Cryptanalysis of Fridrich's chaotic image encryption

Ercan Solak, Cahit Çokal, Olcay Taner Yıldız
Department of Computer Science and Engineering
Isik University, Sile, Istanbul, Turkey

Türker Bıyıkoğlu
Department of Mathematics
Isik University, Sile, Istanbul, Turkey

November 23, 2009

Abstract

We cryptanalyze Fridrich's chaotic image encryption algorithm. We show that the algebraic weaknesses of the algorithm makes it vulnerable against chosen-ciphertext attacks. We propose an attack that reveals the secret permutation that is used to shuffle the pixels of a round input. We demonstrate the effectiveness of our attack with examples and simulation results. We also show that our proposed attack can be generalized to other well-known chaotic image encryption algorithms.

Short Title: Cryptanalysis of Fridrich's cipher

1 Introduction

In the last three decades, there has been a growing research effort to apply chaos to cryptography, [Amigo et al., 2007]. The noise-like statistical properties of chaotic signals and the sensitivity of chaotic systems naturally relate them to confusion and diffusion properties of encryption algorithms, [Menezes et al., 1996; Dachselt and Schwarz, 2001].

Despite the apparent affinity between the concepts of chaos and cryptography, using chaos within an encryption algorithm is far from trivial. In fact, many chaotic ciphers fall short of satisfying even the most basic security requirements, [Alvarez and Li, 2006; Kelber and Schwarz, 2007; Masuda et al., 2006].

Early attempts at using chaos in cryptography included the synchronization based methods where synchronized chaotic signals are used to modulate and demodulate message signals, [Yang et al., 1997]. While such an approach provides a spread spectrum communication setup, the analog nature of the systems makes approximation based attacks possible, [Parker and Short, 2001; Yang et al., 1998]. Identification methods can also be used to reveal secret system parameters, [Liu et al., 2004].

Another approach is the discretization of chaotic signals at some stage in the system and using the resulting sequence to modify plaintext, possibly in multiple rounds [Masuda and Aihara, 2002a; Ozoguz et al., 2006]. Some of these schemes use chaotic systems to generate a pseudo-random sequence which is then simply XORed with the plaintext. Other proposals use iterated low-dimensional chaotic systems to implement complex nonlinear functions that are similar to S-boxes used in classical block ciphers, [Szczepanski et al., 2005]. Still others use chaos to implement permutations of plaintext blocks, [Fridrich, 1998; Masuda and Aihara, 2002b; Xiang et al., 2006].

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 motivate researchers to seek new ways to incorporate chaos in image encryption, [Pisarchik et al., 2006; Arroyo et al., 2008; Mao et al., 2004; Chen et al., 2004].

The rich variety of chaotic ciphers provides a motivation for cryptanalysts to find statistical or structural weaknesses in these proposals, [Li et al., 2008]. As a result of this dual effort to design and break chaotic ciphers, one might expect to see the emergence of robust applications of chaos in strong cryptography. While many chaotic ciphers have been shown to have weaknesses, new modifications are being proposed to implement defenses to resist the discovered attacks.

In this paper, we cryptanalize Fridrich's cipher which is one of the earliest

chaotic image encryption algorithms, [Fridrich, 1998]. We show that the algorithm can be broken using a chosen-ciphertext attack. We show that the attack reveals secret permutation of the algorithm. Although more than a decade has passed since its publication in 1998, to our knowledge, the present work is the first successful cryptanalysis of Fridrich's cipher.

The organization of the paper is as follows. Section 2 gives a mathematical description of Fridrich's image encryption algorithm. Section 3 gives a detailed description of our attack. Section 4 presents an illustrative example and simulation results for realistic image sizes. The paper finishes with concluding remarks. In conclusion section, we also discuss application of our results to other similar encryption algorithms and possible defense mechanisms.

2 Desciption of the Encryption Algorithm

The plaintext $\mathbf{P}$ 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 $\mathbf{p} \in S^n$ represent this vectorized image, where $S = \{0, 1, \dots 255\}$ and $n = NM$. Thus, the plaintext is the vector $\mathbf{p} = [p_1\ p_2\ \cdots p_n]$.

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

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

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

In the second step of the round, $\mathbf{f}$ is passed through a nonlinear function as

$$c_i = f_i + g(c_{i-1}) + h_i \bmod 256,\ 1 \leq i \leq n, \tag{2}$$

where $g : S \to S$ is a fixed nonlinear function and $\mathbf{h} \in S^n$ is a fixed vector. In Eq. (2), c_0 is taken to be a fixed system parameter.

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

Combining Eq. (1) and Eq. (2), we obtain one round encryption as

$$c_i = p_{b(i)} + g(c_{i-1}) + h_i \mod 256,\ 1 \leq i \leq n. \tag{3}$$

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

$$j = b(i) \ \Leftrightarrow\ i = u(j). \tag{4}$$

Substituting Eq. (4) in Eq. (3), we obtain

$$p_j = c_{u(j)} - g(c_{u(j)-1}) - h_{u(j)} \mod 256. \tag{5}$$

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 b. A set of secret keys are used generate this permutation. For example, in one of the schemes proposed in [Fridrich, 1998], the original image $\mathbf{P}$ 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.

3 Chosen Ciphertext Attack

A naive attack might try to reveal the keys that were used to generate the permutation b. However, anyone who knows the permutation b 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.

3.1 Causality in decryption

The function g in Eq. (3) 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 Eq. (3), 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. This is a desirable property of an encryption and is also known as the diffusion property [Menezes et al., 1996].

However, the situation is quite different in decryption. Using Eq. (5), 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 Eq. (5) with c_k as the plaintext pixel that is input to second round, we obtain

$$c_k = d_{u(k)} - g(d_{u(k)-1}) - h_{u(k)} \bmod 256,\ 1 \le k \le n. \tag{6}$$

Substituting $k = u(j)$ in Eq. (6), we find

$$c_{u(j)} = d_{u^2(j)} - g(d_{u^2(j)-1}) - h_{u^2(j)}. \tag{7}$$

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

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

$$c_{u(j)-1} = d_{u(u(j)-1)} - g(d_{u(u(j)-1)-1}) - h_{u(u(j)-1)}. \tag{8}$$

Thus, we see from Eq. (5), Eq. (7) and Eq. (8) 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 1 *We illustrate the causal relations in the decryption for two rounds. Here, $n = 6$ and the permutation u is given as*

$$u = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 1 & 5 & 6 & 3 \end{pmatrix}. \tag{9}$$

The causality paths are given in Fig. 1. 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 causality chain from the ciphertext $\mathbf{d}$ *to the plaintext* $\mathbf{p}$ *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 causality paths going from d_5 to p_4.

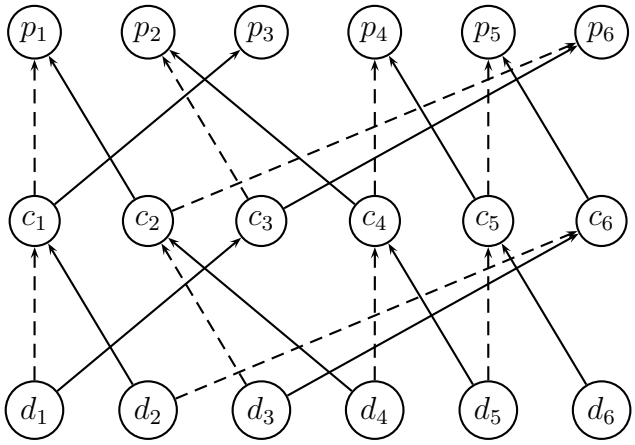

Figure 1: The causality paths for the permutation given in Eq. (9). A solid arrow indicates that the causality is through u, while a dashed arrow indicates that the causality is through $u-1$.

3.2 Detecting causality 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 $\mathbf{z}$, the ciphertext image after R rounds of encryption. The attacker wants to know if there is a causality path from the ciphertext pixel z_i to the plaintext pixel p_j. Assume that the attacker knows a plaintext-ciphertext image pair $(\mathbf{p}, \mathbf{z})$. 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 causality 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 causality paths from z_i to p_j. This is due to the nonlinearity of encryption/decryption that operates in a finite domain. In order to detect all the causality paths, the attacker needs to try more than one change to z_i. It is highly unlikely that p_j remains fixed for all of these changes. Thus, in order to construct the matrix T for n pixel images, the attacker needs to choose $O(n)$ ciphertext images.

Detecting changes for all i, $1 \leq i \leq n$, the attacker constructs a binary matrix T showing the causality 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 $\mathbf{z}$, each column of T contains at most 2^R 1's. All the other entries are zero.

Example 2 *The matrix T for the permutation u used in Example 1 is given as*

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

3.3 Finding $b(1)$.

Writing Eq. (5) 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 causality 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 2, the attacker can see that the third column has the least sum. Thus, he concludes that $u(3) = 1$.

3.4 Tree of causality

In order to generalize the attack to the rest of u, we define an operation to denote the causality 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 Eq. (5), 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

$$L(\{k\}) = \begin{cases} \{u(k)\} & \text{if } u(k) = 1, \\ \{u(k), u(k) - 1\} & \text{otherwise} \end{cases} \tag{10}$$

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 3 *For the permutation given in Example 1, we have*

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

3.5 Overlapping sets of leaves

Using the chosen-ciphertext attack given in the beginning of this section, the attacker constructs the matrix T. This is the same as attacker knowing the sets $L^R(k)$, $\forall k \in \{1, 2, \ldots, n\}$. The attacker uses this knowledge to reveal the secret permutation u. First, we need the following fact.

Lemma 4 *Let x, y and z be integers in $\{1, 2, \ldots n\}$ such that they satisfy*

$$\begin{aligned} u(x) + 1 &= u(y), \\ u(y) + 1 &= u(z). \end{aligned}$$

Then, for every positive integer R larger than 1,

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

Proof. By the definition of L, we have for every integer a in $\{1, 2, \ldots n\}$,

$$L^R(a) = \begin{cases} L^{R-1}(u(a)) & \text{if } u(a) = 1, \\ L^{R-1}(u(a)) \cup L^{R-1}(u(a) - 1) & \text{otherwise.} \end{cases} \tag{11}$$

For the case $a = y$, we have $u(y) \neq 1$, and therefore

$$\begin{aligned} L^R(y) &= L^{R-1}(u(y)) \cup L^{R-1}(u(y) - 1), \\ &= L^{R-1}(u(y)) \cup L^{R-1}(u(x)). \end{aligned} \tag{12}$$

Similarly, for $a = z$,

$$L^R(z) = L^{R-1}(u(z)) \cup L^{R-1}(u(y)). \tag{13}$$

Finally, for $a = x$,

$$L^R(x) = \begin{cases} L^{R-1}(u(x)) & \text{if } u(x) = 1, \\ L^{R-1}(u(x)) \cup L^{R-1}(u(x) - 1) & \text{otherwise.} \end{cases} \tag{14}$$

Using Eq. (12) and Eq. (14) together with the identity $A \setminus B = A \cap \overline{B}$, we obtain

$$L^R(y) \setminus L^R(x) = \begin{cases} L^{R-1}(u(y)) \cap \overline{L^{R-1}(u(x))} & \text{if } u(x) = 1, \\ L^{R-1}(u(y)) \cap \overline{L^{R-1}(u(x))} \cap \overline{L^{R-1}(u(x) - 1)} & \text{otherwise.} \end{cases} \tag{15}$$

Note that, we take the set complements with respect to the universal set $\{1, 2, \ldots n\}$. Comparing Eq. (15) with Eq. (13), the result immediately follows. Figure 3.5 illustrates the overlap and the intersection of the sets $L^R(a)$ and $L^{R-1}(a)$. ■

Lemma 5 *Let x and y be integers such that $u(x) = 1$ and $u(y) = 2$. Then, $L^R(x) \subset L^R(y)$.*

Proof. Using Eq. (11) with $a = x$, we obtain $L^R(x) = L^{R-1}(u(x)) = L^{R-1}(1)$. Using Eq. (11) once more with $a = y$, and applying the assumptions of the lemma, we obtain $L^R(y) = L^{R-1}(2) \cup L^{R-1}(1)$. The result immediately follows. ■

3.6 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.

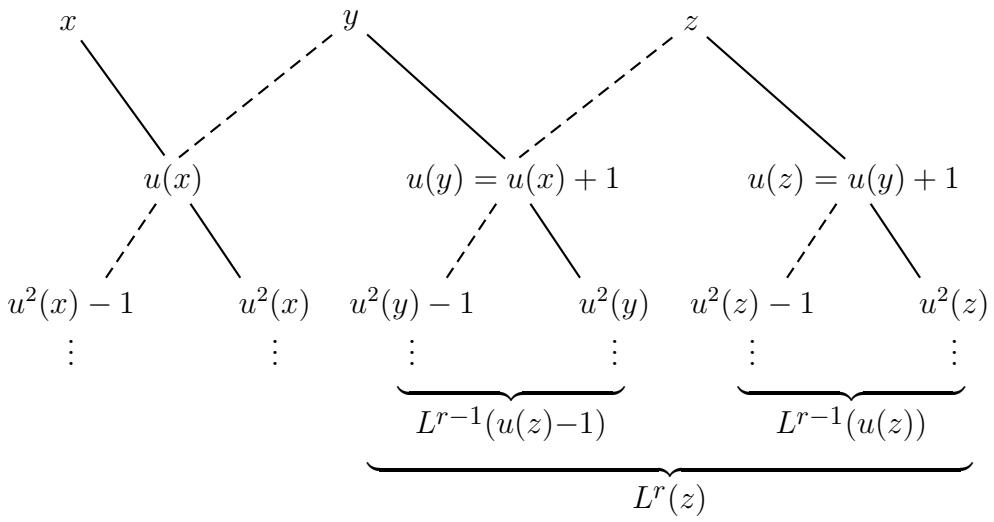

Figure 2: The sets $L^R(a)$ and $L^{R-1}(a)$. Note that the sets are the leaves of overlapping causality trees.

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 5, $x_2 \in X_2$. 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 4, $x_3 \in X_3$. 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 3, 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.

The space complexity of the attack depends on the memory required to store the matrix T. Assuming that we have a $N \times M$ image and R rounds, T is stored in N^2M^2 bits. However, T is sparse when the attack is feasible. Therefore the memory requirement drops down to $O(2^R NM)$ bits.

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 ← {x | L^R(b(i)) \ L^R(b(i−1)) ⊂ L^R(x)} \ {b(1), b(2), ..., b(i)}
4      i ← i + 1
5      if Z ≠ ∅ then
6          foreach z ∈ Z do
7              b(i) ← z
8              if i = n then
9                  exit
10             FindNext()
11         i ← i − 1
12     else
13         return
14 end
```

4 Simulations

We first 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 5 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.

We now give the results of the attack for an image size of 256×256.

We used $R = 10$ as suggested in [Fridrich, 1998]. We generated the permutation u using the Baker's Map with the keys as $K = [2 \quad 4 \quad 2 \quad 8 \quad 4 \quad 8 \quad 4]$. The fixed functions g and h are again chosen randomly.

Simulations are performed under GNU gcc compiler running on Mac OS X 10.5.4 with Intel Core 2 Duo 2.16 GHz processor and 2 GB RAM. The whole map u is recovered successfully in less than 20 minutes. Moreover, the total memory requirement for the whole implementation was approximately 1 GB. In order to make the Algorithm 1 run more efficiently, we used a sparse matrix data structure for the matrix T. Thus, we reduced the search time needed for constructing the set Z in Line 3 of the algorithm.

5 Conclusions

Since its proposal in 1998, Fridrich's chaotic image encryption has generally been considered secure. Indeed, many image encryption algorithms have taken it as a benchmark against which their efficiencies are measured.

In this paper, we gave a cryptanalysis of Fridrich's chaotic image cipher. We demonstrated that rather than attacking the underlying secret keys, an attacker can instead reveal the secret permutation. In attacking the permutation, the attacker uses the overlaps among the sets of indices of ciphertext pixels that affect related plaintext pixels. We demonstrated the success of the attacks on an image of realistic size.

The chosen-ciphertext attack that we proposed can be applied to other encryption algorithms that have a structure similar to the one in [Fridrich, 1998]. Naturally, every encryption algorithm has a particular algebraic structure that has to be taken into account in its cryptanalysis. Here, for the sake of illustration, we briefly discuss a particular application of our proposed attack.

One of the well-known chaotic image encryption algorithms in the literature is the one proposed in [Chen et al., 2004]. Similar to [Fridrich, 1998], the algorithm consists of multiple rounds of permutation and diffusion. The permutation step is implemented using a 3D cat map. Similar to [Fridrich, 1998], the diffusion step mixes consecutive pixels of the permuted image using a chaotically modulated XOR function. Although a 3D chaotic map provides a better scattering of pixels compared to 2D maps, our attack is still applicable in this case because we do not

make any assumptions on how the secret permutation b in Eq. (5) is obtained. Likewise, our attack does not depend on any assumptions on the nonlinear function g in Eq. (5). Hence, our attack can be launched without modification against the algorithm proposed in [Chen et al., 2004].

The success of our attack mainly depends on the small size of the set Z on Line 3 of Algorithm 1. For realistic image sizes and a 10-round encryption, Z contains only a single element in most cases. If the number of rounds R is increased, the sets $L^R(x)$ contain more and more elements. Assuming that the causality paths do not have overlaps, the set $L^R(x)$ contains 2^R elements. When R is increased to $R = \log_2 n$, $L^R(x)$ contains n elements, i.e. $L^R(x) = \{1, 2, \ldots, n\}$. Thus, Z in Line 3 of the algorithm contains all the unrevealed elements. Therefore, the number of trials in Line 6 of the attack algorithm increases to n. This makes the present attack infeasible.

Although the increase of rounds appear to suggest an obvious defense against our attack, larger images would still present a problem because in that case the sets Z would become smaller making the attack feasible again. One solution might be to break the images into smaller fragments and encrypt each fragment using a fixed number of rounds. If each partition has n_0 pixels, in order to resist our attack, n_0 has to be smaller than 2^R.

Modifying the way the pixels are passed through the nonlinear function might also provide a defense against our attack. In the proposed scheme, for one round, a plaintext pixel is affected by at most two ciphertext pixels. This dependence can be increased by several methods. One particularly easy method would be to nonlinearly mix more than two pixels. Indeed, if m pixels of ciphertext affect one pixel of plaintext, then having R satisfy $R > \log_m n$ would provide a dependence complicated enough to preclude the present attack.

Another possible defense is to modify the permutation function b in each round. This can be done using a key scheduling mechanism that generates R different permutations depending the key.

Of course, each of these defenses require a rigorous analysis of the whole new algorithm for weaknesses and statistical properties.

6 Acknowledgments

Ercan Solak and Cahit Çokal are supported by the The Scientific and Technological Research Council of Turkey (TÜBİTAK) under Project No. 106E143. Türker Bıyıkoğlu is supported by Turkish Academy of Sciences through Young Scientist Award Program (TÜBA-GEBİP/2009).

The authors thank an anonymous referee for suggestions to improve the clarity in various parts of the paper and for pointing out possible generalizations of

methods developed in the paper.

References

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

J. M. Amigo, L. Kocarev, and J. Szczepanski [2007] "Theory and practice of chaotic cryptography," *Physics Letters A* **366**(3), 211–216.

D. Arroyo, R. Rhouma, G. Alvarez, S. J. Li, and V. Fernandez [2008] "On the security of a new image encryption scheme based on chaotic map lattices," *Chaos* **18**(3), 033112.

G. R. Chen, Y. B. Mao, and C. K. Chui [2004] "A symmetric image encryption scheme based on 3D chaotic cat maps," *Chaos Solitons Fractals* **21**(3), 749–761.

F. Dachselt and W. Schwarz [2001] "Chaos and cryptography," *IEEE Tran. Circuits and Systems – I: Fundamental Theory and Applications* **48**(12), 1498–1509.

J. Fridrich [1998] "Symmetric ciphers based on two-dimensional chaotic maps," *Int. J. Bifurcation and Chaos* **8**(6), 1259–1284.

K. Kelber and W. Schwarz [2007] "Some design rules for chaos-based encryption systems," *Int. J. Bifurcation and Chaos* **17**(10), 3703–3707.

S. Li, C. Li, G. Chen, and K.-T. Lo [2008] "Cryptanalysis of the RCES/RSES image encryption scheme," *Journal of Systems and Software* **81**(7), 1130–1143.

L. Liu, X. G. Wu, and H. P. Hu [2004] "Estimating system parameters of Chua's circuit from synchronizing signal," *Physics Letters A* **324**(1), 36–41.

Y. B. Mao, G. R. Chen, and S. G. Lian [2004] "A novel fast image encryption scheme based on 3D chaotic baker maps," *Int. J. Bifurcation Chaos* **14**(10), 3613–3624.

N. Masuda and K. Aihara [2002] "Cryptosystems with discretized chaotic maps," *IEEE Tran. Circuits and Systems – I: Fundamental Theory and Applications* **49**(1), 28–40.

N. Masuda and K. Aihara [2002] "Dynamical characteristics of discretized chaotic permutations," *Int. J. Bifurcation and Chaos* **12**(10), 2087–2103.

N. Masuda, G. Jakimoski, and L. Kocarev [2996] "Chaotic block cipphers: from theory to practical algorithms," *IEEE Tran. Circuits and Systems – I: Regular Papers* **53**(6), 1341–1352.

A. Menezes, P. van Oorschot, and S. Vanstone [1996] *Handbook of Applied Cryptography.* CRC Press.

S. Ozoguz, A. S. Elwakil, and S. Ergun [2006] "Cross-coupled chaotic oscillators and application to random bit generation," in *IEEE Proceeding – Circuits Devices and Systems* pp. 506–510.

A. T. Parker and K. M. Short [2001] "Reconstructing the keystream from a chaotic encryption scheme," *IEEE Tran. Circuits and Systems – I: Fundamental Theory and Applications* **48**(5), 624–630.

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

J. Szczepanski, J. M. Amigo, T. Michalek, and L. Kocarev [2005] "Cryptographically secure substitutions based on the approximation of mixing maps," *IEEE Tran. Circuits and Systems – I: Regular Papers* **52**(2), 443–453.

T. Xiang, X. F. Liao, G. P. Tang, Y. Chen Y, and K.-W. Wong [2006] "A novel block cryptosystem based on iterating a chaotic map," *Physics Letters A* **349**(1-4), 109–115.

T. Yang, C. W. Wu, and L. O. Chua [1997] "Cryptography based on chaotic systems," *IEEE Tran. Circuits and Systems – I: Fundamental Theory and Applications* **44**(5), 469–472.

T. Yang, L. B. Yang, and C. M. Yang [1998] "Application of neural networks to unmasking chaotic secure communication," *Physica D* **124**(1-3), 248–257.