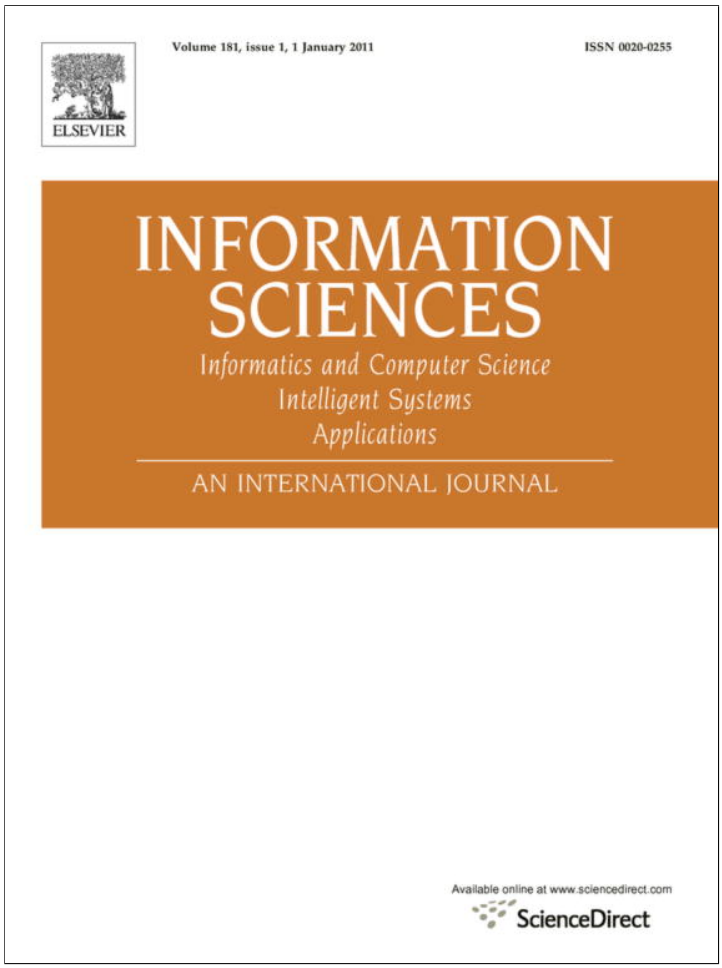

Information Sciences 181 (2011) 227–233

Contents lists available at ScienceDirect

Information Sciences

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

Algebraic break of image ciphers based on discretized chaotic map lattices

Ercan Solak *, Cahit Çokal

Department of Computer Science and Engineering, Isik University, 34980 Istanbul, Turkey

ARTICLE INFO

Article history:
Received 12 June 2009
Received in revised form 18 August 2010
Accepted 3 September 2010

Keywords:
Communication using chaos
Algebraic structures and number theory
Nonlinear dynamics and chaos
Algorithms
Image processing

ABSTRACT

In this paper, we provide an algebraic cryptanalysis of a recently proposed chaotic image cipher. We show that the secret parameters of the algorithm can be revealed using chosen-plaintext attacks. Our attack uses the orbit properties of the permutation maps to deduce encryption values for a single round. Once a single round encryption is revealed, the secret parameters are obtained using simple assignments.

1. Introduction

During the last decade there has been a steady increase in the research effort towards designing cryptographic algorithms and protocols which employ some form of a chaotic system at their core. Among the earliest attempts at chaotic encryption was the use of the iterated logistic map as the encryption transformation and its parameter as the secret key [8]. This algorithm was broken using a chosen ciphertext attack [5]. Still, one dimensional discrete maps in general and the logistic map in particular are used in many chaotic encryption proposals because the simple expression of such maps make them suitable for implementation in both hardware and software [1,4,9,12,14,17,18].

A particular area of interest within the chaos cryptography is the image encryption. Naturally, a fast and strong image encryption has the potential for application in diverse areas of multimedia applications. By treating the image as a sequence of bits, classical block ciphers like DES and AES can be used with an appropriate mode of operation. Still, the desire to obtain faster ciphers motivates researchers to seek new ways to incorporate chaos in image encryption [2,6,11,12,16].

Recently, a new chaotic image encryption algorithm based on chaotic map lattices (CML) was proposed in [12]. Subsequently, some of the weaknesses and implementation problems of the algorithm was analyzed in [2]. One of the suggestions for improvements offered in [2] was that digitized computations must be used within the phase-space. In this paper, we show that whatever be the internal precision, the algorithm is vulnerable to attacks devised in [15]. In particular, we show that an attacker can choose a moderate number of custom plaintext images and use these to compute the secret parameters in an equivalent expression of the algorithm.

For the set of 2-pixel images, one round encryption is equivalent to a permutation over a set of size 2^{16}. The encryption over r rounds becomes equivalent to rth power of this permutation. An attacker chooses all possible 2-pixel images as plaintexts and obtains their corresponding ciphertext images. Thus, the attacker obtains the rth power of the permutation. Using the results of [15], the attacker reconstructs the permutation and then the secret map.

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

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.09.009

The organization of the paper is as follows. In the next section, we describe the original encryption algorithm. In Section 3, we give the equivalent representation. In Section 4, we use permutation orbit attack and consistency check to reveal secret parameters of the equivalent representation. We analyze the complexity of the attack in Section 5. We provide examples and simulation results in Section 6. The paper finishes with some concluding remarks.

2. Description of the system

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. $m = NM$. $\mathbf{Z}_{256}$ 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

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

   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 $y_i^{(0)} = x_i$, $1 \leqslant i \leqslant m$. The transformation for the jth cycle is given as

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

   where the function $A : (2x_{\min}, 2x_{\max}) \to \Omega$ guarantees that the LHS of (2) falls within the attractor. The plot of A is given in Fig. 1.
   In (2), 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 $i \geqslant 2$ 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

   $$d_i = g_2(y_i^{(r)}) = \text{round}\left[(y_i^{(r)} - x_{\min})\frac{255}{x_{\max} - x_{\min}}\right]. \tag{3}$$

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

In the original description of the systems given in [12], it might be interpreted that (3) is applied only after the last round. However, such an implementation would make the whole encryption non-invertible as highlighted in [2]. Moreover, we compared our description of the algorithms given in (1)–(3) against the actual implementation code of the authors of [12]. Indeed, our description agrees with their implementation. Therefore, in the subsequent analysis, we assume that (3) is applied in every round. For further discussion related to implementation issues of a similar algorithm see [3,13].

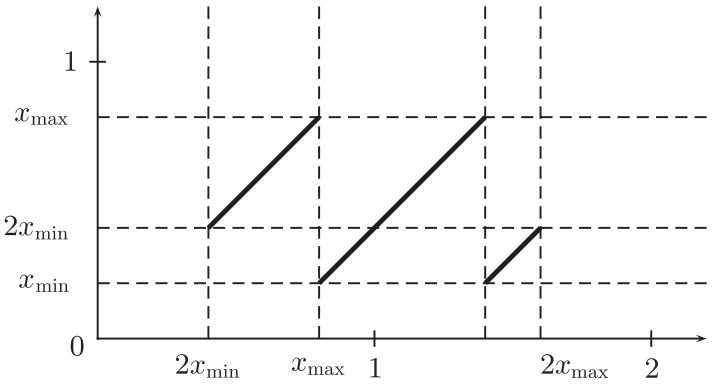

Fig. 1. The plot of the function $A : (2x_{\min}, 2x_{\max}) \to (x_{\min}, x_{\max})$. The function wraps around the attractor like modulus.

3. 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 (1) and (3), respectively. For one round of encryption, we can write (2) as

$$d_i = g_2(y_i) = g_2(A(f^n(y_{i-1}) + y_i)) = g_2(A(f^n(g_1(d_{i-1})) + g_1(c_i))), \quad 2 \leqslant i \leqslant m. \tag{4}$$

Note that the mapping in (4) 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

$$s(i,j) = g_2(A(f^n(g_1(i)) + g_1(j))), \quad 0 \leqslant i,j \leqslant 255. \tag{5}$$

Now, we can write the single round encryption as

$$\begin{aligned} d_1 &= s(c_m, c_1), \\ d_i &= s(d_{i-1}, c_i), \quad 2 \leqslant i \leqslant m. \end{aligned} \tag{6}$$

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 an attack that recovers the secret map s, assuming that r is known.

4. Recovering s

The map s has 256^2 unknown entries each of which can take 256 different values. Hence, trying to reveal s using brute force requires $256^{256^2} = 2^{2^{19}}$ trials. Clearly, this is infeasible.

However, using a chosen-plaintext attack, the unknown map s can be recovered within a few hours. First, we need a few results related to the powers of permutations. For the proofs of lemmas see [15].

Definition 1 [7]. 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 $\pi(a_0) = a_1$, $\pi(a_1) = a_2$, …, $\pi(a_{n-2}) = a_{n-1}$, $\pi(a_{n-1}) = a_0$. n is the length of the ordered orbit.

Theorem 2 [7]. *A permutation defined on a finite set partitions the set into disjoint ordered orbits.*

From now on, we refer to an ordered orbit simply as an orbit.

Remark 3. 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.

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

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

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

Remark 6. An immediate result of Lemmas 4 and 5 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 7. *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,*

$$\pi(b_j) = b_{(j+r^*) \bmod t}, \quad 0 \leqslant j < t, \tag{7}$$

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

Lemma 8. *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*

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

We now give the attack based on the structure of the orbits of permutations.

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 (6) with $m = 2$, we obtain

$$\begin{aligned} d_1 &= s(c_2, c_1), \\ d_2 &= s(d_1, c_2). \end{aligned} \tag{8}$$

Thus, (8) 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 (8), 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.

4.1. Permutation orbit attack

Once the attacker obtains π^r, he calculates its orbit structure using the procedure in Remark 3. 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 5 to reveal t_1 points in π. From those, reveal at most $2t_1$ points of s using (8). 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 $0 \leqslant j < t_1$ 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 7 to reveal qt_2 more points in π, where t_2 is the length of an orbit in the collection. Again, use (8) 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.

4.2. 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 6.

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 5, $\pi(b_j) = b_{(j+r^*) \bmod t_3}$, $0 \leqslant j < n_3$. Thus, we conclude that, for $0 \leqslant j < t_3$,

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

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 $b_{(j+r^*) \bmod t_3,1} \neq z$, 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 $b_{(j_1+r^*) \bmod t_3,1} \neq b_{(j_2+r^*) \bmod t_3,2}$, 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 4 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 4, in π^5, the length 10 orbit will be split into two length 5 orbits and length 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 8, 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 $\pi$and that there exist two integers $0 \leqslant i,j < t_4$ 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, $0 \leqslant j < t_4$, 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 8 and possibly reveals $2t_4$ new points on the map s as

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

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 $0 \leqslant j < t_4$ 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.

5. 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 $\log n$, [10, 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.

6. Simulations

We first illustrate the calculations involved in the attacks using a small map s.

Assume that there are only 4 pixels values $\{0,1,2,3\}$, so we will be using mod 4 instead of mod 256. For the purpose of illustration, let us choose the secret map s as in Table 1. Let us choose the number of rounds $r = 2$.

The attacker first chooses all 16 of the 2-pixel images (c_1,c_2) for $0 \leqslant c_1 \leqslant 3$, $0 \leqslant c_2 \leqslant 3$ and requests corresponding ciphertext images $(z_1,z_2) = \pi^2(c_1,c_2)$. These are given in Table 2.

In Table 1, the row and column numbers start from 0 and correspond to c_1 and c_2, respectively. Thus, $(1,2)$ is encrypted to $(0,3)$ in 2-round encryption and so on. Upon the orbit decomposition of the permutation π^2, the attacker sees that it has one orbit of length 9, one orbit of length 1 and two orbits of length 3 each. The orbits are given as

Table 1
The secret map $s(i,j)$, $s : \mathbf{Z}_4 \times \mathbf{Z}_4 \to \mathbf{Z}_4$.

i	j			
	0	1	2	3
0	2	3	1	0
1	3	1	2	0
2	1	3	0	2
3	0	3	1	2

Table 2
The permutation π^2 which corresponds to 2-round encryption of the pairs $(c_1,c_2) \in \mathbf{Z}_4 \times \mathbf{Z}_4$.

c_1	c_2			
	0	1	2	3
0	(2,3)	(2,2)	(3,1)	(2,1)
1	(0,2)	(1,1)	(0,3)	(2,0)
2	(3,2)	(1,0)	(3,3)	(3,0)
3	(1,2)	(0,0)	(1,3)	(0,1)

$$o_1 = (0,0),(2,3),(3,0),(1,2),(0,3),(2,1),(1,0),(0,2),(3,1),$$
$$o_2 = (0,1),(2,2),(3,3),$$
$$o_3 = (1,3),(2,0),(3,2),$$
$$o_4 = (1,1).$$

For the first lone orbit o_1, the attacker uses Lemma 5. Note that the inverse of 2 in mod9 is 5. Hence, by Lemma 5, $\pi((0,0)) = (2,1)$, $\pi((2,3)) = (1,0)$, and so on. Using (8), the attacker reveals that $s(0,0) = 2$, $s(2,0) = 1$, $s(3,2) = 1$, $s(1,3) = 0$, and so on. In total, the attacker manages to reveal nine points of the secret map s using the lone orbit o_1.

The other lone orbit o_4 helps to reveal that $s(1,1) = 1$.

There are two remaining orbits, o_2 and o_3. Since $3 \not< r$, the attacker cannot use Lemma 7 directly.

The attacker tries if any one of these orbits were obtained by the shuffling of an orbit in π. First, he tries o_2. Note that the inverse of 2 in mod3 is *2*. Assuming that o_2 were obtained through shuffling, the attacker obtains $\pi((0,1)) = (3,3)$, $\pi((2,2)) = (0,1)$ and $\pi((3,3)) = (2,3)$. Using these points on the permutation π with (8), the attacker reveals a consistent set of new points on s as $s(1,0) = 3$, $s(3,1) = 3$, $s(2,2) = 0$, $s(0,2) = 1$, $s(3,3) = 2$, $s(2,3) = 3$. Thus, all of the secret map is revealed at this stage of the attack, and the attack concludes successfully.

In order to simulate our proposed attack on a realistic image representation with 256 pixel levels, we chose a random secret map s and $r = 6$. The map s is displayed as an image in Fig. 2a.

When we find the orbit structure of π^6, we find that there are three lone orbits of lengths 58,501, 1597 and 109. Also, there are three orbits of length 877, 12 orbits of length 198, four orbits of length 79 and six orbits of length 1. Starting with the longest lone orbit and applying Lemma 5, the attacker reveals 58,501 points of the secret map s. The other two lone orbits

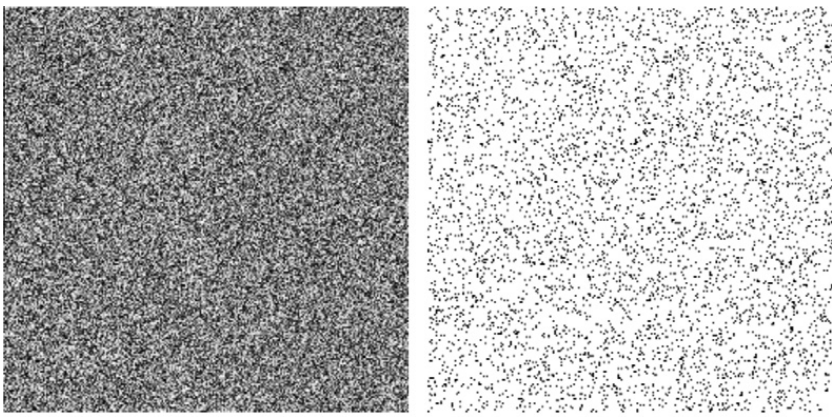

Fig. 2. (a) The secret map s. (b) The revealed (white) and unrevealed (black) portions of the map after the lone orbits are used.

reveal 1597 and 109 new points, respectively. Note that after using the lone orbits, the attacker reveals about 92% of the secret map s. In Fig. 2a, the revealed points are shown in white and unrevealed points are shown in black. The three orbits of length 877 turn out be split from a larger orbit and they reveal a total of 877 new points. Twelve orbits of length 198 also turn out to be split orbits. They reveal a total of 2376 new points on the secret map s. Four orbits of length 79 reveal 316 points. Finally, six singleton orbits reveal one new point each. Thus we reveal the full secret map.

The attack takes about 40 min under MATLAB running on Mac OS X 10.6.2 with Intel Core 2 Duo 2.16 GHz processor and 3.3 GB RAM.

7. Conclusions

In an encryption algorithm, it is essential that the algorithm provides an equal degree of security for the whole range of inputs. Otherwise, an attacker can launch a chosen-plaintext attack using the set of plaintexts for which the algorithm leaks information about secret parameters.

In this paper we demonstrated a chosen-plaintext attack on an equivalent representation of a CML-based image encryption algorithm. Our attack uses orbit structures of permutation functions defined by the encryption of 2-pixel images. We showed that an attacker can use these orbit structures to deduce the equivalent secret parameters of the algorithm.

The computational complexity of the attack is moderate so that the attack takes less than 1 h with a non-optimized software implementation.

The main weakness of the encryption scheme is the use of small blocks of 8 bits in calculating the initial conditions to the chaotic map. In general, using n bits in the scheme yields a permutation map of 2^{2n} elements. Thus, in order to prevent our proposed attack, one possible modification would be to combine a few adjacent pixels in calculating the initial condition. This would increase the size of the permutation. Of course, the whole scheme needs to be analyzed again for any new weaknesses brought about by any such modifications.

Acknowledgements

This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under Project No. 106E143.

The authors thank anonymous reviewers for their suggestions to improve the content and the presentation of the paper in several places.

References

[1] G. Alvarez, S.J. Li, Cryptanalyzing a nonlinear chaotic algorithm (NCA) for image encryption, Communications in Nonlinear Science and Numerical Simulation 14 (11) (2009) 3743–3749.
[2] D. Arroyo, R. Rhouma, G. Alvarez, S.J. Li, V. Fernandez, On the security of a new image encryption scheme based on chaotic map lattices, Chaos 18 (3) (2008) 033112.
[3] D. Arroyo, S.J. Li, J.M. Amigo, G. Alvarez, R. Rhouma, Comment on Image encryption with chaotically coupled chaotic maps, Physica D: Nonlinear Phenomena 239 (12) (2010) 1002–1006.
[4] M.S. Baptista, Cryptography with chaos, Physics Letters A 240 (1–2) (1998) 50–54.
[5] E. Biham, Cryptoanalysis of the chaotic-map cryptosystem suggested at Eurocrypt'91, in: EUROCRYPT'91, 10th Annual International Conference on the Theory and Applications of Cryptographic Techniques, 1991, pp. 532–534.
[6] G.R. Chen, Y.B. Mao, C.K. Chui, A symmetric image encryption scheme based on 3D chaotic cat maps, Chaos Solitons and Fractals 21 (3) (2004) 749–761.
[7] J.B. Fraleigh, A First Course in Abstract Algebra, Addison Wesley, 2002.
[8] T. Habutsu, Y. Nishio, I. Sasase, S. Mori, A secret key cryptosystem by iterating a chaotic map, in: EUROCRYPT'91, 10th Annual International Conference on the Theory and Applications of Cryptographic Techniques, 1991, pp. 127–140.
[9] L. Kocarev, G. Jakimoski, Logistic map as a block encryption algorithm, Physics Letters A 289 (4–5) (2001) 199–206.
[10] L. Lovasz, Combinatorial Problems and Exercises, AMS, 2007.
[11] Y.B. Mao, G.R. Chen, S.G. Lian, A novel fast image encryption scheme based on 3D chaotic baker maps, International Journal of Bifurcation Chaos 14 (10) (2004) 3613–3624.
[12] A.N. Pisarchik, N.J. Flores-Carmona, M. Carpio-Valadez, Encryption and decryption of images with chaotic map lattices, Chaos 16 (3) (2006) 033118.
[13] A.N. Pisarchik, M. Zanin, Reply to: "Comment on: Image encryption with chaotically coupled chaotic maps" [Physica D 2010], Physica D: Nonlinear Phenomena 239 (12) (2010) 1001.
[14] N. Singh, A. Sinh, Chaos-based secure communication system using logistic map, Optics and Lasers in Engineering 48 (3) (2010) 398–404.
[15] E. Solak, C. Cokal, Algebraic break of a cryptosystem based on discretized two-dimensional chaotic maps, Physics Letters A 373 (15) (2009) 1352–1356.
[16] X.J. Tong, M.G. Cui, True random number generator based on mouse movement and chaotic hash function, Signal Processing 89 (4) (2009) 480–491.
[17] Y. Wang, X.F. Liao, D. Xiao, K.W. Wong, One-way hash function construction based on 2D coupled map lattices, Information Sciences 178 (5) (2008) 1391–1406.
[18] Q. Zhou, X.F. LiaoX, K.W. Wong, Y. Hu, D. Xiao, Image encryption scheme based on 3D baker with dynamical compound chaotic sequence cipher generator, Information Sciences 179 (19) (2009) 3442–3450.