

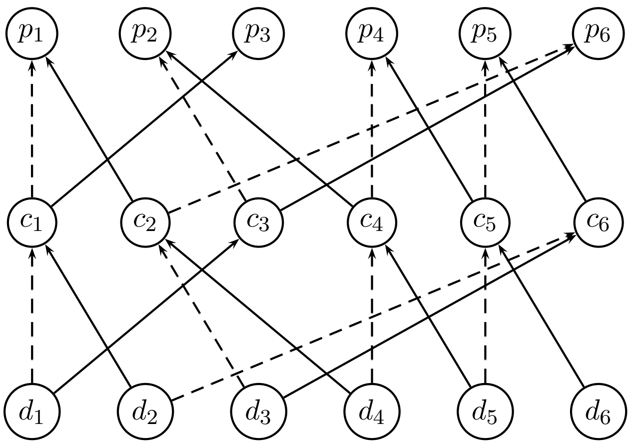

Fig. 5 The dependency paths for the permutation given in (35). A solid arrow indicates that the dependency is through u, while a dashed arrow indicates that the dependency is through $u - 1$.

The dependency paths are given in Fig. 5. In the figure, the directed arrows indicate which pixels affect the computation of the destination pixel. For example, two arrows going from c_5 and c_4 to p_4 means that p_4 is affected by c_5 and c_4.

The dependency chain from from the ciphertext d to the plaintext p is given as follows

$$\begin{aligned} p_1 &\leftarrow c_1, c_2 \leftarrow d_1, d_2, d_3, d_4, \\ p_2 &\leftarrow c_3, c_4 \leftarrow d_1, d_4, d_5, \\ p_3 &\leftarrow c_1 \leftarrow d_1, d_2, \\ p_4 &\leftarrow c_4, c_5 \leftarrow d_4, d_5, d_6, \\ p_5 &\leftarrow c_5, c_6 \leftarrow d_5, d_6, d_2, d_3, \\ p_6 &\leftarrow c_2, c_3 \leftarrow d_3, d_4, d_1. \end{aligned}$$

Note that p_3 is affected by only c_1 because $u(3) = 1$. c_1 is, in turn, affected by two ciphertext pixels d_1 and d_2. Also note that p_4 is affected by three ciphertext pixels rather than four because $u(u(4) - 1) = u^2(4) - 1 = 5$. This also means that there are two distinct dependency paths going from d_5 to p_4.

Detecting dependency using chosen ciphertext images

In general, for the decryption in an R round algorithm, a particular plaintext pixel p_j is affected by at most 2^R ciphertext pixels. For a 256×256 image encrypted in 10 rounds, we have $n = 65536$ and $2^R = 1024$. Hence, only about $\frac{1024}{65536} \approx 2\%$ of ciphertext pixels affect any given fixed plaintext pixel.

Let us denote by z, the ciphertext image after R rounds of encryption. The attacker wants to know if there is a dependency path from the ciphertext pixel z_i to the plaintext pixel p_j. Assume that the attacker knows a plaintext-ciphertext image pair (p, z). He changes the value of z_i and requests the plaintext for the changed ciphertext. If p_j changed in the new plaintext, then there is a dependency path from z_i to p_j so that z_i affects p_j.

Note that, for some changes to z_i, p_j might remain the same even when there are dependency paths from z_i to p_j. This is due the nonlinearity of encryption/decryption that operates in a finite domain. In order to detect all the dependency paths, the attacker needs to try more than one changes to z_i. It is highly unlikely that p_j remains fixed for all of these changes.

Detecting changes for all i, $1 \leq i \leq n$, the attacker constructs a binary matrix T showing the dependency relations between ciphertext and plaintext pixels in decryption. If $T_{ij} = 1$, then it means that z_i affects p_j. Since p_j is affected by at most 2^R pixels of z, each column of T contains at most 2^R 1's. All the other entries are zero.

Example 4. The matrix T for the permutation u used in Example 1 is given as

$$T = \begin{bmatrix} 1&1&1&0&0&1 \\ 1&0&1&0&1&0 \\ 1&0&0&0&1&1 \\ 1&1&0&1&0&1 \\ 0&1&0&1&1&0 \\ 0&0&0&1&1&0 \end{bmatrix}.$$

Finding $b(1)$

Writing (31) for $p_{b(1)}$, we have

$$p_{b(1)} = c_1 - g(c_0) - h_1 \bmod 256.$$

Hence, for one round, $p_{b(1)}$ is affected by only c_1, the first pixel of the output of the first round. The rest of the rounds generate at most 2^{R-1} distinct dependency paths. Therefore the column $b(1)$ of T contains at most 2^{R-1} 1's. Thus, the column of the matrix T with the least number of 1's gives the attacker a starting point for the attack. Once an attacker constructs the matrix T, he can reveal $b(1)$ by choosing the column k with the least column sum. Then he knows that $b(1) = k$ or $u(k) = 1$.

For example, by inspecting the matrix T in Example 4, the attacker can see that the third column has the least sum. Thus, he concludes that $u(3) = 1$.

Tree of dependency

In order to generalize the attack to the rest of u, we define an operation to denote the dependency relations between the sets.

Given a permutation u on the set $\{1, 2, \ldots n\}$, define the operation L on a set A as follows.

$$L(A) = \{y \mid \exists x \in A \text{ such that } y = u(x) \text{ or } y = u(x) - 1\}.$$

The set $L(A)$ has natural meaning in terms of decryption. Using (31), we see that the set $L(A)$ is the set of ciphertext pixels that affect the set A of plaintext pixels in one round of decryption. In particular, for an integer $k \in \{1, 2, \ldots, n\}$, $L(\{k\})$ is given as

$$L(\{k\}) = \begin{cases} \{u(k)\} & \text{if } u(k) = 1, \\ \{u(k), u(k) - 1\} & \text{otherwise} \end{cases} \tag{36}$$

When L operates on a set with a single element k, we drop the set notation in $L(\{k\})$ and use instead $L(k)$.

We can naturally compose L with itself to define its higher powers. Thus, for $L^2(k)$, we have

$$\begin{aligned} L^2(k) &= L(\{u(k), u(k) - 1\}) \\ &= \{u^2(k), u^2(k) - 1, u(u(k) - 1), u(u(k) - 1) - 1\}. \end{aligned}$$

Here, we implicitly assumed that $1 \notin \{u(k), u^2(k), u(u(k) - 1)\}$. If we have $u(k) = 1$ and $u^2(k) \neq 1$, then, by the definition of L, we have

$$\begin{aligned} L^2(k) &= L(u(k)) \\ &= \{u^2(k), u^2(k) - 1\}. \end{aligned}$$

Again, the powers of L has a natural interpretation in terms of multi-round encryption. For an integer k, $L^i(k)$ is the set of the indices of ciphertext pixels that affect the plaintext p_k in i round decryption. This set is also the set of row indices where the k^{th} column of T has nonzero entries.

Example 5. For the permutation given in Example 1, we have

$$L(1) = \{1,2\},$$
$$L^2(1) = \{1,2,3,4\},$$
$$L^2(\{1,6\}) = \{1,2,3,4\}.$$

Overlapping sets of leaves

Using the chosen-plaintext attack given in the beginning of this section, the attacker constructs the matrix T. This is the same as attacker knowing the sets $L^R(k)$, $\forall k \in \{1,2,\dots,n\}$. The attacker uses this knowledge to reveal the secret permutation u. First, we need the following facts. For the proofs, see [Solak et al., 2010a].

Lemma 5. *Let x, y and z be integers in $\{1,2,\dots n\}$ such that they satisfy*

$$u(x) + 1 = u(y),$$
$$u(y) + 1 = u(z).$$

Then, for every positive integer R larger than 1,

$$L^R(y) \setminus L^R(x) \subset L^R(z).$$

Lemma 6. *Let x and y be integers such that $u(x) = 1$ and $u(y) = 2$. Then, $L^R(x) \subset L^R(y)$.*

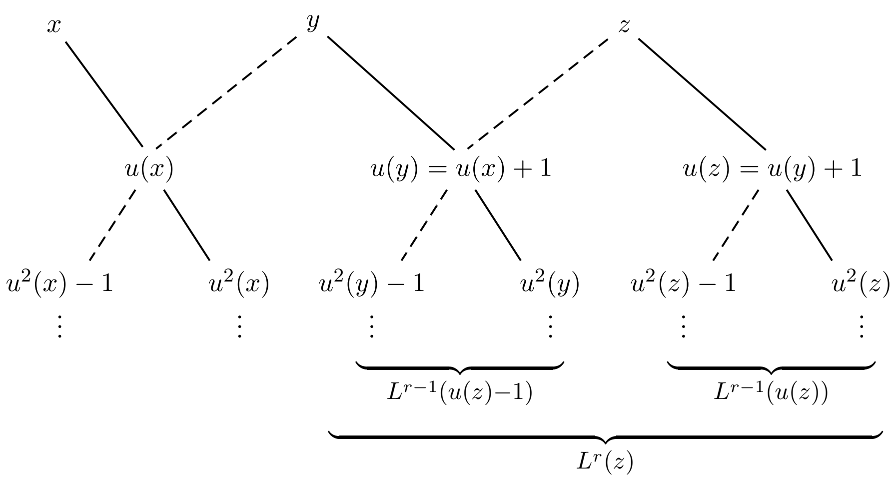

Fig. 6 The sets $L^R(a)$ and $L^{R-1}(a)$. Note that the sets are the leaves of overlapping dependency trees.

The attack

The attack starts with determining the integer x_1 that satisfy $u(x_1) = 1$. For this, the attacker chooses the set $L^R(x_1)$ that has the least number of elements. This also corresponds to choosing the column of the matrix T with the least column sum. It might happen that there are more than one candidate for x_1. For such cases, the attacker repeats the rest of the procedure for each candidate until he encounters a contradiction that he can use to eliminate the candidate.

Once the attacker knows x_1, he goes on to determine x_2 such that $u(x_2) = 2$. Define the set X_2 as

$$X_2 = \{x \mid L^R(x_1) \subset L^R(x)\}.$$

By Lemma 6, $x_2 \in X_2$. In the likely case that X_2 contains a single element, the attacker uniquely pins down x_2. If there are more then one candidate for x_2, the attacker again repeats the rest of the procedure until he can eliminate candidates.

Now, the attacker knows x_1 and x_2 such that $u(x_1) = 1$ and $u(x_2) = 2$. He then searches for x_3 such that $u(x_3) = 3$. In order to pin down x_3, the attacker finds the set defined by

$$X_3 = \{x \mid L^R(x_2) \setminus L^R(x_1) \subset L^R(x)\}.$$

By Lemma 5, $x_3 \in X_3$. If X_3 contains a single element, then the attacker has just found x_3 that satisfies $u(x_3) = 3$.

The attacker continues in this fashion and uses his knowledge of x_i and x_{i+1} to reveal x_{i+2} such that $u(x_i) = i$, $u(x_{i+1}) = i+1$ and $u(x_{i+1}) = i+2$. The attack concludes when all the entries of the secret permutation u are revealed.

In cases when X_{i+1} contains $z_1, z_2, \ldots z_v$, the attacker applies the procedure for each z_m, $1 \leq m \leq v$, each time assuming that $u(z_m) = i+1$.

For false candidates, we expect the iteration to yield an empty set at some point. Namely, if the set $L^R(z_m) \setminus L^R(x_{i+1})$ is not contained in any $L^R(w)$, then $u(z_m) \neq i+1$ and we eliminate the candidate z_m.

The iterations of the attack are expressed as a recursion in Algorithm 1. The recursive function is `FindNext()` which takes no arguments. The constant data of the algorithm are the sets $L^R(k)$, $\forall k \in \{1, 2, \ldots, n\}$. The algorithm manipulates the global variables b and i. The variable i shows the portion of b that is assumed to have been revealed. Namely, the function `FindNext()` assumes that $b(1), b(2), \ldots, b(i)$ have already been revealed. Note that we also assume that the values $b(1)$ and $b(2)$ are initially known.

In Algorithm 1, Line 1, we find the candidates for $b(i+1)$. In doing this, we exclude the set $\{b(1), b(2), \ldots, b(i)\}$ which is assumed to have been revealed so far. For each candidate z, Lines 6-10 recursively apply the algorithm assuming that $u(z) = i + 1$. The function `FindNext()` returns in Line 13 when no candidates are found. It means that the recursion can not go any deeper because a wrong assumption about the permutation value has been made. In this case, Line 11 backtracks once and another candidate is tried.

Algorithm 1. `FindNext()`

Data: $L^R(k),\ \forall k \in \{1, 2, \ldots, n\},\ b(1),\ b(2)$.
Result: b
Global Variable: b and i. Initially $i \leftarrow 2$.

1 `FindNext()`;
2 **begin**
3     $Z \leftarrow \left\{x \mid L^R(b(i)) \setminus L^R(b(i-1)) \subset L^R(x)\right\} \setminus \{b(1), b(2), \ldots, b(i)\}$;
4     $i \leftarrow i + 1$;
5   **if** $Z \neq \emptyset$ **then**
6   **foreach** $z \in Z$ **do**
7             $b(i) \leftarrow z$;
8   **if** $i = n$ **then**
9   exit
10   ;
11   `FindNext()`;
12         $i \leftarrow i - 1$
13   **else**
14   return
15 **end**

Example 6. We illustrate the attack with an artificially small image size. We choose $R = 3$ and assume an image size of 4×4. Therefore, the secret permutation u maps within the set $\{1, 2, \ldots 16\}$. We generated the permutation randomly and it is given as

$$u = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 9 & 8 & 6 & 12 & 1 & 11 & 14 & 15 & 7 & 3 & 10 & 2 & 16 & 5 & 4 & 13 \end{pmatrix}.$$

The other fixed functions g and h are chosen randomly. The attacker calculates the matrix T as

$$T = \begin{pmatrix}
0&0&0&0&0&0&1&1&0&1&0&0&1&1&1&0\\
0&0&1&1&0&1&0&0&1&0&0&0&1&1&1&0\\
1&1&1&1&0&1&1&0&1&0&0&0&0&0&0&1\\
1&1&0&0&0&0&1&0&1&0&0&0&0&0&0&1\\
1&1&0&0&0&1&0&1&1&0&1&0&1&0&0&0\\
0&0&0&1&1&1&0&1&0&0&1&1&1&0&0&0\\
0&0&0&1&1&1&0&0&0&0&1&1&0&0&0&1\\
0&0&1&1&0&1&0&0&0&0&1&0&0&1&0&1\\
0&0&1&1&0&0&0&0&1&0&0&0&0&1&1&1\\
1&0&1&0&0&0&0&0&1&1&1&0&0&1&1&0\\
1&0&0&0&0&0&1&1&0&1&1&0&1&0&1&0\\
0&0&0&0&0&0&1&1&0&0&0&0&1&0&0&1\\
1&0&0&0&0&0&1&0&0&1&1&1&0&0&0&1\\
1&0&0&0&1&0&0&0&0&1&1&1&0&0&0&0\\
0&1&0&0&1&0&0&0&1&1&0&1&1&0&0&0\\
0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&0
\end{pmatrix}.$$

For the i^{th} column of the matrix T, the row indices of the 1's give the set $L^3(i)$.

First, the attacker reveals $b(1)$. For this, he finds the minimum sum column which is the column 5. Thus, the attacker reveals that $u(5) = 1$, or equivalently that $b(1) = 5$. From the 5^{th} column, the attacker sees that $L^3(5) = \{6, 7, 14, 15\}$. He then uses Lemma 6 and searches for the column that has 1's in its 6^{th}, 7^{th}, 14^{th} and 15^{th} rows. This column turns out to be the 12^{th} one. Hence, he concludes $b(2) = 12$. Now that the attacker knows the values of $b(1)$ and $b(2)$. Next, he applies Algorithm 1. Using the matrix T, he calculates that $L^3(12) \setminus L^3(5) = \{13\}$. Searching through the columns of T, the attacker finds that columns 1, 7, 10, 11, 16 have 1 in their 13^{th} rows. Thus, $Z = \{1, 7, 10, 11, 16\}$. Now, he tries those as candidates for $b(3)$. First, he assumes $b(3) = 1$. On this assumption, he calculates the set $L^3(1) \setminus L^3(12) = \{3, 4, 5, 10, 11\}$. But, there is no column that has 1's in its rows corresponding to this set. Hence, $b(3) \neq 1$. Next, he tries $b(3) = 7$. He calculates $L^3(7) \setminus L^3(12) = \{1, 3, 4, 11, 12\}$. Again, there is no column that has 1's in its rows corresponding to this set. The third candidate is 10, which happens to be the correct one. Assuming $b(3) = 10$, the attacker quickly reveals the rest of the secret permutation b.

4 Conclusion

The rich and complex behavior of chaotic systems attracted many researchers into designing chaotic ciphers using the inherent noise-like character of chaotic signals. During the last two decades, we have seen many proposals for chaotic ciphers. At the same time, many of these proposals have been shown to be very weak or in some cases even basically flawed.

We see that in many chaotic ciphers, it is possible to bypass the chaotic subsystems and attack the intermediate parameters instead. In such a case, it does not matter how rich and complex the chaotic behavior are. An attacker always tries to exploit the weakest link in the encryption chain.

In yet many other cases, algebraic structure of the chaotic cipher contains weaknesses that can be exploited by an attacker. Interestingly, in only rare cases, the chaos is the weak point. Rather, the break comes through the way that the chaotic signals are used in encryption.

A healthy co-development of analysis and design is crucial for the chaos cryptography to become a mature field. The designers should be well aware of the existing attacks and use strong and well-known structures in their designs. Also, chaos cryptography needs to incorporate rigorous tools and methods developed in mainstream cryptography.

References

Alvarez, G., Li, S.J.: Some basic cryptographic requirements for chaos-based cryptosystems. Int. J. Bifurcation Chaos 16(8), 2129–2151 (2006)

Amig, J.M., Kocarev, L., Szczepanski, J.: Theory and practice of chaotic cryptography. Physics Letters A 366(3), 211–216 (2007), ISSN 0375-9601

Anstett, F., Millerioux, G., Bloch, G.: Chaotic cryptosystems: Cryptanalysis and identifiability. IEEE Tran. Circuits and Systems I-Regular Papers 53(12), 2673–2680 (2006), ISSN 1057-7122

Çokal, C., Solak, E.: Cryptanalysis of a chaos-based image encryption algorithm. Physics Letters A 373(15), 1357–1360 (2009) ISSN 0375-9601

Dachselt, F., Schwarz, W.: Chaos and cryptography. IEEE Tran. Circuits and Systems I - Fundamental Theory and Applications 48(12), 1498–1509 (2001), ISSN 1057-7122

Fraleigh, J.B.: A First Course in Abstract Algebra. Addison Wesley, Reading (2002)

Fridrich, J.: Symmetric ciphers based on two-dimensional chaotic maps. International Journal of Bifurcation and Chaos 8(6), 1259–1284 (1998)

Guan, Z.-H., Huang, F., Guan, W.: Chaos-based image encryption algorithm. Physics Letters A 346(1-3), 153–157 (2005), ISSN 0375-9601, doi:10.1016/j.physleta.2005.08.006

Huang, C.K., Nien, H.H.: Multi chaotic systems based pixel shuffle for image encryption. Optics Communications 282(11), 2123–2127 (2009), ISSN 0030-4018, doi:10.1016/j.optcom.2009.02.044

Kocarev, L., Jakimoski, G.: Pseudorandom bits generated by chaotic maps. IEEE Tran. Circuits and Systems I - Fundamental Theory and Applications 50(1), 123–126 (2003), ISSN 1057-7122

Li, S., Li, C., Chen, G., Bourbakis, N.G., Lo, K.-T.: A general quantitative cryptanalysis of permutation-only multimedia ciphers against plaintext attacks. Signal Processing: Image Communication 23(3), 212–223 (2008), ISSN 0923-5965, doi:10.1016/j.image.2008.01.003

Lovasz, L.: Combinatorial Problems and Exercises. AMS, Providence (2007)

Masuda, N., Jakimoski, G., Aihara, K., Kocarev, L.: Chaotic block ciphers: From theory to practical algorithms. IEEE Tran. Circuits and Systems I-Regular Papers 53(6), 1341–1352 (2006), ISSN 1057-7122

Patidar, V., Pareek, N.K., Sud, K.K.: A new substitution-diffusion based image cipher using chaotic standard and logistic maps. Communications in Nonlinear Science and Numerical Simulation 14(7), 3056–3075 (2009), ISSN 1007-5704, doi:10.1016/j.cnsns.2008.11.005

Pisarchik, A.N., Flores-Carmona, N.J., Carpio-Valadez, M.: Encryption and decryption of images with chaotic map lattices. Chaos 16(3), 033118 (2006)

Rhouma, R., Solak, E., Belghith, S.: Cryptanalysis of a new substitution-diffusion based image cipher. Communications in Nonlinear Science and Numerical Simulation 15(7), 1887 (2010)

Solak, E., Cokal, C., Yildiz, O.T., Biyikoglu, T.: Cryptanalysis of fridrich's chaotic image encryption. Int. J. Bifurcation Chaos 20(5), 1405–1413 (2010a)

Solak, E., Çokal, C.: Cryptanalysis of a cryptosystem based on discretized two-dimensional chaotic maps. Physics Letters A 372(46), 6922–6924 (2008)

Solak, E., Çokal, C.: Algebraic break of a cryptosystem based on discretized two-dimensional chaotic maps. Physics Letters A 373(15), 1352–1356 (2009)

Solak, E., Çokal, C.: Algebraic break of image ciphers based on discretized chaotic map lattices. Information Sciences 181(1), 227–233 (2011)

Solak, E., Rhouma, R., Belghith, S.: Cryptanalysis of a multi-chaotic systems based image cryptosystem. Optics Communications 283(2), 232–236 (2010b)

Xiang, T., Wong, K.-W., Liao, X.: A novel symmetrical cryptosystem based on discretized two-dimensional chaotic map. Physics Letters A 364(3-4), 252–258 (2007)