SECURITY OF CHAOTIC CRYPTOSYSTEMS

CAHIT ÇOKAL
B. S., Electronics Engineering, Işık University, 2006

Submitted to the Graduate School of Science and Engineering
in partial fulfillment of the requirements for the degree of
Master of Science
in
Computer Engineering

IŞIK UNIVERSITY
2008

IŞIK UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

SECURITY OF CHAOTIC CRYPTOSYSTEMS

CAHIT ÇOKAL

APPROVED BY:

Assoc. Prof. Ercan SOLAK (Thesis Supervisor)	(Işık University)	————————
Assist. Prof. Olcay Taner YILDIZ	(Işık University)	————————
Assist. Prof. Onur KAYA	(Işık University)	————————

APPROVAL DATE:

Abstract

In this thesis, we tried to show the weaknesses of chaotic cryptosystems. We break four chaos-based cryptosystems and proved our attacks.

In our first cryptanalysis, we broke a cryptosystem based on two dimensional chaotic maps. We first reveal a portion of the secret key using a chosen-ciphertext attack. After revealing this porion, we used it to reveal the other portions of the secret key. We developed three types of attack using algebraic properties of the permutations in revealing the rest. We finally published two papers for this break.

In our second cryptanalysis, we broke a cryptosystem that encrypts and decrypts images with chaotic map lattices. Here we first show that the encryption algorithm is not invertible for some cases. We showed why these cases not work, and gave some suggestions to improve the algorithm. However, we showed that the algorithm still is not invertible due to finite precision arithmetic. Furthermore, we gave some suggestions to develop the algorithm. At the end of our work, we gave a break for the modified algorithm. Finally, we published a comment for the wrong cases.

In our third cryptanalysis, we broke a chaos-based image encryption algorithm, which uses a two-dimensional chaotic map to shuffle the image pixels and a three-dimensional chaotic map to change the gray levels of the pixels. We used a chosen-plaintext attack and a known-plaintext attack to break the algorithm. Applying either our chosen-plaintext attack or our known-plaintext attack the cryptosystem yields the secret parameters successfully. We published a paper for this break.

Our final cryptanalysis was on an image encryption algorithm based on two-dimensional chaotic maps. We showed that the chaotic map can be revealed using a chosen-ciphertext attack. The attack does not depend on which type of map is used. The attack uses some algebraic properties of permutations and graphs.

Özet

Bu tezde kaotik şifreleme algoritmalarının zayıflıklarını göstermeye çal ıştık. Dört tane kaotik şifreleme algorimasını kırdık ve ataklarımızı ıspatladık.

İlk analizimizde iki boyutlu kaotik fonksiyonlarla çalışan bir algortmayıkırdık. İlk olarak gizli şifrenin bir kısmını önceden belirlenmiş şifrelenmiş mesaj atak metodunu kullanarak buluyoruz. Bu kısmı bulduktan sonra, bulunmuş kısmı kullanarak kalan kısımlarını bulmaya çalışıyoruz. Permutasyonların matematiksel özelliklerini kullanarak kalan kısmı bulmak için üç tane atak ürettik. Kırma işlemlerimizi gösteren iki adet makale ile yayınladık.

İkinci analizimizde tek boyutlu kaotik fonksiyonları kullanarak resim şifreleyen bir algoritmayı kırdık. İlk olarak algoritmanın bazı durumlarda geri dönüşü mümkün olmayan sonuçlar ürettiğini gösterdik. Bunların sebeplerini ve bu durumları düzeltmek için önerilerimizi açıkladık. Fakat bu düzeltmelere rağmen algoritmada sonlu sayı aritmetiği kullanıldığı için, düzeltilmesi imkansız olan durumlar oladuğunu gösterdik. Daha sonra, algoritmayı düzenlemeye yönelik bazı öneriler doğrultusunda algoritmayı yeniden kurguladık. Son olarak, yeniden kurgulanmış olan bu algoritmayı kırdık. Algoritmadaki geri dönüşümü mümkün olmayan bu durumları göstermek icin yazılmış olan makaleye atıfta bulunduk.

Üçüncü analizimizde iki boyutlu ve üç boyutlu kaotik fonksiyonları kullanarak resim şifreleyen bir algoritmayı kırdık. Algoritma iki byutlu fonksiyonu resmin piksellerini karistirmak ın, üç boyutlu olanını da resmin gri değerlerini değisştirmek için kullanıyor. Onceden belirlenmiş mesajları şifreleme metodunu kullanan atağı uygulayarak gizli şifrenin bulunabildiğini gösterdik. Ayrıca önceden belirlenmemiş fakat önceden bilinen mesajları şifreleme metodunu kullanan atak ile de aynı işlemin yapılabildiğini de gösterdik. Son olarak, bu çalışmalarımızı gösteren bir makale yayınladık.

Son analizimiz iki boyutlu kaotik fonksiyonları kullanan bir resim şifreleme algoritması üzerine idi. Önceden belirlenmiş şifrelenmiş mesaj atak metodunu kullanarak gizli şifreyi bulabileceğimizi gösterdik. Atak kullanılan kaotik fonksiyona bağli değildir. Bu atakta permutasyonarın ve grafiklerin bazı özelliklerini kullanıyoruz.

Acknowledgements

First of all, I am glad to finish my Master of Science in Computer Engineering, in Işık University.

I am very pleased to work with my supervisor Assoc. Prof. Ercan SOLAK during my masters. I thank him due to his help and patience. He helped me to learn anything about the project regularly. Even he gave me many advices while taking my graduate courses. Also, I am glad to be his teaching assistant in undergraduate laboratory courses.

I also thank Prof. Yorgo ISTEFANOPULOS, the Dean of Engineering Faculty of Işık University, and my friends Uğur KIRMIZIBEKMEZ and Ahmet SOYLU for their advices to choose Işık University for Master of Science. I thank all my other friends for their encouragement and support during my graduate studies.

This work was supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under Project No. 106E143, and I thank TÜBİTAK for scholarship I got through the project.

My family also encouraged and supported me spiritually and financially during my Master of Science, so I thank them for their kindness.

Finally, I thank my wife for her endless support.

To Mercan...

Table of Contents

List of Figures

List of Abbreviations

ECB	:	Electronic Codebook
CBC	:	Cipher Block Chaining
CTR	:	Counter
CFB	:	Ciphertext Feedback
OFB	:	Output Feedback
SPN	:	Substitution Permutation Networks
DES	:	Data Encryption Standard
FIPS	:	Federal Information Processing Standard
NSA	:	National Security Agency
AES	:	Advanced Encryption Standard
TDCM	:	Two-Dimensional Chaotic Maps
ROL	:	Rotate Left
ROR	:	Rotate Right
TEA	:	Tiny Encryption Algorithm
D/A	:	Digital to Analog
A/D	:	Analog to Digital

Chapter 1
Introduction

Cryptography is the science of concealing a message during communication. Plaintext is the raw message, which the sender wishes to transmit to the receiver(s). Encryption is the process of transforming plaintext using an algorithm to make it unreadable to anyone except those possessing special knowledge. The resulting output of the encryption process is called as ciphertext. Decryption is the reverse of the encryption algorithm to recover the concealed message. To transform either the plaintext into ciphertext or ciphertext into plaintext we need encryption and decryption algorithms.

Transposition means the movements of plaintext characters into new positions in the ciphertext based on the algorithm. Swapping is a transposition method, for example. Certainly, there can be more than one transposition inside the algorithm, if needed.

Substitutions are done by using mapping techniques. Mapping is a method of replacing the characters (or blocks) by other characters (or blocks). The method can be done by inversion, displacement or shift of the characters. The mapping technique can be more complex. For example, characters are encoded with binary sequences in computers, and these sequences can be manipulated via boolean operations for the encryption [1].

Encryption and decryption may be done by using a symmetric key or asymmetric keys, private and public keys. In symmetric encryption both the transmitter and the receiver uses the same secret key. On the other hand, public key cryptography uses a public and a private key for each side. The plaintext is encrypted via the public key of the receiver side, and the receiver decrypts the ciphertext by using his own private key. Public keys are published and anyone can reach it.

The first known use of cryptography was done by Julius Caesar. During World War 2, cryptology was advanced rapidly. Machines were produced only for encryption and decryption of messages. Enigma cipher machine is invented by the Germans. It was broken successfully by the Polish before the war had even started. British used the Polish methods to break the Enigma messages during the war [2].

Cryptology relies heavily on advanced mathematics, number theory, in the

modern world. Hence, those wishing to study modern cryptology are advised to study advanced mathematics.

1.1 Block Ciphers

Block ciphers may be either symmetric-key or public-key. The main focus of this section is symmetric-key block ciphers; public-key encryption is addressed later in this chapter.

In block ciphers, the transmitter side divides the plaintext into n-bit blocks and encrypts them into n-bit ciphertext blocks. The recipient side computes the same plaintext blocks by decrypting the received ciphertext blocks. Because the process is a symmetric encryption-decryption, the key used by the receiver and transmitter are the same. The block ciphers are the same as the classic codebook method, except the book. In classical codebook method, there is a list of the words and the codewords tuples, and the book should be kept securely against attackers. However, in the block ciphers the codewords are computed via an algorithm, instead of having a book, due to the fact that the alphabet size is too large to be stored in a book. The only thing to be kept secure is the symmetric key [3].

The content of the codebook depends on the key. A change in key causes a very effective change in the codebook. There are 2^k different codebooks assuming that the key is k-bit long. Hence, by changing the key we can avoid the codebook attack.

Block ciphers can be used in different ways, called modes of operations, according to the connections between the plaintext or ciphertext blocks. Some of the popular modes of operations are; electronic codebook mode (ECB), cipher block chaining mode (CBC), counter mode (CTR), ciphertext feedback mode (CFB), output feedback mode (OFB) and so on. There are detailed explanations of modes of operations in [3, 4, 5, 6].

1.1.1 Substitution Permutation Networks (SPN)

For a secure block cipher, confusion and diffusion are two sine qua non properties, which were identified in the paper of Claude Shannon [7], "Communication Theory of Secrecy Systems" published in 1949.

Confusion is defined, in Shannon's original definition, as making the relationship between the key and the ciphertext as complex and involved as possible, and confusion is defined to be the property that redundancy in the statistics of the plaintext is "dissipated" in the statistics of the ciphertext. Namely, diffusion is the dependency of output bits on input bits.

A substitution permutation network (SP network or SPN) is a series of linked mathematical operations used in block ciphers. S-boxes and P-boxes are used, in these networks, to transform blocks of input bits into output bits. Many modern-day cryptosystems use product cipher methods. Product ciphers are a combination of permutation and substitution operations to achieve both confusion and diffusion respectively. They commonly use iterated cipher methodology today. Iterated cipher means that the cipher consists of a round function and a key scheduling algorithm. The cryptosystem proceeds the same round operations N_r times, where N_r is the number of rounds.

A well-designed S-box should have the feature that at least half of the output bits change when only a bit is inverted from the input (this is termed the Strict Avalanche Criterion). This also means that each output bits depends on every input bits. This also is an essential property for S-boxes. Either a single S-box is so narrow that it provides a limited amount of confusion nor a P-box do. Even though they provide limited confusion individually, a well-designed SPN having enough rounds causes good diffusion and confusion so that every input bit is fully diffused across every output bit of the whole message.

Let us use a random binary key K with a previously defined length. There should be N_r round keys (also called subkeys), produced by using K and the key scheduling algorithm. The key schedule (round keys) is denoted by the list $(K^1, ..., K^{N_r})$. The key scheduling algorithm is public.

The round function, say F, needs two inputs; a round key (K^r) and a current state, denoted by w^{r-1}. To calculate the next state, we use the procedure; $w^r = F(w^{r-1}, K^r)$. The Plaintext P is defined to be the initial state, w^0, and the ciphertext, C, is defined to be the final state, w^{N_r}.

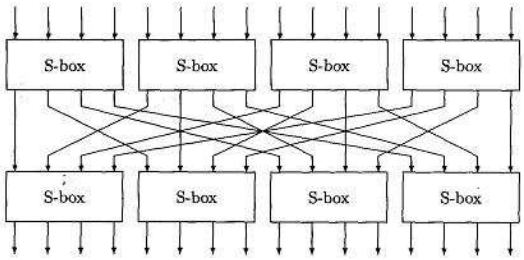

Figure 1. 1 Substitution permutation network

Figure 1. 1 shows a single round of iterated block cipher encryption function. The figure includes substitutions via S-boxes and permutations.

1.2 Classical Block Cipher Examples

1.2.1 Data Encryption Standard

Data Encryption Standard (DES), was jointly developed in 1974 by IBM and USA government (US patent 3,962,539), is an example of symmetric secret key encryption scheme, and became a standard in 1977. The Encryption scheme uses a 56-bit key, can be cracked by only a brute force attack with a moderate effort. Therefore, it is considered to be expired by many cryptologists. A variance of the DES, named as Triple-DES (3DES or TDES), is used today, and is known to be more secure. The block size of the plaintext and ciphertext blocks are 64-bits long.

The original algorithm was using 64-bit key and started out as the "Lucifer" algorithm developed by IBM. After it was adopted as Federal Information Processing Standard (FIPS) standard 46-3 and ANSI standard X3.92, the US National Security Agency (NSA) made several modifications and reduced the key size to 56-bits.

Description of DES

DES is a block cipher that transforms fixed-length string of plaintext bits into a ciphertext cit string of same length using a public algorithm, consisting a series of complicated operations [5]. The size of the plaintext and ciphertext blocks is 64 bits. DES uses a secret symmetric key either in encryption and decryption. The key appears to be 64-bits long; however, the used part of the key is only 56-bits. The unused eight bits are used for parity check merely, and are discarded thereafter. Hence, the effective key length is 56-bits.

Overall Structure

There are 16 rounds, which are identical. The algorithm consists also an initial and a final permutation, termed IP and FP, aside from 16 rounds. IP and FP are inverses of each other. Namely, FP undoes the procedure of IP.

After IP, the plaintext block is split into two halves, and the processes are applied alternately to them. This is the Feistel method [3], and the method ensures that the encryption and decryption algorithms are very similar. The only difference of the decryption algorithm is that the subkeys are applied in reverse order. This structure simplifies the hardware implementation due to the fact that there is no need for different encryption and decryption algorithms.

1.2.2 Advanced Encryption Standard

The Advanced Encryption Standard (AES), is the winner of the contest, held in 1997 by the US Government, after the Data Encryption Standard was found too

weak because of its small key size and the technological advancements in processor power. Fifteen candidates were accepted in 1998 and based on public comments the pool was reduced to five finalists in 1999. In October 2000, one of these five algorithms was selected as the forthcoming standard. The winner algorithm was a slightly modified version of the Rijndael's.

1.2.3 Description of AES

AES is an iterated block cipher with a fixed block size of 128 and a variable key length. The different transformations operate on the intermediate results, called state. The state is a rectangular array of bytes and since the block size is 128 bits, which is 16 bytes, the rectangular array is of dimensions 4×4. The cipher key is similarly pictured as a rectangular array with four rows. The number of columns of the cipher key is equal to the key length divided by 32 [6].

The key schedule and the encryption algorithm involves several operations and work similarly. The operations of the encryption algorithm are explained in the following section.

1.2.4 Encryption Algorithm of AES

The algorithm involves four stages which are called as, SubBytes, ShiftRows, MixColumns and AddRoundKey. The operations are so simple that their hardware or software implementations are very easy. AES has 10 identical rounds. The only difference is that the tenth round omits the MixColumns step. This modification makes the encryption and decryption algorithm identical.

Step 1. In **SubBytes** step, the algorithm uses a fixed S-box, given in [6], and substitutes the bytes of the state with the bytes in the fixed S-box. The S-Box is invertible and is constructed by the composition of two transformations. the first transformation is taking the multiplicative inverse in Rijndael's finite field [6]. The second one applies an affine transformation which is documented in the Rijndael documentation.

Step 2. In **ShiftRows** step, the rows of the state are shifted to the left with fixed amounts. The first row is shifted by 0 byte, the second row is shifted by 1 byte, the third row is shifted by 2 bytes and the last row is shifted by 3 bytes.

Step 3. In **MixColumns** step, the four bytes of each column of the state are combined using an invertible linear transformation. The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with ShiftRows, MixColumns provides

diffusion in the cipher. The operations of MixColumns step is described in detail in [6].

Step 4. In **AddRoundKey** step, the subkey is combined with the state using simple bitwise XOR operation. The subkey is derived from the cipher key by the means of the key schedule [6].

1.3 Attack Types and Cryptanalysis

Cryptographic attacks are designed to subvert the security of cryptographic algorithms, and they are used to attempt to decrypt data without prior access to a key. Cryptanalysis deals with the weaknesses of cryptosystems.

Here I will try to describe some common attack types. There are many applied attack types in [3]. You can see many more attacks in [3] other than those I describe.

1.3.1 Cryptographic Attacks

There are six related cryptographic attack methods; three plaintext-based methods and three ciphertext-based methods. The plaintext based methods are: known plaintext attack, chosen plaintext attack and adaptive chosen plaintext attack. The ciphertext based methods are: ciphertext-only attack, chosen ciphertext attack and adaptive chosen ciphertext attack.

Known-Plaintext Attack

A known-plaintext attack is a cryptographic attack in which an attacker has the plaintext and the corresponding ciphertext.

Ciphertext-Only Attack

Here, the attacker is assumed to have access only to a set of ciphertexts. The attack is completely successful if the corresponding plaintexts can be deduced. Even the attacker may reveal the secret key, which is a better result. The ability to obtain any information at all about the underlying plaintext is still considered a success. With simple ciphers, such as the Caesar Cipher, frequency analysis can be used to break the cipher [8].

Chosen-Plaintext Attack

In a chosen plaintext attack the attacker has the ability to obtain the ciphertexts of the plaintexts of his choice. This appears, at first glance, to be an unrealistic model. It certainly is unlikely that an attacker could force a human cryptographer

to encrypt large amounts of plaintexts of the attacker's choice. On the other hand, modern cryptography is implemented in software or is hardwired and is used for a diverse range of applications. For many cases, it is often very feasible to apply a chosen-plaintext attack.

In both adaptive attacks, a cryptanalyst chooses further plaintexts or ciphertexts (adapts the attack) based on prior results.

Chosen-Ciphertext Attack

A chosen-ciphertext attack is an attack on a cryptosystem in which the cryptanalyst chooses ciphertext and causes it to be decrypted with an unknown key and accesses the corresponding plaintext. As I mentioned in chosen plaintext attack, even though the attack appears unlikely to be performed, it usually is very feasible to perform due to the modern technology and the implementations.

The adaptive chosen-ciphertext attack is an interactive form of chosen-ciphertext attack in which an attacker sends a number of ciphertexts to be decrypted, then uses the results of these decryptions to select subsequent ciphertexts

1.3.2 Brute Force Attacks

A brute force attack systematically attempts every possible key. It is most often used in a known plaintext or ciphertext-only attack. Given a finite key length and sufficient time, a brute force attack is always successful. Encryption algorithms can become susceptible to brute force attacks over time as CPU speeds increase. The easiest remedy for the brute force attack is to increase the key length. For example, 56-bit DES key can be cracked within days using specialized hardware. However, if a machine could crack one DES key per second, it would take 149 trillion years to crack a 128-bit AES key [8].

1.4 Chaos and Cryptography

The most attractive feature of deterministic chaotic systems is the highly unpredictable and random-looking nature of chaotic signals. There are some common features of chaos and cryptography such as sensitivity to initial conditions and parameters, random like behavior and unstable orbits with long periods, depending upon the precision of the numerical implementation. In cryptography, rounds of encryption lead to the desired diffusion and confusion properties of the algorithm. Similarly, chaotic map iterations spread the initial region over the entire phase space. This is similar to the diffusion and confusion properties since the parameters of the chaotic map may represent the key of the encryption algorithm [9, 10].

Below items show the differences between classical and chaotic cryptography simply.

1. Classical cryptography works with integer values on finite fields, however chaotic cryptography works with continuous values using fixed or floating point representation.

2. Classical cryptography makes digital realization by integer arithmetic, however chaotic cryptography makes digital realization by non integer arithmetic.

A block encryption algorithm can be written as a discrete time dynamical system:

$$x_{n+1} = F(x_n)$$

where x_0, the initial condition, is the plaintext to be encrypted, and the final state x_k is the ciphertext. Since we can express a chaotic map by a similar equation. This property shows that a slight change in a digit of an initial condition spreads over almost the whole output [9, 10].

There are many types of chaotic maps in the classes of one-dimensional, two-dimensional and three-dimensional. In this report I give some cryptosystems based on some of these types.

Chapter 2
Our Methodology of Cryptanalysis

2.1 The Purpose of the Thesis

Computer science is developing rapidly, consequently, it is getting easier to attack a cryptosystem, for an attacker. There are many attack types introduced by cryptanalysts as well, from day to day. Hence, there is a considerable need for cryptanalysts to test and protect cryptosystems, in use, against these new attacks and improvements, before the attackers do.

One of the areas of cryptology, is chaos, and there are lots of proposed chaotic cryptosystems in literature. It is necessary to analyze the security of chaotic cryptosystems.

Our aim was to analyze the security of some chaos-based cryptosystems, using the classical attack types and new attacks if possible. It is possible to produce new attack types using mathematical features of the chaotic functions used in cryptosystems. In the case of introducing new attack types, the attacks may be generalized to all the chaotic cryptosystems.

Chapter 3
Cryptanalysis of an Algorithm that Uses Discretized Two Dimensional Chaotic Maps

A chaotic cryptosystem [11], based on two-dimensional chaotic maps, was published on Physics Letters A. In this chapter, I describe our cryptanalysis and a complete break [12, 13] of the proposed cryptosystem. The cryptosystem involves discretized two-dimensional chaotic maps (TDCM) and chaotic S-boxes inside the round functions.

The definitions of proposed TDCM are given in Section 3.1. Then I give our description for the proposed cryptosystem in detail, in Section 3.2. There are symbolic differences between the proposed algorithm and our description. The differences make the algorithm easier to understand. In Section 3.3, I show that the secret key contains redundancies and some interrelated parameters inside. These facts reduce the effective key length. Next, I demonstrate our chosen ciphertext attack, which reveals a portion of the key, in Section 3.4. The attack varies with respect to the round number. In Section 3.5, I describe three different attacks, which can be applied in combination. The names of the attacks are the permutation orbit, expansion and skipping attacks. The attacks effectively yield the rest of the key when they work in a combination. We produced the attacks using some features of permutations stated in the section. In Section 3.6, I illustrate the success of our break with simulation results. All types of attacks are used in given examples. Finally, I give our concluding remarks about the break, in Section 3.7.

3.1 Two-Dimensional Chaotic Maps

The Standard map [14] (also known as Chirikov-Taylor map or Chirikov standard map) is a two dimensional chaotic map. The TDCM is an area-preserving chaotic map from a square with side 2π onto itself.

The map is discretized for use in the proposed cryptosystem. The discretized version of Standard Map is given in [11] as

$$\begin{aligned} x_{n+1} &= x_n + y_{n+1} \mod N \\ y_{n+1} &= y_n + K\sin\frac{2\pi x_{n+1}}{N} \mod N \end{aligned} \tag{3. 1}$$

We may prefer round function in order to have integer values for discretized Standard Map. Result of second line in Eq.(3. 2) should be rounded.

Arnold's Cat Map is described in [15]. It is possible to define a discrete analogous of the cat map. One of this map's features is that image being apparently randomized by the transformation but returning to its original state after a number of steps. The discretized Cat Map is given in [11] as

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \mod N. \tag{3. 2}$$

where the determinant $(a_{11} \times a_{22} - a_{12} \times a_{21}) = 1$.

Baker's Map is a chaotic bijection of the unit square $I \times I$ onto itself, and is described in [16]. The discretized version of Baker's Map is given in [11] as

$$\begin{aligned} x_{n+1} &= \frac{N}{k_j}(x_n - N_j) + y_n \mod N/k_j, \\ y_{n+1} &= \frac{k_j}{N}(y_n - y_n \mod N/k_j) + N_j \end{aligned} \tag{3. 3}$$

where $k_0+k_1+\ldots+k_t = N$, $k_0+k_1+\ldots+k_j = N_j$, $0 \leq y_n \leq N$, $N_j \leq x_n \leq N_j+k_{j+1}$, $0 \leq j \leq t-1$, $k_0 = 0$.

3.2 Description of the Cryptosystem

The cryptosystem is a block cipher, which is based on 16-bit block sizes of plaintext and ciphertext. The preliminary process is to partition plaintext into fixed-size blocks. Plaintext and ciphertext sequences P_i, C_i, $1 \leq i \leq n$, are given as

$$\begin{aligned} \text{Plaintext} &: \ P_1P_2\cdots P_n, \\ \text{Ciphertext} &: \ C_1C_2\cdots C_n. \end{aligned}$$

The secret key of proposed cryptosystem is a collection of several parameters, used in different parts of the algorithm. The key parameters are (r, m, t, C_0, K_s, K_c). This collection is defined as the master key, in [11]. The parameters of master key are; number of rounds for encryption and decryption processes r, circular shift amount m, iteration number of chaotic maps t, the initial ciphertext block C_0, which is used for the initial value, initial value of subkey K_s, and collection of TDCM parameters K_c. Figure 3. 1 illustrates parameters of master key clearly.

The subkey, K_i, is used in encryption of each plaintext block P_i. And the initial subkey is

$$K_0 = K_s. \tag{3. 4}$$

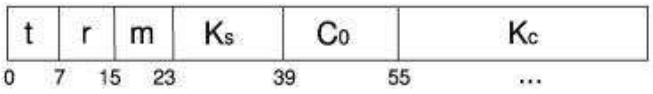

Figure 3. 1 The master key of encryption scheme, taken from [11]

The preliminary work for encryption of each plaintext block P_i is to update the subkey. Update process for K_i is given as

$$K_i = \begin{cases} K_{i-1} \oplus C_{i-1} & \text{if} \quad C_{i-1} \neq K_{i-1}, \\ K_{i-1} & \text{if} \quad C_{i-1} = K_{i-1}. \end{cases} \tag{3. 5}$$

i^{th} plaintext block is encrypted subsequently as

$$C_i = E(K_i, P_i), \tag{3. 6}$$

where function E is the encryption function and involves the following round operations.

$$\begin{aligned} v_0 &= P_i, \\ v_j &= \sigma(v_{j-1} \oplus \text{ROL}(K_i, jm)),\ 1 \leq j \leq r, \\ C_i &= v_r. \end{aligned} \tag{3. 7}$$

v_j denotes the output of round j. The output of last round is the ciphertext block C_i, of the corresponding plaintext block P_i. $\text{ROL}(\cdot, jm)$ denotes the rotate left operation (circular left shift) by jm bits. As the round number increases, the rotation amount increases also. The amount of rotation is derived using number of rounds r. The relation is given as

$$m = \begin{cases} \lfloor 16/r \rfloor & r \leq 16, \\ 1 & \text{else}. \end{cases} \tag{3. 8}$$

The round function σ is a collection of several functions and given as

$$\sigma = w \circ z^{-1} \circ \text{TDCM}^t_{K_c} \circ z \circ S. \tag{3. 9}$$

In Eq.(3. 9), S represents the substitution box (S-box). For a good confusion, encryption algorithm involves S-box, which is designed using iterations of chaotic functions. The mapping process is invertible and works with 16-bit quantities. S-box is not key dependent, so its value is not secret, and does not change for an algorithm. The S-box is designed to have desirable nonlinear properties. [17] gives examples of S-boxes, shown in Figure 3. 2 and Figure 3. 3.

z maps 16-bit quantities to 2D vectors of integers, and is also an invertible

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	60	c4	56	52	88	17	82	ac	28	96	4f	4a	ff	20	b5	6a
1	92	83	bc	a7	b2	9a	ee	70	35	e1	25	61	9d	a4	9c	47
2	b7	7d	2f	24	c7	7e	c5	c8	77	14	8d	cc	fd	8a	ef	36
3	76	2c	12	11	2a	29	a8	b8	22	84	c3	e9	e6	e2	15	57
4	e0	3c	69	ce	05	d4	cd	fa	30	f8	dd	75	cf	a0	0c	55
5	9f	41	f3	6f	ea	d2	a2	65	23	89	81	39	e4	93	ba	6b
6	a9	b0	1f	f7	34	43	1b	08	04	fc	0b	aa	73	94	eb	8e
7	c2	d6	53	48	18	27	8f	5b	5d	d0	ec	f4	f5	31	4b	ab
8	4e	97	79	bb	13	b6	5e	8b	10	50	49	1d	f6	99	00	68
9	3f	95	ad	e7	e8	87	8c	51	64	1e	d9	e5	5a	da	de	f0
a	0f	46	f1	1c	71	e3	09	a5	dc	9e	bf	40	80	3b	45	02
b	a6	42	d1	ed	d7	fe	16	9b	63	72	c0	78	b4	67	26	03
c	01	54	07	90	38	21	62	3d	d8	ca	7f	b1	0a	d5	44	a1
d	0d	c9	f2	2e	b9	59	6c	66	b3	74	32	bd	df	58	6d	37
e	3a	2d	db	6e	f9	1a	c6	06	5f	a3	2b	19	7c	fb	7b	af
f	be	0e	85	5c	33	7a	c1	4d	cb	86	91	4c	d3	ae	3e	98

Figure 3. 2 The S-box for encryption, taken from [11]

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	8e	c0	af	bf	68	44	e7	c2	67	a6	cc	6a	4e	d0	f1	a0
1	88	33	32	84	29	3e	b6	05	74	eb	e5	66	a3	8b	99	62
2	0d	c5	38	58	23	1a	be	75	08	35	34	ea	31	e1	d3	22
3	48	7d	da	f4	64	18	2f	df	c4	5b	e0	ad	41	c7	fe	90
4	ab	51	b1	65	ce	ae	a1	1f	73	8a	0b	7e	fb	f7	80	0a
5	89	97	03	72	c1	4f	02	3f	dd	d5	9c	77	f3	78	86	e8
6	00	1b	c6	b8	98	57	d7	bd	8f	42	0f	5f	d6	de	e3	53
7	17	a4	b9	6c	d9	4b	30	28	bb	82	f5	ee	ec	21	25	ca
8	ac	5a	06	11	39	f2	f9	95	04	59	2d	87	96	2a	6f	76
9	c3	fa	10	5d	6d	91	09	81	ff	8d	15	b7	1e	1c	a9	50
a	4d	cf	56	e9	1d	a7	b0	13	36	60	6b	7f	07	92	fd	ef
b	61	cb	14	d8	bc	0e	85	20	37	d4	5e	83	12	db	f0	aa
c	ba	f6	70	3a	01	26	e6	24	27	d1	c9	f8	2b	46	43	4c
d	79	b2	55	fc	45	cd	71	b4	c8	9a	9d	e2	a8	4a	9e	dc
e	40	19	3d	a5	5c	9b	3c	93	94	3b	54	6e	7a	b3	16	2e
f	9f	a2	d2	52	7b	7c	8c	63	49	e4	47	ed	69	2c	b5	0c

Figure 3. 3 The inverse S-box for decryption, taken from [11]

mapping. First 16-bit plaintext block is split into two bytes, and each byte is used as one of the integer coordinates in 2D discrete state space. For example, $z(\texttt{0x1A27}) = [26,\ 39]$ because $26 = (\texttt{1A})_{16}$ and $39 = (\texttt{27})_{16}$.

$\text{TDCM}^t_{K_c}$ denotes the TDCM process. Chaotic map is iterated t-times, and chaotic system parameters are denoted by K_c. The choice of chaotic map depends on algorithm design, and is a part of the algorithm. The proposed cryptosystem [11] considers The Standard Map, Generalized Cat Map, and Generalized Baker's Map. In order to have an invertible encryption operation chaotic maps must be

one-to-one and onto (bijective). Next mapping function z^{-1} takes the output of TDCM as input and maps the final 2D state of TDCM to a 16-bit integer.

At the end of each round, the bytes are swapped, and the process is denoted by w in Eq.(3. 9).

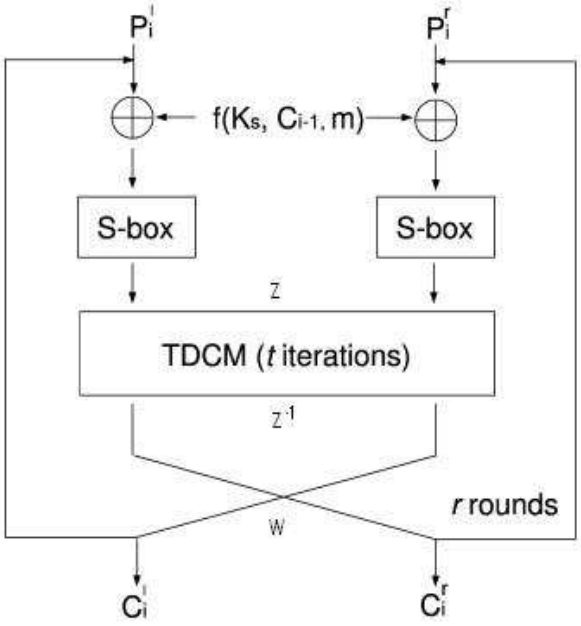

Figure 3. 4 The round operations of encryption scheme

All the round operations are illustrated in Figure 3. 4. Here the function $f(K_s, C_{i-1}, m)$ includes the shift and update operation, which is expressed in Eq.(3. 5).

After encryption of plaintext block P_i, the block key is updated again for next plaintext block P_{i+1}. The update process is given as

$$K_i \leftarrow \mathrm{ROL}(K_i, rm). \tag{3. 10}$$

Total shift amount in K_i is rm. Since K_i has 16-bits, the effective amount of rotation on K_i is $rm \mod 16$ in encryption of one block. Same round processes continues until all plaintext blocks are converted into ciphertext blocks.

The decryption process is the symmetric inverse of encryption process. The round operations, subkey update and swap operations are in reverse order. Moreover, inverse substitution and inverse chaotic map are used. The rotate left process (ROL) is replaced with rotate right process (ROR) and the σ function takes the mapping operations with reverse order. Also, S-box and TDCM are the inverse of the original ones.

3.3 Key Space Weakness

The master key of the cryptosystem, which is described in previous section, involves several secret parameters, r, m, t, C_0, K_s and K_c. All of the parameters, except TDCM parameter K_c, are formed in fixed length. r, m and t have 8 bits, C_0 and K_s have 16 bits. The length of K_c depends on the chosen TDCM. For example, Standard Map uses a single parameter, which is represented with 16 bits. In the case of using Standard Map, master key has 72 bits. Revealing the secret keys using a brute force attack requires 2^{71} trails on average, which takes a huge amount of time by the speed of today's technology. On the other hand, the baker map is designed using a variable length key, which makes harder to apply a brute force attack.

Effective key is the portion of secret key which could not be derived using any other portion, or a part of the algorithm. Algebraic dependencies in a system, among the key parameters, reduce the effective key size. The proposed cryptosystem has a dependent portion and a redundancy in the master key, and unfortunately, the effective key is only a subset of the master key.

If an attacker knows round number r, it is very easy to compute m using Eq.(3. 8), since the algorithm is public. Due to this fact, one cannot choose the shift amount m freely. Hence, the effective key does not include m,it should be treated as an internal parameter derived using one of the secret parameters.

During preliminary processes for encryption of the first plaintext block P_1, Eq.(3. 5) also causes an effective reduction in master key. If an attacker knows $K_1 = K_s \oplus C_0$, there is no need to have any knowledge about either K_s or C_0, since the encryption algorithm does not use the initial value of neither K_s nor C_0 in the sequel. Indeed, $K_s \oplus C_0$ becomes a secret parameter, so we cannot treat either of K_s and C_0 as a secret parameter. We will assume without loss of generality that $C_0 = \texttt{0x0000}$, in the succeeding sections.

Indeed, secret parameters of effective key are, r, t, $C_0 \oplus K_s$ and K_c. One may think that the collection of TDCM parameters makes the key size long enough against a brute force attack in spite of the weaknesses. However, there are several attacks to reveal the TDCM parameters. The attacks does not depend on which type of TDCM is used and how long the parameter K_c is. The attacks are described in subsequent sections in detail.

3.4 Chosen Ciphertext Attack on K_s

In this section, I will describe how an attacker can find the initial subkey K_s without a knowledge about rest of the key. The attack differs with choice of round number r. Hereafter, we assume that the attacker knows the parameter r. The

assumption is not restrictive since r is not too long to be revealed. The attacker should try only 255 different nonzero values as a brute force attack on parameter r, if he has efficient attacks to reveal the remaining secret parameters. Hence, in the subsequent parts of this chapter, r will be considered to be known by the attacker.

The attacks we developed to reveal K_s and the rest has no high computational requirements. The operations are so simple that the attacks are very efficient in time. Moreover, they have little memory requirement.

There are two cases to be considered when attacking on K_s. The first case is $rm \equiv 0 \bmod 16$ and the second one is $rm \not\equiv 0 \bmod 16$. Our chosen ciphertext attack differs for each of cases. The attacks are described in this section.

Note that the attacker does not know TDCM parameters, K_c and t, and just so he has no knowledge about inside of function E. He only knows the round number r.

3.4.1 $rm \equiv 0 \bmod 16$

Assume that the attacker chooses two ciphertext blocks as

$$C_1 = C_2 = j. \tag{3. 11}$$

Using Eq.(3. 4) and Eq.(3. 6), the encryption of first block is

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

Since $rm \equiv 0 \bmod 16$, using Eq.(3. 10) the subkey of first block K_1 becomes

$$\mathrm{ROL}(K_s, rm) = K_s.$$

In the case when $j = K_s$, using Eq.(3. 5) and Eq.(3. 6) the subkey for second block K_2 and just so the encryption becomes

$$K_2 = K_1 = K_s \;\Rightarrow\; j = E(K_s, P_2).$$

Indeed, the encryption of two block plaintext becomes

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

Since E is bijective, we have $P_1 = P_2$ for fixed K_s.

However, in the case when $j \neq K_s$, the second subkey, after updating process, becomes

$$K_2 = K_1 \oplus C_1 = K_s \oplus j$$

so, 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 first case, in which $P_1 = P_2$, yields a good result to decide on if $K_s = j$.

The progress of chosen ciphertext attack on K_s proceeds as follows. After choosing two ciphertext blocks C_1C_2 as in Eq.(3. 11), the attacker requests corresponding plaintext blocks. By comparing plaintext blocks, the attacker decides whether j is a candidate for K_s. If the plaintext blocks are equal, j is a candidate. By trying all the possibilities of two blocks of ciphertext and recording the results, the attacker may have one or more candidates for K_s. There are $2^{16} - 1$ different values for j. Hence, the attacker makes $2^{16} - 1$ trials in total.

The reason why the attacker may have more than one candidates even if $j \neq K_s$ is that we might have $E(K_1, P) = E(K_2, P)$ for some $K_1 \neq K_2$, and P. We have another chosen ciphertext attack to eliminate false candidates. The attacker needs to apply the following tests during elimination.

The attacker chooses two of the candidates, j_1 and j_2, and the corresponding plaintext blocks P_1 and P_2 from the previous attack, which satisfy

$$j_1 = E(K_s, P_1),\ j_2 = E(K_s, P_2). \tag{3. 12}$$

The elimination process proceeds by choosing new ciphertext blocks $\overline{C}_1$ and $\overline{C}_2$ as $\overline{C}_1 = j_1$ and $\overline{C}_2 = j_2$. The attacker obtains the corresponding plaintext blocks $\overline{P}_1$ and $\overline{P}_2$. The attack helps the attacker only to decide whether j_1 is the subkey or not. Let's see how $\overline{P}_1$ and $\overline{P}_2$ differ for each cases, which are $j_1 = K_s$ and $j_1 \neq K_s$.

When $j_1 = K_s$, using Eq.(3. 4) and Eq.(3. 6), the encryption of first block becomes

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

After key update process shown in Eq.(3. 5) and Eq.(3. 6), second subkey and encryption of second block becomes

$$K_2 = K_1 = K_s \ \Rightarrow\ j_2 = E(K_s, \overline{P}_2).$$

Indeed, when $j_1 = K_s$, we have

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

Comparing this with Eq.(3. 12), we obtain $\overline{P}_1 = P_1$ and $\overline{P}_2 = P_2$.

On the other hand, in the case $j_1 \neq K_s$ we find the second subkey as

$$K_2 = K_1 \oplus C_1 = K_s \oplus j_1.$$

so, we have

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

Comparing this with Eq.(3. 12), we conclude $\overline{P}_1 = P_1$ and $\overline{P}_2$ is a random 16-bit integer.

Only P_2 varies in two cases, so decision of the attacker has to depend on this difference. The attacker decides $j_1 \neq K_s$ only when $\overline{P}_2 \neq P_2$. Since the first case $\overline{P}_2 = P_2$ may occur even when $j_1 \neq K_s$, he cannot be sure whether $j_1 = K_s$. Indeed, the test can be concluded as $j_1 \neq K_s$ only when $\overline{P}_2 \neq P_2$.

The attacker now have a method to eliminate the false candidates for the subkey. By continuing in this fashion, the attacker will have only one candidate for K_s, and the attack will terminate successfully.

3.4.2 $rm \not\equiv 0 \bmod 16$

In this case, the attacker chooses ciphertext blocks as

$$C_1 = C_2 = \cdots = C_{k-1} = 0,\ C_k = j,\ C_{k+1} = 0, \tag{3. 13}$$

where $k = \frac{\mathrm{lcm}(16,u)}{u}$ and $u = rm \pmod{16}$. The attacker chooses k such that the subkey repeats itself at $(k+1)^{th}$ block, that is $K_1 = K_{k+1}$. Let's see how the subkeys are updated for the chosen ciphertexts in Eq.(3. 13) using Eq.(3. 4), Eq.(3. 5) and Eq.(3. 10).

$$K_1 = K_s,\ K_2 = \mathrm{ROL}(K_s, u),\ \ldots\ ,\ K_k = \mathrm{ROL}(K_s, (k-1)u)$$

and, last key depends on j, such that

$$K_{k+1} = \begin{cases} j \oplus K_s & \text{if} \quad j \neq K_s, \\ K_s & \text{if} \quad j = K_s. \end{cases}$$

Hence, the attacker obtains

$$
\begin{aligned}
0 &= E(K_s, P_1), \\
0 &= E(\mathrm{ROL}(K_s, u), P_2), \\
0 &= E(\mathrm{ROL}(K_s, 2u), P_3), \\
&\vdots \\
0 &= E(\mathrm{ROL}(K_s, (k-2)u), P_{k-1}), \\
j &= E(\mathrm{ROL}(K_s, (k-1)u), P_k),
\end{aligned}
\tag{3. 14}
$$

and

$$
0 = \begin{cases} E(j \oplus K_s, P_{k+1}) & \text{if} \quad j \neq K_s, \\ E(K_s, P_{k+1}) & \text{if} \quad j = K_s. \end{cases}
\tag{3. 15}
$$

Using Eq.(3. 14) and Eq.(3. 15), if he finds that $P_1 = P_{k+1}$, then $j = K_s$. While applying a chosen ciphertext attack the attacker ought to use this fact.

For all possible 16-bit nonzero numbers j, the attacker applies a chosen ciphertext attack by choosing the ciphertext sequence in Eq.(3. 13). After analyzing the corresponding plaintext blocks $P_1, \cdots, P_{k+1}$ as in Eq.(3. 14) and Eq.(3. 15), the attacker decides whether j is a candidate for the subkey K_s. During this analysis the attacker records the candidates. The attacker tries $2^{16} - 1$ different values for j.

Even if $j \neq K_s$, the equality $P_1 = P_{k+1}$ may occur with a very low probability. Hence, the task should be elimination of the false keys as in first case. To eliminate such a false key j, the attacker chooses the ciphertext sequence $\overline{C}_1 = \mathrm{ROL}(j, u)$, $\overline{C}_2 = 0$ and obtains the corresponding plaintext sequence $\overline{P}_1\overline{P}_2$. Using Eq.(3. 4) and Eq.(3. 6), encryption process of first block becomes

$$
\mathrm{ROL}(j, u) = E(K_s, \overline{P}_1).
$$

Due to the process in Eq.(3. 10), at the end of the encryption of first block, K_1 becomes $\mathrm{ROL}(K_s, u)$ because the total amount of rotation process is $u = rm \pmod{16}$. Hence, using Eq.(3. 5), the attacker obtains

$$
0 = \begin{cases} E(\mathrm{ROL}(j, u) \oplus \mathrm{ROL}(K_s, u), \overline{P}_2) & \text{if} \quad j \neq K_s, \\ E(\mathrm{ROL}(K_s, u), \overline{P}_2) & \text{if} \quad j = K_s. \end{cases}
\tag{3. 16}
$$

The attacker eliminates a false key using Eqns.(3. 14, 3. 16). If $P_2 \neq \overline{P}_2$, the attacker eliminates a false key j. However, if $j = K_s$, $P_2 = \overline{P}_2$ the test has no conclusion. After repeating this process until one candidate remains, the attacker will obtain the correct key.

As a result, the attacker can reveal a portion of the master key, K_s, either $rm \equiv 0 \bmod 16$ or $rm \not\equiv 0 \bmod 16$. The attack slightly differs for either cases.

3.5 Attacking the Function E

As we saw, revealing K_s does not depend on any other secret parameters. The only parameter to be known is round number r. After revealing the subkey K_s, the next task, for the attacker, is to reveal the remaining secret parameters. Now, let's see how the attacker proceeds revealing the other secret parameters t and K_c. If an attacker has the subkey K_i and round number r, it will be enough to break the whole algorithm by revealing the parameters t and K_c. Those parameters specify the function E because the parameters are the iteration number and the secret parameter of TDCM. Other operations, S, z, z^{-1} and w, are public and do not depend on any secret parameter.

One may think that by the increase of weakness in the key, it would be possible to apply a brute force attack for the remaining secret parameters. To apply a brute force attack, the attacker needs to try all the possible values of t and K_c using a known plaintext-ciphertext pair. For instance, by assuming that K_c has 32 bits, the attacker has to try 2^{39} different value for a brute force attack. However, we have a general attack, which requires 2^{16} chosen ciphertext/plaintext blocks, to reveal rest of the key. Moreover our attacks have no high computational requirements. Furthermore, our attacks do not become more complex with an increase in the key size because they don't depend on the choice of TDCM.

3.5.1 Sampling E

Here we will show that an attacker can obtain the ciphertext block C, encrypted using a subkey K_i and plaintext block P of his choice. Similarly, he can obtain the plaintext block P, decrypted using a subkey K_i and ciphertext block C of his choice. Indeed, the attacker is able to make a user to encrypt or decrypt a chosen plaintext or ciphertext block with the subkey K_i of his choice. The only thing for the attacker is to choose a subkey K_i and one of C or P in the relation

$$C = E(K_i, P), \tag{3. 17}$$

and obtain the other. Note that the attacker already knows round number r, shift amount m and the initial subkey K_s by applying the attacks described in preceding sections. To see how the attacker can do this, the attacker applies the following operations. First he chooses two plaintext blocks. Let's write the encryption equations for a sequence of two plaintext blocks as

$$C_1 = E(K_s, P_1),$$
$$C_2 = E(\mathrm{ROL}(K_s, rm) \oplus C_1, P_2). \quad (3.\ 18)$$

Note that the proposed cryptosystem especially requires that $\mathrm{ROL}(K_s, rm) \oplus C_1 \neq 0$ for subkey to be updated.

By choosing $C_1 = K_i \oplus \mathrm{ROL}(K_s, rm)$, since $\mathrm{ROL}(K_s, rm) \oplus C_1 = K_i$, the second line in Eq.(3. 18) becomes

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

By using a two block decryption, we can obtain the desired plaintext block P, which is decrypted via a chosen subkey K_i and plaintext block P. The second block is the desired plaintext block. Therefore, in order to obtain the desired ciphertext block, that is encrypted with an attacker defined subkey K_i, the attacker ought to choose the ciphertext sequence C_1C, in which $C_1 = K_i \oplus \mathrm{ROL}(K_s, rm)$, and obtains the corresponding plaintext sequence P_1P_2. Let's see the process more clearly as

$$C_2 = E(\mathrm{ROL}(K_s, rm) \oplus K_i \oplus \mathrm{ROL}(K_s, rm), P_2) = E(K_i, P_2).$$

Similarly, If he wants to know C for a particular P in Eq.(3. 17), he chooses the plaintext sequence P_1P and obtains C_1C_2. The desired ciphertext block C is the second one, C_2.

As a result of the attack described above, an attacker is able to sample the function E using his choices of K_i and one of C and P, at any desired point (P, C) of his choice.

Revealing the remaining secret parameters is equivalent to revealing the function σ in Eq.(3. 7). As stated above the mappings w, z, S are fixed and the attacker already knows r, m, and K_i. An attacker can reveal the remaining parameters, t and K_c if he has obtained r, m, and K_i. To achieve this, the attacker has to use the attacks, described below, in a combination. Although the attacks work in a combination, it sometimes is not needed to use all of the attacks together.

3.5.2 Permutation Orbit Attack

The function σ is composed of several invertible maps, so σ is also an invertible map. The mapping is done from the set $\{0, 1, \ldots, 2^{16} - 1\}$ onto itself. Indeed, we can say that σ is a permutation on the set $\{0, 1, \ldots, 2^{16} - 1\}$. Permutation orbit attack lets the attacker reveal some portions of permutation σ using some particular

choices of K_i.

The attacker chooses K_i such that the rotation process does not cause a change in K_i. We can specify such subkeys as

$$\mathrm{ROL}(K_i, m) = K_i. \tag{3. 19}$$

Using such subkeys in Eq.(3. 7) makes the round process

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

Now, Let's define a new permutation π, which is a modification of the permutation σ, as

$$\pi(x) = \sigma(x \oplus K_i). \tag{3. 20}$$

Since Eq.(3. 10) has no effect in encryption process for the subkeys in Eq.(3. 19), for $x \in \{0, 1, \ldots, 2^{16} - 1\}$, we can express the encryption process as

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

By revealing a point in π such that $Y = \pi(P)$, an attacker also reveals a point in σ such that $Y = \sigma(P \oplus K_i)$.

When choosing a subkey K_i, which turns the function E into the $r-$power of a permutation, we should consider the shift amount m. For example, let us take $m = 2$. The subkeys, that satisfy Eq.(3. 19), are `0101010101010101` (`0x5555`), `1010101010101010` (`0xAAAA`) and `1111111111111111` (`0xFFFF`). If $m = 1$, there would be only one invariant subkey, that is (`0xFFFF`).

The attacker obtains different permutations π for each value of K_i that satisfies Eq.(3. 19). We can show why they are different permutations clearly. Let's assume there are two different keys, K_1 and K_2, which turns the function E into the $r-$power of a permutation. Note that if $K_1 \neq K_2$, $K_1 \oplus x \neq K_2 \oplus x$. Also, let's assume the permutations π_1^r and π_2^r are the power permutations of E depending on K_1 and K_2 respectively. If $\pi_1(x) = Y$ and $\pi_2(x) = Z$, then $Y \neq Z$. We can see this easily as $\pi_1(x) = \sigma(x \oplus K_1) = Y$ and $\pi_2(x) = \sigma(x \oplus K_2) = Z$. Note that $K_1 \oplus x \neq K_2 \oplus x$. Permutations are one-to one and onto, so two different value cannot map to the same value. Hence, we prove that the permutations are different.

An attacker can reveal the whole permutation π^r. To achieve this, the attacker needs totally 2^{16} sampling processes, given in Section 3.5.1. He should obtain $\pi^r(P)$ for every P in $\{0, 1, \ldots, 2^{16} - 1\}$, to reveal π^r.

Permutation π^r does not help the attacker to reveal σ. He needs to gather information about the permutation π for revealing σ using Eq.(3. 20). Using

permutation orbit attack he can obtain some information about π.

Definition 3.5.1 *[18] 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. In other words, an orbit is a subset of a permutation, which each value maps the next value circularly.*

Theorem 3.5.2 *[18] A permutation defined on a finite set partitions the set into disjoint ordered-orbits.*

I refer to an ordered orbit simply as an orbit hereafter. To find the orbits of a permutation π defined on a set A, we start from any element $a_0 \in A$ and form the orbit elements as $(a_0, \pi(a_0), \pi^2(a_0), \ldots, \pi^{n-1}(a_0))$ until $\pi^n(a_0) = a_0$. Here we found an orbit with length n. To find another orbit we then select an unvisited element $b_0 \in A$ and form the orbit just like the first orbit. By continuing in this fashion we form all the orbits until there is no unvisited element in set A.

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

$$\pi^i(a_0) = \pi^{i \bmod n}(a_0) \tag{3. 21}$$

because the orbits are circular. After iterating the permutation π n times, we will see the starting value as last mapped value. Hence the $(n+1)^{th}$ iteration is equal to 1^{st} iteration.

Example 3.5.3 *Let an orbit, α, of the permutation π has length $n = 5$ and is given as $\alpha = \{123, 323, 21, 223, 23\}$ and let $i = 7$.*

From the definition of the orbit we know $\pi(123) = 323$, $\pi(323) = 21$, $\pi(21) = 223$, $\pi(223) = 23$ and $\pi(23) = 123$.

Now, we can calculate $\pi^7(123) = \pi \circ \pi \circ \pi \circ \pi \circ \pi \circ \pi \circ \pi(123) = 21$.

Also, the result of $\pi^{7 \bmod 5} = \pi^2 = \pi \circ \pi(123) = 21$ which is equal to the result of $\pi^7(123)$.

Lemma 3.5.4 *Let $\alpha = (a_0, a_1, \ldots, a_{n-1})$ be an orbit of length n in the permutation π, where $\gcd(n, r) = d$. Then, α is split into d equal length orbits in π^r. Moreover, the elements $a_0, a_1, \ldots, a_{d-1}$ are in different orbits in π^r.*

Proof 3.5.5 *The first thing to be proven is that any orbit in π^r that contains a_j, $0 \leq j < n$, has length n/d. Using the fact that $\mathrm{lcm}(n, r) \equiv 0 \pmod{n}$ and Eq.(3. 21) we have*

$$(\pi^r)^{n/d}(a_j) = \pi^{rn/d}(a_j) = \pi^{\mathrm{lcm}(n,r)}(a_j) = a_j.$$

Thus, maximum length of an orbit in π^r containing a_j is n/d. Moreover, n/d is also the minimum length. Let's show this fact via a contradiction. Assume that there is an integer $i < \frac{n}{d}$ such that $ri \equiv 0 \pmod{n}$. This shows that ri is divisible by n. Also, we know that ri is divisible by r. Hence, ri is a common multiple of n and r. Furthermore, we have $ri < \operatorname{lcm}(n,r)$ and $\operatorname{lcm}(n,r) = rn/d$. Since any common multiplier cannot be smaller than the least one, this is a contradiction. Therefore, any orbit in π^r that contains a_j has length n/d.

Second, we show that the elements $a_0, a_1, \ldots, a_{d-1}$ are in different orbits in π^r. To prove this fact let us again use a contradiction. Assume that there exists integers i, j, $0 \leq i < j < d$, such that $\pi^{rk}(a_i) = a_j$, for some integer k. Using the definition of orbits we can also say that $a_j = \pi^{j-i}(a_i)$. Using this fact and Eq.(3. 21) we have $rk \equiv j - i \pmod{n}$. Now, as you see $rk = j - i + xn$ where x is an integer. Consequently, from the last equality we see that $j - i$ must be divisible by d, because xn and rk are also divisible by d. However, this is a contradiction since $0 \leq i < j < d$. As a result we see that $a_0, a_1, \ldots, a_{d-1}$ are in different orbits in π^r.

Thus, α is split in π^r into d non-overlapping orbits of length n/d each.

To see the Lemma 3.5.4 clearly, let's give an example.

Example 3.5.6 *Let an orbit, α, of the permutation π has length $n = 8$ and is given as $\alpha = \{123, 323, 21, 223, 23, 12, 345, 4\}$ and let $r = 6$.*

Since $\gcd(8, 6) = 2$ the orbit α should be split into two new orbits in π^6.

Repeating the same process in Example 3.5.3 for each element in α yields, $\pi^6(123) = 345$, $\pi^6(323) = 4$, $\pi^6(21) = 123$, $\pi^6(223) = 323$, $\pi^6(23) = 21$, $\pi^6(12) = 223$, $\pi^6(345) = 23$, $\pi^6(4) = 12$.

Hence, from the mappings found, orbits of π^6 which come from the orbit, α, of permutation π are, $\alpha_1 = \{123, 345, 23, 21\}$, $\alpha_2 = \{323, 4, 12, 223\}$.

Moreover, you see that 123 and 323 are in different orbits which shows the second fact of Lemma 3.5.4.

When the process in Example 3.5.6 is applied to all the orbits of π, resulting orbits will be the orbits of π^r.

Lemma 3.5.7 *Let $\beta = (b_0, b_1, \ldots, b_{n-1})$ be an orbit in the permutation π^r, and it is the only orbit with length n. Then,*

$$\pi(b_j) = b_{j+r^* \pmod{n}}, \ 0 \leq j < n,$$

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

In other words, the orbit β is not split in π^r, it is only shuffled during iterations of rounds. And the shuffling formula is the given equality.

Proof 3.5.8 *Let's first show that* $\gcd(n, r) = 1$ *using a contradiction. Assume that* $\gcd(n, r) = d$ *and* $d > 1$. *As stated in Lemma 3.5.4, any orbit of length* n *in* π *is split into* d *orbits in* π^r. *Thus, only when there is an orbit of length* nd *in* π, *there would be an orbit of length* n *in* π^r. *However, we would have* d *orbits of length* n *in* π^r *in such a case. This is a contradiction since we assumed that there is only one such orbit in* π^r. *Therefore,* $\gcd(n, r) = 1$.

Now, let's show that α, *which is an orbit in* π *and contains the element* b_0, *contains all elements of* β. *As we know from the definition of an orbit,* α *contains powers* $\pi^i(b_0)$ *for all integers* i. *Hence,* $\pi^{rj}(b_0)$ *is also an element of* α *for* $1 \leq j < n$. *Moreover,* $\pi^{rj}(b_0) = b_j$, $0 \leq j < n$. *Therefore, this is the proof which shows that* α *contains all elements of* β.

Let's similarly show that β *contains all the elements of* α. *Let* $a_j = \pi^j(b_0)$ *be an element in* α, *and let* $k \equiv jr^* \pmod{n}$, *where* $rr^* \equiv 1 \pmod{n}$. *Certainly there exists such an* r^* *since* $\gcd(n, r) = 1$. *By the construction of* β *in* π^r, $(\pi^r)^k(b_0)$ *is in* β. *Substituting* k, *we have*

$$(\pi^r)^k(b_0) = \pi^{rk}(b_0) = \pi^{rjr^*}(b_0) = \pi^j(b_0) = a_j.$$

Therefore, β *contains all the elements of* α *and during the conversion from the orbit* β *in* π^r *to the orbit* α *in* π *the elements are just shuffled, and we finally derive the formula for the shuffle as*

$$b_i = (\pi^r)^{i-j}(b_j),\ 0 \leq i, j < n. \tag{3. 22}$$

Choosing $i = j + r^* \pmod{n}$, *and substituting in Eq.(3. 22), we obtain*

$$b_{j+r^* \pmod{n}} = \pi(b_j),\ 0 \leq j < n.$$

Let's give an example to show Lemma 3.5.7 more clearly.

Example 3.5.9 *Let an orbit,* α, *of the permutation* π *has length* $n = 8$ *and is given as* $\alpha = \{123, 323, 21, 223, 23, 12, 345, 4\}$ *and let* $r = 5$.

Since $\gcd(8, 5) = 1$ *the orbit* α *should not be split into smaller orbits in* π^5.

Repeating the same process in Example 3.5.3 for each element in α *yields,* $\pi^5(123) = 12$, $\pi^5(323) = 345$, $\pi^5(21) = 4$, $\pi^5(223) = 123$, $\pi^5(23) = 323$, $\pi^5(12) = 21$, $\pi^5(345) = 223$, $\pi^5(4) = 23$.

Hence, from the mappings found, orbits β *of* π^5 *which come from the orbit,* α, *of permutation* π *is,* $\beta = \{123, 12, 21, 4, 23, 323, 345, 223\}$.

Now, let's construct back the orbit α *from the orbit* β *using the formula in Lemma 3.5.7.*

First we found that $r^* = 5 (\text{mod } 8)$. *Then,*
for $j = 0$, $b_0 = 123$, $\pi(b_0) = b_{0+r^*} = b_5 = 323$,
for $j = 1$, $b_1 = 12$, $\pi(b_1) = b_{1+r^*} = b_6 = 345$,
for $j = 2$, $b_2 = 21$, $\pi(b_2) = b_{2+r^*} = b_7 = 223$,
for $j = 3$, $b_3 = 4$, $\pi(b_3) = b_{3+r^*} = b_0 = 123$,
for $j = 4$, $b_4 = 23$, $\pi(b_4) = b_{4+r^*} = b_1 = 12$,
for $j = 5$, $b_5 = 323$, $\pi(b_5) = b_{5+r^*} = b_2 = 21$,
for $j = 6$, $b_6 = 345$, $\pi(b_6) = b_{6+r^*} = b_3 = 4$,
for $j = 7$, $b_7 = 223$, $\pi(b_7) = b_{7+r^*} = b_4 = 23$.
The reconstructed orbit $\alpha' = \{123, 323, 21, 223, 23, 12, 345, 4\}$ *which is equal to the orbit* α.

Lemma 3.5.10 *Let* $\beta = (b_0, b_1, \ldots, b_{n-1})$ *be one of the* q *orbits of length* n *in the permutation* π^r. *Let* d *be the least divisor of* r *larger than 1. Assume that* $q < d$. *Then,*

$$\pi(b_j) = b_{j+r^* \pmod{n}}, \ 0 \leq j < n, \tag{3. 23}$$

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

Proof 3.5.11 *As stated in Lemma 3.5.4, the orbits are split in* π^r *into* d *equal length orbits, where* $d = \gcd(n, r)$ *and* n *is the size of the original orbit in* π. *Thus, if an orbit is split in* π^r, *there should be at least* d *orbits of equal length in* π^r.

Let's show a contradiction to simply prove the thesis. Assume that $\gcd(n, r) = d$ *and* $d > q$. *As stated in Lemma 3.5.4, any orbit of length* n *in* π *is split into* d *orbits in* π^r. *Thus, only when there is an orbit of length* nd *in* π, *there would be an orbit of length* n *in* π^r. *However, we would have* d *orbits of length* n *in* π^r *in such a case. This is a contradiction since we assumed that there* q *orbits of length* n *in* π^r. *This fact shows that none of these orbits are split. Therefore,* $\gcd(n, r) = 1$.

As stated in Lemma 3.5.7, let $\beta = (b_0, b_1, \ldots, b_{n-1})$ *be one of the* q *orbits of length* n *in the permutation* π^r. *The orbit* α, *which is an orbit in* π *and contains the element* b_0, *contains all elements of* β. *And similarly,* β *contains all the elements of* α. *The elements are just shuffled during iterations of the permutation* π. *The shuffling formula is given in Eq.(3. 23).*

Using Lemma 3.5.7 and Lemma 3.5.10 we can reveal some portions of the permutation π using the appropriate orbits in π^r. However, there are some remaining orbits , which we cannot be used to reveal more data from.

Lemma 3.5.12 *Let $\beta^1 = (b_0^1, b_1^1, \ldots, b_{n-1}^1)$ and $\beta^2 = (b_0^2, b_1^2, \ldots, b_{n-1}^2)$ be two orbits of length n in π^r. If $\pi(b_i^1) = b_j^2$ for some i, j then*

$$\pi(b_{i+k \bmod n}^1) = b_{j+k \bmod n}^2,\ 1 \leq k < n.$$

Proof 3.5.13 *Assume that we somehow now that $\pi(b_i^1) = b_j^2$ for some i, j. We can express the relation between b_i^1 and b_{i+k}^1 as $\pi^{rk}(b_i^1) = b_{i+k}^1$. Operating by π on both sides, we obtain $\pi^{rk+1}(b_i^1) = \pi(b_{i+k}^1)$, or $\pi^{rk}(\pi(b_i^1)) = \pi(b_{i+k}^1)$. Using $\pi(b_i^1) = b_j^2$, we get $\pi^{rk}(b_j^2) = \pi(b_{i+k}^1)$. But the left hand side is just $b_{j+k \bmod n}^2$ and the result follows.*

Let's assume that the original orbit, which the orbits β^1 and β^2 comes from, is α. Even though we are sure that these two orbits are come from α, we cannot say that α is the composition of only these two since we cannot say that α is split into exactly two orbits. The orbit α might be split into several orbits more than two.

Using Lemma 3.5.12 we can recover some more data if we have some sample points from the orbits of unrecovered portion. Let's give an example for Lemma 3.5.12.

Example 3.5.14 *Let $\beta_1 = \{123, 345, 23, 21\}$ and $\beta_2 = \{323, 4, 12, 223\}$ are two of the remaining orbits in π^r and assume that we know that $\pi(345) = 223$.*

By using this sample point we can say that $\pi(23) = 323$, $\pi(21) = 4$ and $\pi(123) = 12$.

Or, let's assume these two orbits comes from the orbit α in π, so we can say that a portion of the orbit α is

$$\alpha = \{\ldots, 345, 223, \ldots, 23, 323, \ldots, 21, 4, \ldots, 123,\ 12, \ldots\}.$$

For second case please see Example 3.5.9.

After giving some necessary definitions, proofs and theorems, let's see how an attacker applies the permutation orbit attack. The attack proceeds as follows:

The attacker determines the appropriate subkeys K_i that satisfy Eq.(3. 19). Let's assume that there are k such subkeys. By using the sampling method shown in Eq.(3. 18) and those subkeys K_i^j, 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. After finding those permutations, the attacker starts to reveal some portions of the permutation π using the given lemmas. He performs the following steps for each K_i^j.

Step 1. Extract the power permutations π_j^r by applying the sampling method given in Eq.(3. 18) using all possible subkeys K_i^j shown in Eq.(3. 19).

Step 2. Compute all the orbits of the extracted power permutations using the method shown in Algorithm **??**.

Step 3. Select all lone orbits, which are stated in Lemma 3.5.7. Then, if there exist such an orbit of length n_1 in π_j^r, use Lemma 3.5.7 to reveal n_1 points in π_j. From those, reveal n_1 points of σ using Eq.(3. 20). Repeat the same process for all the lone orbits.

Step 4. Start to count the number of same sized orbits in π_j^r. If there is a collection of q orbits with same size, where q is less than the least divisor of r larger than 1, then use Lemma 3.5.10 to reveal qn_2 more points in π_j, where n_2 is the length of an orbit in the collection. Then convert the revealed values of π to the appropriate places in σ. Repeat all the processes to all the orbits.

Step 5. Do the above steps for each permutation π_j^r and reveal different portions of σ. Then, by Lemma 3.5.12, start to use the revealed points due to each power permutation π_j^r as a sample point for the other power permutations if eligible.

To understand Step 5 more clearly let's give more explanation. Let $R_j \subset \{0, 1, \ldots, 2^{16} - 1\}$ be the points for which $\sigma(R_j)$ is revealed using Steps 1, 2, 3 and 4 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 3 or Step 4 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 4 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 3.5.12 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 3.5.12 to reveal $n_3 - 1$ more points in π_j.

3.5.3 Expansion Attack

The permutation orbit attack, which we described in previous section, partially reveals the permutation σ. Here, we describe an attack, which we call as expansion attack, which works with a partially revealed permutation σ.

Let's denote the revealed portion of σ by R, and unrevealed portion of σ by U. Hence, R and U are two disjoint subsets of $\{0, 1, \ldots, 2^{16} - 1\}$ such that $R \cup U =$

$\{0, 1, \ldots, 2^{16} - 1\}$. Indeed, 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 $x \in U$.

By using the sampling method described in preceding sections, an attacker knows any triple (C, P, K_i) of his choice such that $C = E(K_i, P)$. Assume that the attacker has such a triple, where $C \in U$ i. e. he does not know the value which is mapped by σ to C. The attacker has P and K_i, so he can carry out the calculations in the steps of Eq.(3. 7) until he arrives an unrevealed value $v_j \oplus \mathrm{ROL}(K_i, (j+1)m) \in U$ where j is the round number. He starts out with $v_0 = P$. He can calculate v_1 if $v_0 \oplus \mathrm{ROL}(K_i, m) \in R$ since

$$v_1 = \sigma(v_0 \oplus \mathrm{ROL}(K_i, m)).$$

Once he knows v_1, he can calculate v_2 if $v_1 \oplus \mathrm{ROL}(K_i, 2m) \in R$ since

$$v_2 = \sigma(v_1 \oplus \mathrm{ROL}(K_i, 2m)).$$

Assume that he continues in this fashion, reaches the penultimate step and calculates v_{r-1}. Certainly, $v_{r-1} \oplus \mathrm{ROL}(K_i, rm) \notin R$ since otherwise we would have

$$C = v_r = \sigma(v_{r-1} \oplus \mathrm{ROL}(K_i, rm)) \in \sigma(R)$$

which contradicts the assumption. However, this is what we expect to finalize the expansion attack since the attacker has just revealed the value of the map σ at a new point $v_{r-1} \oplus \mathrm{ROL}(K_i, rm)$, and he already knows C. As a result, if the attacker reaches the last step without encountering an unrevealed value, he expands R by one point and shrinks U by one point. The new point is

$$\sigma(v_{r-1} \oplus \mathrm{ROL}(K_i, rm)) = C.$$

The attacker tries all possible such triples to reveal more and more points. Note that each revealed point is a sample point, and if the attacker uses this sample point as in Lemma 3.5.12. He may add a new orbit or two orbits into the revealed portion R by the way. Hence, at the end of each success of expansion attack, the attacker uses Lemma 3.5.12 to reveal the remaining points faster. This also increases the probability of the success in expansion attack since R is expanded.

3.5.4 Skipping Attack

The expansion attack attack described in previous section give successful results when an applicable portion of the permutation σ is not revealed. Unless there are enough points in revealed portion R, we may not reach the last round success-

fully. We can see from Eq.(3. 8) that when $r \geq 9$ we have $m = 1$. Thus, there is only one subkey satisfying Eq.(3. 19) when $m = 1$, so the attacker cannot use Lemma 3.5.10 in permutation orbit attack. Moreover, there may not be such many orbits satisfying Lemma 3.5.7, so he may reveal a very small portion of the permutation σ. Also when r is an even number, its smallest divisor is 2 and he can not use Lemma 3.5.10 again. This adversely affects the size of the revealed set R that can be used in the expansion attack.

Now let me explain our skipping 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 Eq.(3. 7).

Assume that a nonzero K_i satisfies

$$\mathrm{ROL}(K_i, 2) = K_i. \tag{3. 24}$$

which is slightly different from Eq.(3. 19).

Since the skipping attack skips the odd rounds, we now substitute the odd round outputs into even round expressions. Using Eq.(3. 24) with Eq.(3. 7) with those substitutions 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

$$\gamma(x) = \sigma(\sigma(x \oplus \mathrm{ROL}(K_i, 1)) \oplus K_i), \tag{3. 25}$$

we can express the relation between C and P as

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

Thew attacker applies the permutation orbit attack with $K_i =$`0xFFFF`. Then, he obtains the permutation π^r, and using Lemma 3.5.7 and Lemma 3.5.10, he reveals a portion of σ. Let R denote the revealed portion of the map σ.

The skipping attack proceeds as follows. The attacker does the same process he did in permutation orbit attack. He chooses every $P \in \{0, 1, \ldots, 2^{16} - 1\}$ and obtains their corresponding ciphertext block with $K_i^1 =$ `0x5555` and $K_i^2 =$ `0xAAAA`

satisfying Eq.(3. 24). Hence, 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 using permutation orbit attack.

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

$$y = \sigma(\sigma(x \oplus \mathrm{ROL}(K_i^j, 1)) \oplus K_i^j). \tag{3. 26}$$

There are two ways the attacker can use Eq.(3. 26) to reveal a new point on σ.

In the first case, 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 as

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

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 as

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

Thus, with the skipping attack, the attacker reveals some new points on the map σ. He subsequently uses Lemma 3.5.12 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.

3.6 Simulation Results

We give simulation results for three examples that illustrate our methods. Simulations are performed under MATLAB running on Mac OS X 10.5.4 with Intel Core 2 Duo 2.16 GHz processor and 2 GB RAM.

In the first simulation, we used the cryptosystem with secret parameters given in the example in [11]. We have $r = 8$, $m = 2$, $t = 12$, $C_0 =$ `0x4ED3`, $K_s =$ `0x8F4C`. Using the equivalence explained in Section 3.3, this is equivalent to $C_0 =$`0x0000`, $K_s =$`0xC19F` = `0x4ED3`$\oplus$`0x8F4C`. We used the standard map as TDCM. The secret TDCM parameter is $K_c = 53246$.

We first applied the chosen ciphertext attack on K_s given in Section 3.4. We found only one nonzero candidate for K_s. Hence we do not have false candidates for the subkey.

Next, we applied the permutation orbit attack with the block keys $K_i^1 =$`0xFFFF`, $K_i^2 =$ `0x5555` and $K_i^3 =$`0xAAAA` and obtained the orbit structures of π_1^8, π_2^8, π_3^8 for π_1, π_2, π_3 defined in Eq.(3. 20).

The orbit structure of π_1^8 with $K_i^1 =$`0xFFFF` is given as $(8, 6714)$, $(1, 3895)$,

$(8, 804)$, $(1, 699)$, $(1, 449)$, $(4, 63)$, $(1, 25)$, $(8, 9)$. Here a pair (q, n) means that there are q orbits of length n.

Similarly, the orbit structure of π_2^8 with $K_i^2 =$`0x5555` is given as $(1, 32489)$, $(2, 4729)$, $(4, 4535)$, $(1, 1805)$, $(1, 1455)$, $(1, 935)$, $(1, 153)$, $(8, 106)$, $(1, 57)$, $(2, 33)$, $(4, 21)$, $(1, 19)$, $(2, 13)$, $(1, 1)$.

Finally the orbit structure of π_3^8 with $K_i^3 =$ `0xAAAA` is given as $(1, 39285)$, $(1, 17431)$, $(1, 3115)$, $(2, 1805)$, $(1, 1307)$, $(1, 513)$, $(4, 23)$, $(8, 21)$, $(1, 11)$, $(4, 1)$.

We apply Lemma 3.5.7 to lone orbits of length 3895, 699, 449 and 25 in π_1^8 to reveal 5068 entries in σ. Applying the Lemma 3.5.7 to π_2^8 and π_3^8, we reveal 36914 and 61662 entries, respectively. These correspond to $|R_1| = 5068$, $|R_1| = 36916$, $|R_1| = 61662$. Some of these overlap. In total, with the orbit attack we reveal 63945 entries of σ. Hence, $|R_1 \cup R_2 \cup R_3| = 63945$. This corresponds to the 97.57% of the map σ.

From the revealed portion R of σ, we see that $0 \in R$ and that $\sigma(0) = 34114$ but $0 \notin R_1$. Searching for `0x0000`$\oplus$`0xFFFF` in the orbits of π_1^8, we see that it is mapped across two orbits of length 6714 in π_1^8. Using this knowledge with Lemma 3.5.12, we reveal 6714 new points in σ. Continuing in this fashion, we reveal all of σ in 15 applications of Lemma 3.5.12. In total, the attack takes less than a minute.

In the second example, we illustrate the combination of permutation orbit and expansion attacks for the case when $r = 5$ and $m = 3$. The rest of the parameters are the same.

We first apply the attack on K_s for the case $rm \not\equiv 0 \bmod 16$ given in Section 3.4.2. In this case, we have $u = 15$, so K_i is rotated left by 15 bits after the encryption of every block of plaintext. We found two nonzero candidates for K_s; `0xC19F` and `0xCFE1`. Using the elimination method given at the end of Section 3.4.2, we arrived at the correct subkey $K_s =$`0xC19F`.

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 $(1, 53712)$, $(1, 6432)$, $(5, 779)$, $(1, 699)$, $(1, 449)$, $(1, 252)$, $(1, 72)$, $(5, 5)$.

We apply Lemma 3.5.7 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 3.5.12, 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. The attack takes less than a minute.

In the last example, we illustrate the skipping attack when $m = 1$. We choose $r = 10$. The rest of the parameters are the same. After we recover the subkey K_s as in the previous example, we apply the permutation orbit attack with $K_i^1 =$`0xFFFF` and we obtain the orbit structure of π_1^{10} as $(2, 26856)$, $(2, 3216)$, $(5, 779)$, $(1, 699)$, $(1, 449)$, $(2, 126)$, $(2, 36)$, $(5, 5)$.

We apply Lemma 3.5.7 only to lone orbits of length 699, 449 in π_1^{10} to reveal 1148 entries in σ. This corresponds to 1.75% of the map σ.

Now we apply the skipping attack to reveal the rest of σ. We first extract the orbit structures of γ_1^5 and γ_2^5 for $K_i^1 =$`0x5555` and $K_i^2 =$`0xAAAA`, respectively. They have the same structure and it is given as $(1, 27196)$, $(1, 13976)$, $(1, 13571)$, $(1, 9404)$, $(1, 307)$, $(1, 226)$, $(1, 207)$, $(5, 118)$, $(1, 28)$, $(1, 27)$, $(2, 2)$.

We first use the lone cycles in γ_1^5 with Lemma 3.5.7, to recover a portion of γ_1. From the partially revealed portion of γ_1, we know $\gamma_1(3) = 1898$ and $1898 \notin \sigma(R)$. We want to use this to reveal $\sigma^{-1}(1898)$. Writing Eq.(3. 25) with $x = 3$, we have

$$1898 = \sigma(\sigma(\texttt{0x0003} \oplus \texttt{0xAAAA}) \oplus \texttt{0x5555}).$$

Seeing that `0x0003`$\oplus$`0xAAAA`$\in R$ and $\sigma($`0x0003`$\oplus$`0xAAAA`$) = 35007$, we obtain $1898 = \sigma(35007 \oplus$`0x5555`$)$ or $1898 = \sigma(56810)$. Using this with Lemma 3.5.12 in π_1^{10}, we see that $\sigma(56810) = 1898$ maps across two orbits of length 3216 in π_1^{10}. Hence, we reveal 3215 new points in σ. Continuing in this fashion with points in γ_1, we quickly reveal all of the map σ. The attack takes less than a minute.

3.7 Concluding Remarks

In this chapter, we gave a complete break of a cryptosystem that uses discretized two-dimensional chaotic maps. We showed a dependence among secret parameters that yield a smaller key space. We next showed that 16 bits of the key can be revealed using a chosen ciphertext attack. We gave an equivalent representation of the cryptosystem that correspond to replacing two of the secret parameters with an unknown permutation over a small set. We gave permutation orbit, expansion and skipping attacks that, in combination, reveals the unknown permutation. Using simulation with different parameters, we also demonstrated the feasibility of our attacks and gave the simulation results.

Chapter 4
Cryptanalysis of Chaotic Map Lattice Based Systems

An cryptosystem must obey all the rules to work correctly in all the cases. For example, an encryption algorithm must be invertible. Otherwise, it becomes impossible to uniquely recover the concealed messages. Moreover, the algorithm needs to operate correctly for all defined inputs and in machines working with finite precision arithmetic.

In this chapter, I demonstrate our published comment [19] on an article [20], that a recently proposed chaotic encryption system is not invertible under double precision arithmetic. In Section 4.1, I describe the Logistic Map in detail. Then I describe the proposed encryption scheme in Section 4.1. In Section 4.2 we show that the algorithm works incorrectly for some inputs. After stating the wrong cases, I give a modified propose for the cryptosystem in Section 4.3. In Section 4.4, I give Our break for the modified scheme. Finally, I illustrate the break by giving some simulation results and concluding remarks.

4.1 Description of the Cryptosystem

The proposed encryption scheme in [20] uses logistic map as a one-dimensional chaotic map. Mathematically, the logistic map is written in [21] as

$$x_{n+1} = rx_n(1 - x_n). \tag{4. 1}$$

Here xn is a number between zero and one, and represents the population at year n, and hence x_0 represents the initial population (at year 0). Also r is a positive number, and represents a combined rate for reproduction and starvation.

The parameters should be chosen in appropriate ranges for the logistic map to behave as a chaotic map. The article [20] describes the chaotic logistic map as

$$x(k + 1) = f(x(k)) = ax(k)(1 - x(k)). \tag{4. 2}$$

The algorithm is proposed as a image encryption scheme. Let us denote the set of gray levels $\{0, 1, \ldots, 255\}$ by T, and let the vector $c \in T^m$ represent a vector, which is produced by converting an image into a column vector. For an $N \times M$ image,

m is the total number of pixels, i.e. $m = NM$. The encryption algorithm takes the vector c as the plaintext input, and it generates another vector $d \in T^m$ as the ciphertext output. The algorithm needs to follow three steps for this transformation; a D/A conversion to work with real numbers, chained chaotic iteration for a number of cycles, and finally an A/D conversion to convert the real numbers into integers.

To apply the D/A conversion, each integer pixel value c_i is mapped to one of 256 distinct real values in the chaotic attractor $(x_{\min}, x_{\max})$ using

$$x_i = x_{\min} + (x_{\max} - x_{\min})\frac{c_i}{255}, \quad 1 \leq i \leq m, \tag{4. 3}$$

The maximum value of x is known as $x_{\max} = a/4$, and iterating the maximum value of x once yields the minimum value for x. Hence, $x_{\min} = (4a^2 - a^3)/16$.

After the D/A conversion process, the real values x_i are transformed using repeated chaotic iteration as follows. The initial values for cycle 0 are $y_i(0) = x_i$, $1 \leq i \leq m$. The transformation for the j^{th} cycle, $j \geq 1$, is given as

$$\begin{aligned} y_1(j) &= A(f^n(y_m(j-1)) + y_1(j-1)), \\ y_i(j) &= A(f^n(y_{i-1}(j)) + y_i(j-1)) \quad i \geq 2,\ 1 \leq j \leq r, \end{aligned} \tag{4. 4}$$

where the function $A : \mathbf{R} \to \mathbf{R}$ is defined as

$$A(u) = \begin{cases} u, & u \leq x_{\max}, \\ u - (x_{\max} - x_{\min}), & u > x_{\max}, \end{cases} \tag{4. 5}$$

and r denotes the number of cycles in the encryption. In Eq.(4. 4), the logistic map f is iterated n times starting with the initial value $y_{i-1}(j)$ for $i \geq 2$ and with $y_m(j-1)$ for $i = 1$.

In the proposal [20], Eq.(4. 4) is expressed slightly different without the function A. Here, we introduced the function A to emphasize that Eq.(4. 4) tries to make sure that the transformed value $y_i(j)$ remains within the attractor. However, as we show in the sequel, this choice is incorrect.

In the final A/D conversion step, $y_i(r)$ is mapped back to an integer d_i in T using

$$d_i = \text{round}\left[(y_i(r) - x_{\min})\frac{255}{x_{\max} - x_{\min}}\right]. \tag{4. 6}$$

In decryption, the algorithm has the vector d as its input and generates the original plaintext vector c. The decryption also has three steps; D/A conversion, chained chaotic iteration in the reverse direction, and A/D conversion.

In the D/A step, we use

$$y_i(r) = x_{\min} + (x_{\max} - x_{\min})\frac{d_i}{255}, \quad 1 \le i \le m \tag{4. 7}$$

to map the integer values d_i to the fixed locations $y_i(r)$ in the attractor.

In the chained chaotic iteration step, we work backwards through j cycles to get the original real values using

$$\begin{aligned} y_i(j-1) &= B(y_i(j) - f^n(y_{i-1}(j))), \quad i \ge 2,\ 0 \le j \le r \\ y_1(j-1) &= B(y_1(j) - f^n(y_m(j-1))), \end{aligned} \tag{4. 8}$$

where the function $B : \mathbf{R} \to \mathbf{R}$ is defined as

$$B(u) = \begin{cases} u, & u \ge 0, \\ u + (x_{\max} - x_{\min}), & u < 0. \end{cases}$$

In the proposal [20], Eq.(4. 8) is expressed slightly different without the function B. Here, we introduced the function B to emphasize that Eq.(4. 8) tries to make sure that the transformed value $y_i(j-1)$ remains within the attractor.

In the A/D step, we map the real values $x_i = y_i(0)$ to corresponding integer values in T using

$$c_i = \text{round}\left[(y_i(0) - x_{\min})\frac{255}{x_{\max} - x_{\min}}\right], 1 \le i \le m. \tag{4. 9}$$

4.2 Analysis

There are several problems with the proposed cryptosystem based on chaotic Logistic Map.

First of all, it is possible to have values outside the attractor $(x_{\min}, x_{\max})$ in encryption process due to the functions $A(\cdot)$ and $B(\cdot)$. Hence, the succeeding iterations will not behave chaotically and give the values outside the attractor. Moreover, A/D conversion step Eq.(4. 6) may yield pixel values not in T. Indeed, using Eq.(4. 4) with $j = 1$ and $i \ge 2$, we have

$$y_i(1) = A(f^n(y_{i-1}(1)) + y_i(0)), \tag{4. 10}$$

$$y_{i+1}(1) = A(f^n(y_i(1)) + y_{i+1}(0)). \tag{4. 11}$$

Note that, in Eq.(4. 11), the initial value for the iteration of f is $y_i(1)$. Also, $y_i(1)$ is obtained as the value of the function A in Eq.(4. 10). Thus, in order for the initial value $y_i(1)$ to be inside the attractor, the function A must yield values inside the attractor. We demonstrate with a simple example that this is not the case in the scheme proposed in [20].

Let us choose $n = 1$, $a = 3.9$, $j = 1$, $y_{i-1}(1) = 0.5$, and $y_i(0) = x_{\max} - \epsilon$, with $i \geq 2$. Using Eq.(4. 10) and Eq.(4. 2), we have $y_i(1) = A(f(0.5) + x_{\max} - \epsilon) = A(2x_{\max} - \epsilon)$. If ϵ is small enough, we have $2x_{\max} - \epsilon > x_{\max}$ and using Eq.(4. 5) with $u = 2x_{\max} - \epsilon$, we find $y_i(1) = x_{\max} + x_{\min} - \epsilon$. Clearly, for small ϵ, $y_i(1)$ is outside the attractor $(x_{\min}, x_{\max})$. Therefore, using the functions $A(\cdot)$ and $B(\cdot)$ as proposed in [20] leads to incorrect operation.

The proposed encryption algorithm needs to be improved against these incorrect operations, and it should be guaranteed that all the values fall within the attractor. Since the fault arises due to the functions $A(\cdot)$ and $B(\cdot)$, we suggest the following modified functions for $A(\cdot)$ and $B(\cdot)$.

$$A(u) = \begin{cases} u, & u \leq x_{\max}, \\ u - (x_{\max} - x_{\min}), & x_{\max} < u \leq 2x_{\max} - x_{\min}, \\ u - 2(x_{\max} - x_{\min}), & 2x_{\max} - x_{\min} < u. \end{cases}$$

$$B(u) = \begin{cases} u, & u \geq x_{\min}, \\ u + (x_{\max} - x_{\min}), & -x_{\max} + 2x_{\min} \leq u < x_{\min}, \\ u + 2(x_{\max} - x_{\min}), & u < -x_{\max} + 2x_{\min}. \end{cases}$$

Even though these improvements correct the first problem, there still is another problem with the proposal. The problem is that the encryption transformation which is defined in Eqns.(4. 3, 4. 4, 4. 6) is not invertible because it involves many-to-one rounding function.

In the D/A step of the decryption, $y_i(r)$ values calculated by Eq.(4. 7) is one of the 256 fixed points on the attractor. However, the real value $y_i(r)$ calculated using Eq.(4. 4) is not necessarily one of those fixed 256 real values. Indeed, Eq.(4. 6) maps $y_i(r)$ to the integer value d_i corresponding the closest of the fixed points by way of rounding. Thus, when decrypting, the initial value to the chaotic map f in Eq.(4. 8) is one of the points on the fixed grid. Therefore, $y_i(0)$ calculated in Eq.(4. 8) is different than the original x_i used in Eq.(4. 4). If this difference is large enough, then the decrypted color value is different than the original color value c_i. Hence, the encryption as described in [20] is not an invertible function. In the following example, we illustrate the fact with numerical values.

Example 4.2.1 *To illustrate the preceding argument, we pick a vectorized image with only two pixels with*

$$c_1 = 0, \qquad c_2 = 25.$$

Let the encryption keys be given as $a = 3.9$, $n = 25$, $r = 1$. In this case, using Eq.(

4. 3) and Eq.(4. 4), we find

$$x_1 = 0.0950625, \qquad x_2 = 0.181330882352941$$

and

$$y_1(1) = 0.663955819836359, \qquad y_2(1) = 0.875143546668635.$$

By Eq.(4. 6), these values correspond to the encrypted color values

$$d_1 = 165, \qquad d_2 = 226.$$

In decrypting these values, we use Eq.(4. 7) to obtain the fixed points on the grid as

$$y_1(1) = 0.664433823529412, \qquad y_2(1) = 0.874928676470588.$$

Using these values in Eq.(4. 8), we obtain

$$x_1 = y_1(0) = 0.244811426834675, \qquad x_2 = y_2(0) = 0.769496460931825.$$

Finally, we use Eq.(4. 9) to obtain the decrypted color values

$$c_1 = 43, \qquad c_2 = 195.$$

Obviously, these are different than the original values fed into the encryption process. Therefore, the encryption transformation is not invertible.

There is still a remedy for this incorrectness, which might be to omit the A/D step in the encryption and directly communicate the real values $y_i(r),\ 1 \leq i \leq m$ to the decrypting party. Then due to the fact that encryption and decryption must be symmetric, we also omit the D/A step in the decryption. This is not a good remedy since it has the cost of increasing the communication bandwidth. Indeed, if we use double precision floating point, eight bytes need to be transmitted for each encrypted pixel. In the non-invertible scheme, only one byte needs to be transmitted for each pixel.

The most important problem with the proposal is that the multiple round encryption is not invertible with the usage of finite precision arithmetic. This is the most serious fault of the three. Although previous two problems can be overcome by simple modification on how the transformation is performed, we could see no way to avoid the last one as long as finite precision arithmetic is used. The reason is that the addition operation is not invertible in finite precision.

Example 4.2.2 *We illustrate this point by starting off with a two pixel image with*

$$c_1 = 9, c_2 = 30.$$

In all the calculations we used GNU C compiler gcc 4.0.1 running on Mac OS X 10.5.2 with Intel Core 2 Duo 2.16 GHz. We used double type for all the real numbers. Let the encryption keys be given as $a = 3.9$, $n = 75$, $r = 2$. *Again, using Eq.(4. 3) and Eq.(4. 4), we obtain*

$$x_1 = 0.12611911764705890926, \quad x_2 = 0.19858455882352948896$$

and

$$y_1(1) = 0.47374838065180635560, \quad y_2(1) = 0.60677852279589572504.$$

Using the $y(1)$ *values in a second round, we find*

$$y_1(2) = 0.87230863276263015393, \quad y_2(2) = 0.96099263265698176006.$$

Now we start off with $y(2)$ *and work backwards to decrypt. We assume that the receiver has the exact values* $y_1(2)$ *and* $y_2(2)$. *Using these values in Eq.(4. 8), we obtain*

$$y_1(1) = 0.47374838065180641111, \quad y_2(1) = 0.60677852279589572504.$$

Note that $y(1)$ *so obtained is slightly different than the one obtained in encryption. This is due the non-invertible nature of finite precision addition. When* $y(1)$ *is subsequently used in the chaotic iteration Eq.(4. 8) once more, the difference is amplified. Thus we obtain,*

$$x_1 = y_1(0) = 0.36869120468326127549, \quad x_2 = y_2(0) = 0.50596375159066242500.$$

These real values correspond to the color values

$$c_1 = 79, \quad c_2 = 119.$$

Clearly, the encryption algorithm is not invertible.

4.3 Suggested Improvements and the Modified Cryptosystem

In these analysis we pointed out three flaws in a previous proposal for chaotic encryption. We showed that the encryption function is not well-defined for some values. We also showed that the rounding operation and the finite precision arithmetic renders the algorithm non-invertible. We suggested remedies for two of the problems we identified.

A useful way to improve the proposed chaotic scheme is that working with integer values rather than real numbers as suggested in [22]. If we impose this suggestion to the cryptosystem, the encryption and decryption algorithm changes slightly.

Let us denote with say DA the function of D/A conversion shown in Eq.(4. 3), and by AD the function of A/D conversion shown in Eq.(4. 6) as well. Then the encryption algorithm can be slightly modified as

$$\begin{aligned} c_1(j) &= AD(f^n(DA(c_m(j-1)))) + c_1(j-1) \quad \text{mod } L, \\ c_i(j) &= AD(f^n(DA(c_{i-1}(j)))) + c_i(j-1) \quad \text{mod } L, \end{aligned} \tag{4. 12}$$

where $i \geq 2$, $1 \leq j \leq r$ and L is the number of gray levels, which is 256 for RGB images.

After finalizing the process in Eq.(4. 12), we can denote the ciphertext as $d_i = c_i(r)$. Decryption is also becomes

$$\begin{aligned} c_i(j-1) &= c_i(j) - AD(f^n(DA(c_{i-1}(j)))) \quad \text{mod } L, \\ c_1(j-1) &= c_1(j) - AD(f^n(DA(c_m(j-1)))) \quad \text{mod } L, \end{aligned} \tag{4. 13}$$

where $i \geq 2$, $r \geq j \geq 1$ and L is the number of gray levels, which is 256 for RGB images. The final values $c_i(1)$ are the decrypted plaintexts.

From now on, the algorithm is invertible due to the usage of integer values in every step except the chaotic map. Applying the modulus operation to the integer values rather than real numbers, that is replacing A and B with a modulus operation, makes the algorithm invertible.

The algorithm may become faster by the usage of a substitution map for the iterated chaotic function. That is, we can calculate $AD(f^n(DA(t)))$ for all gray levels $t \in T$ just before either encryption or decryption processes, and use the values during the processes. Let's denote the map with S, so that $S(t) = AD(f^n(DA(t)))$. Now encryption process, expressed in Eq.(4. 12), becomes

$$\begin{aligned} c_1(j) &= S(c_m(j-1)) + c_1(j-1) \quad \text{mod } L, \\ c_i(j) &= (S(c_{i-1}(j)) + c_i(j-1)) \quad \text{mod } L, \end{aligned} \tag{4. 14}$$

and similarly decryption process, expressed in Eq.(4. 13), becomes

$$\begin{aligned} c_i(j-1) &= (c_i(j) - S(c_{i-1}(j))) \mod L, \\ c_1(j-1) &= (c_1(j) - S(c_m(j-1))) \mod L. \end{aligned} \tag{4. 15}$$

The last modification makes the encryption algorithm faster. Performing the encryption process via a look up table, instead of iterating the chaotic map n times for each pixel, reduces the number of operations. Moreover, we can increase the number of rounds due to this speed up.

4.4 Chosen Plaintext Attack on Modified Algorithm

The algorithm still has some weaknesses in spite of the suggested modifications. The only thing for an attacker to do is to reveal the S map. The attacker can reveal the map via a chosen plaintext attack. The attack is explained below step by step.

The first step is to obtain ciphertexts for all possible two pixel images. That is the attacker makes the user encrypt 65536 two pixel images.

Let the plain image $c = (c_1, c_2)$. Let d denote the cipherext. After one round encryption we have

$$d = (d_1, d_2) = ((S(c_2) + c_1) \mod L, \quad (S(d_1) + c_2) \mod L) \tag{4. 16}$$

We can express one round encryption as

$$d = (d_1, d_2) = \pi(c_1, c_2), \tag{4. 17}$$

where $\pi : T \times T \to T \times T$ is a one-to one mapping over $T \times T$.

Then the r round encryption becomes

$$d = (d_1, d_2) = \pi^r(c_1, c_2). \tag{4. 18}$$

The decryption of a single round can be expressed as

$$c = (c_1, c_2) = \pi^{-1}(d_1, d_2). \tag{4. 19}$$

then the r round decryption becomes

$$c = (c_1, c_2) = \pi^{-r}(d_1, d_2). \tag{4. 20}$$

If one has a point in permutation of a single round encryption π, he ca reveal two points in S map using Eq.(4. 19) and Eq.(4. 16) as

$$S(c_2) = (d_1 - c_1) \mod L$$
$$S(d_1) = (d_2 - c_2) \mod L) \quad (4.\ 21)$$

Now, I will show that how an attacker can reveal partial information of π using the result of r-round encryption π^r. First, the attacker applies the chosen-plaintext attack mentioned above and obtains the permutation π^r.

As we can see in [20], the round number r should be very small for a fast encryption. Therefore, it is very easy for an attacker to crack it via a brute force attack. Let's assume that the attacker knows the number of rounds and applies the permutation orbit attack, introduced in Chapter 3, to the map π^r. The attacker can reveal partial information using the orbit structure of π^r and the permutation orbit attack. You can see the necessary definitions, lemmas and proofs for the permutation orbit attack in Chapter 3.

Let $(x, y) \in R$, where R denotes the revealed portion of π. That is the attacker knows the value of π at the points (x, y). Using Eq.(4. 21) the attacker can reveal two points in the map S. The attacker can reveal a huge amount of S map even if the revealed portion of π is very small. In general the attacker reveals the whole map. For example, assume that an attacker reveals %0.1953125 of the permutation π using the permutation orbit attack. This means that the attacker has $65536 \times \%0.1953125 = 128$ points in π. Using these points the attacker can reveal at most 256 points in the map S without an overlap. Since there will be overlaps and the revealed points will be less than or equal to 256.

Even if the attacker is unable to reveal all the map using the permutation orbit attack, he can still continue to reveal more points using another attack via consistency checks.

Let (x_i, y_i) be a point in one of the unrevealed orbit

$$\alpha = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

of π^r, where n is the length of the orbit. The attacker checks to see that whether the orbit is a lone orbit by using Lemma 3.7.7 and Eq.(4. 21). The test can be done as

$$\pi(x_i, y_i) = (x_{i+R^* \pmod n}, x_{i+R^* \pmod n}),\ 0 \leq j < n,$$

where R^* is the multiplicative inverse of R in mod n, i.e. $RR^* \equiv 1 \pmod n$. The attacker uses Eq.(4. 21) to see if the equation holds. If Eq.(4. 21) yields revealed points in S and if the results hold with the revealed values, the attacker reveals

more points in S by adding this orbit. If the check yields the unrevealed points of S, the attacker continues with the next value and continues in this fashion until a consistent or an inconsistent point is found. If the attacker fins inconsistent results, the attacker stops this process on this orbit and continues with another unrevealed orbit.

Unless the attacker reveals the whole S map, he continues to make another consistency check. Let (x_i^1, y_i^1) and (x_j^2, y_j^2) be two points in the orbits α^1 and α^2 respectively. Let the orbits α^1 and α^2 be two unrevealed orbits in π^r with same length of n. The attacker uses Lemma 3.7.12 and Eq.(4. 21) to see whether these points maps each other in π. If Eq.(4. 21) yields the revealed points in S and if the values hold with the revealed values, the attacker adds new points in S. If the attacker finds such an arbitrary point in two different orbits α^1 and α^2, he starts to add these two orbits into the revealed portion of π and reveal more points in S using Eq.(4. 21) and Lemma 3.7.12. Similarly if the check yields the unrevealed points of S, the attacker continues with the next value and continues in this fashion until a consistent or an inconsistent point is found.

Adding a large orbit may cause to find all the unrevealed portion of the S map. Hence, the attacker starts to apply the consistency check from the large sized orbits.

The attacker uses all these attacks in a combination and reveals the map S successfully.

4.5 Simulation Results

We give simulation result for an example that illustrates our method of cryptanalysis. Simulations are performed under MATLAB running on Mac OS X 10.5.4 with Intel Core 2 Duo 2.16 GHz processor and 2 GB RAM.

In our example, we select the parameters as; $a = 3.81$, iteration number $n = 75$ and round number $j = 20$. After applying the chosen-plaintext attack as explained above, we obtained the two dimensional map and calculated its orbits as in Chapter 3.

After applying the permutation orbit attack, we revealed 2.9% of the two dimensional map, which helps us to reveal 52.73% of the S map using Eq.(4. 21).

Then we apply the consistency check and revealed the remaining portion of S map.

The attack takes less than a minute.

4.6 Concluding Remarks

We showed that the proposed cryptosystem in [20] is not invertible for some cases due to finite precision arithmetic and rounding operation. Then we gave another proposal to improve the proposed encryption scheme. The new proposition slightly alters the algorithm. Also, we give a complete break for the new proposed algorithm. In our break we use chosen plaintext attack to reveal the necessary parameters.

Chapter 5
Cryptanalysis of a Chaos-Based Image Encryption Algorithm

A chaos-based image encryption algorithm [23] was proposed on Physics Letters A. The encryption algorithm first shuffles the image pixels using Arnold's Cat Map [15]. After shuffling process the proposed algorithm changes the gray levels of the image pixels using Chen's Chaotic System [24]. In this chapter, I give our complete break [25] for the proposed cryptosystem using either a chosen plaintext attack or a known plaintext attack.

First, Arnold's Cat Map and Chen's Chaotic System are described in Sections 5.1 and 5.2 respectively. Then the proposed encryption algorithm is described in detail, in Section 5.3. In Section 5.4, I demonstrate our chosen plaintext attack that reveals all the secret parameters. In Section 5.5, I give our known-plaintext attack which does the same as in previous section. In Section 5.6, I illustrate the success of our break with some simulation examples. Finally, in Section 5.7, I give some concluding remarks.

5.1 Arnold's Cat Map

I described Arnold's Cat Map in Chapter 3 in detail. Here, I give the description of the two-dimensional chaotic map given in [23].

Arnold's Cat Map is applied to $N \times N$ images. Assume that we have a $N \times N$ image P with the pixel coordinates $I = \{(x, y)|x, y = 0, 1, 2, ..., N-1\}$. In [23], Arnold's Cat Map is given as,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = A \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & p \\ q & pq+1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \mod N, \qquad (5.\ 1)$$

where p, q are positive integers and x', y' are the coordinate values of the shuffled pixel. Using this map with iterations, images are shuffled chaotically.

5.2 Chen's Chaotic System

Chen's chaotic system is a set of differential equations, and is given in [23] as,

$$\begin{aligned} \dot{x} &= a(y-x), \\ \dot{y} &= (c-a)x - xz + cy, \\ \dot{z} &= xy - bz. \end{aligned} \tag{5. 2}$$

where a, b and c are the parameters of the system. Chen's system is chaotic when the parameters have the values; $a = 35$, $b = 3$ and $c \in [20, 28.4]$. There is an illustration of the chaotic system for some parameter values in Figure 5. 1.

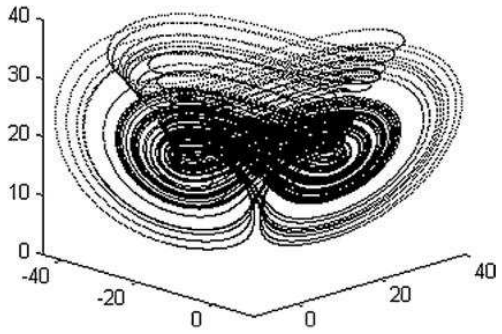

Figure 5. 1 Chaotic behavior of Chen's system

To discretize the chaotic system, [23] chooses a value for Runge-Kutta step size. In the proposed encryption scheme the step of Runge-Kutta is chosen as 0.001. The Eq.(5. 2) is solved using the chosen step size.

5.3 Description of the Cryptosystem

In this section, we describe the encryption algorithm in detail. The encryption process involves two parts. In the first part, the algorithm takes an image P as the plaintext input and shuffles its pixels using Arnold Cat map. The second part of the algorithm involves the process of changing the gray levels of the pixels using Chen's chaotic system.

By the n times iterations of Arnold's Cat Map, we obtain

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = A^n \begin{bmatrix} x \\ y \end{bmatrix} \mod N = M \begin{bmatrix} x \\ y \end{bmatrix} \mod N, \tag{5. 3}$$

where

$$M = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} = A^n \mod N.$$

The resulting shuffled image S has a relation with the original image P such that $S(x', y') = P(x, y),\ 0 \leq x, y \leq N-1$.

5.3.1 Encryption Algorithm

The parameters of Arnold's Cat Map p, q, n and the initial values of Chen's Chaotic System are used as secret parameters of the cryptosystem. Then the encryption algorithm is given as follows:

1. Shuffle the image P using Arnold Cat Map and obtain the shuffled image S.

2. Rearrange the image pixels as the sequence $S = \{s_1, s_2, ..., s_{N\times N}\}$ by scanning the image S using the usual row-scan method.

3. Chose Runge-Kutta step size as 0.001. Iterate Chen's chaotic system $N_0 = (N \times N)/3$ times and obtain the real values x_i, y_i, z_i , $1 \leq i \leq N_0$.

4. Obtain the key sequence $K = \{k_1, k_2, ..., k_{N\times N}\}$ as,

$$\begin{aligned} k_{3(i-1)+1} &= |x_i - \lfloor x_i \rfloor| \times 10^{14} \mod 256, \\ k_{3(i-1)+2} &= |y_i - \lfloor y_i \rfloor| \times 10^{14} \mod 256, \\ k_{3(i-1)+3} &= |z_i - \lfloor z_i \rfloor| \times 10^{14} \mod 256, \end{aligned} \tag{5. 4}$$

   where $\lfloor x \rfloor$ denotes the largest integer not larger than x. Here, we assume that the encryption setup represents the real numbers with 14 decimal digits after the point.

5. Obtain the encrypted sequence $C = \{c_1, c_2, ..., c_{NxN}\}$ as,

$$c_i = s_i \oplus k_i, \ 1 \leq i \leq N \times N, \tag{5. 5}$$

   where $\oplus$ represents bitwise exclusive OR operation.

6. Reshape the sequence C into an $N \times N$ image, and obtain the ciphertext image.

5.4 Chosen-Plaintext Attack

The secret parameters of the proposed encryption algorithm can be extracted using a chosen-plaintext attack, and here we explain how this can be achieved.

Once the attacker extracts the parameters M and K, he can encrypt and decrypt a ciphertext image. Indeed, the secret parameters are only M and K. Once the attacker has the key K, he can obtain the shuffled image S by using Eq.(5. 5). If he also knows the parameter M, he obtains the original image P using Eq.(5. 3). The attack consists of two steps. First, the parameter K is recovered. Then, the attacker uses K in calculating the parameter M of Arnold's Cat map.

5.4.1 Extracting K

First, the attacker chooses a zero image P_1, which consists of $N \times N$ zero-valued pixels. Then he obtains the corresponding ciphertext image, C_1.

The shuffling process does not change the image because P_1 is identically 0. Hence, the shuffled image S_1 is equal to the image P_1. The attacker uses this fact with Eq.(5. 5), and obtains that the cipher-image C_1 is exactly equal to the key K as

$$C_1 = S_1 \oplus K = P_1 \oplus K = 0 \oplus K = K.$$

The attack does not require further analysis on the obtained ciphertext image since the desired parameter is exactly the obtained ciphertext image.

5.4.2 Extracting M

After finding the parameter K, the attacker chooses another image P_2 and obtains the corresponding ciphertext image C_2. The attacker chooses image P_2 such that it contains only two non-zero and distinct pixels $P_2(1,1) = v_1$, $P_2(1,2) = v_2$ so that $v_1 \neq v_2$ and $v_{1,2} \neq 0$. By using Eq.(5. 5) with the knowledge of K, the attacker obtains the corresponding shuffled image S_2 from C_2 as

$$S_2 = C_2 \oplus K.$$

Now, the attacker has the image P_2 and the corresponding shuffled image S_2. The attacker easily recovers the shuffled image by only applying an XOR process to the chosen ciphertexts.

He then starts to search the pixel values v_1, v_2 in S_2, and the attacker determines the shuffled coordinates (x_1', y_1') and (x_2', y_2').

Using Eq.(5. 3), the attacker obtains the following sets of equations.

$$\begin{aligned} \begin{bmatrix} x_1' \\ y_1' \end{bmatrix} &= \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \mod N, \\ \begin{bmatrix} x_2' \\ y_2' \end{bmatrix} &= \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \mod N. \end{aligned} \tag{5. 6}$$

After solving these equations, the attacker obtains that $m_2 = x_2' - x_1' \mod N$, $m_1 = x_1' - m_2 \mod N$, $m_4 = y_2' - y_1' \mod N$, $m_3 = y_1' - m_4 \mod N$. By evaluating these results the attacker easily obtains the parameters m_1, m_2, m_3 and m_4. Hence, the key M is extracted. Now, the attacker has all the secret parameters of the encryption algorithm.

5.5 Known-Plaintext Attack

The secret parameters of the proposed encryption algorithm can be extracted also by using a known-plaintext attack, and here we explain how this can be achieved.

In this case, the attacker does not choose plaintexts. Instead, we assume that he has obtained some plaintext-ciphertext pairs as the requirement of the known plaintext attack.

The attack consists of two steps. First the attacker tries to calculate the key of Arnold Cat map, M, and then he tries to extract the key K of Chen's chaotic system.

5.5.1 Extracting M

Let's assume the attacker knows two plaintext-ciphertext image pairs (P_1, C_1) and (P_2, C_2). Let us define the difference images as $\Delta P = P_1 \oplus P_2$ and $\Delta C = C_1 \oplus C_2$. Using Eq.(5. 5), the attacker calculates

$$\Delta C = S_1 \oplus K \oplus S_2 \oplus K = S_1 \oplus S_2 = \Delta S$$

Here, we see that the attacker is able to calculate ΔS as shown in above equation. Going from ΔP to ΔS, there is only shuffling by the Arnold Cat map. Now, we give a method to reveal the parameters of this map.

Let $\Delta P(x_1, y_1) = v_1$, $\Delta P(x_2, y_2) = v_2$ be two pixels of ΔP with different values, i.e. $v_1 \neq v_2$, and let (x_1', y_1') and (x_2', y_2') denote their respective coordinates in ΔS. Using these facts and Eq.(5. 3), the attacker obtains

$$\begin{bmatrix} x_1' & x_2' \\ y_1' & y_2' \end{bmatrix} = MU = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} U \bmod N, \tag{5. 7}$$

where $U = \begin{bmatrix} x_1 & x_2 \\ y_1 & y_2 \end{bmatrix}$.

Let V_1 denote the set of coordinates at which ΔS has value v_1. Namely,

$$V_1 = \{(i, j) | \Delta S(i, j) = v_1\}. \tag{5. 8}$$

Similarly, define the set V_2 as

$$V_2 = \{(k, l) | \Delta S(k, l) = v_2\}. \tag{5. 9}$$

Assuming that U is invertible, let us define the set V of matrices as

$$V = \{ \begin{bmatrix} i & k \\ j & l \end{bmatrix} U^{-1} \mid (i,j) \in V_1, (k,l) \in V_2 \}. \quad (5.\ 10)$$

By the definitions of the sets V_1 and V_2, we have $(x_1', y_1') \in V_1$ and $(x_2', y_2') \in V_2$. So, using Eq.(5. 7), we conclude that $M \in V$.

Thus, once the attacker constructs the set V, he has $|V| = |V_1|\,|V_2|$ candidates for M. Repeating the procedure with another pair of pixels $\Delta P(\bar{x}_1, \bar{y}_1) = \bar{v}_1$, $\Delta P(\bar{x}_2, \bar{y}_2) = \bar{v}_2$, $\bar{v}_1 \neq \bar{v}_2$, he finds another set $\bar{V}$ that contains M. Obviously,

$$M \in V \cap \bar{V}. \quad (5.\ 11)$$

Continuing in this fashion, he intersect more and more sets until the intersection contains only one element. This element is necessarily M.

In order to make the set V of candidates smaller, the attacker chooses rare pixel pairs in ΔP.

5.5.2 Extracting K

The attacker knows the image P_1 and the corresponding cipher image C_1. Using Eq.(5. 3) and the Arnold cat map parameter M, which is revealed in the previous step, the attacker calculates S_1.

Using S_1 in Eq.(5. 5), the attacker obtains Chen's chaotic system parameter K as

$$K = C_1 \oplus S_1.$$

Thus, the attacker knows all the secret parameters of the encryption algorithm.

5.6 Simulation Results

In order to illustrate each type of attacks, we give simulation results for two cases. Simulations are performed under MATLAB running on Mac OS X 10.5.4 with Intel Core 2 Duo 2.16 GHz processor and 2 GB RAM. The secret parameters are chosen from the example in [23] . Arnold cat map parameters are; $p = 1, q = 1, n = 5$, Chen's chaotic system parameters are; $a = 35, b = 3, c = 28$, The initial conditions for the Chen system are; $x_0 = -10.058$, $y_0 = 0.368$, $z_0 = 37.368$.

In the first simulation, we illustrate the chosen-plaintext attack. The attacker chooses the following two 9×9 images as plaintexts.

$$P_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad P_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Note that the first image consists of zero valued pixels and the second image has only two nonzero pixels. By (5. 6), the first secret parameter

$$M = \begin{bmatrix} 7 & 1 \\ 1 & 8 \end{bmatrix} \bmod 9.$$

The ciphertext C_1, which is also the Chen key K, is given as

$$C_1 = K = \begin{bmatrix} 255 & 0 & 0 & 210 & 15 & 170 & 32 & 187 & 27 \\ 225 & 138 & 115 & 22 & 29 & 95 & 140 & 62 & 213 \\ 152 & 97 & 255 & 222 & 66 & 34 & 164 & 1 & 148 \\ 225 & 142 & 38 & 133 & 203 & 253 & 201 & 115 & 133 \\ 70 & 223 & 18 & 151 & 239 & 179 & 137 & 251 & 101 \\ 64 & 67 & 157 & 22 & 225 & 39 & 70 & 171 & 25 \\ 83 & 19 & 3 & 209 & 152 & 89 & 121 & 220 & 249 \\ 87 & 170 & 185 & 30 & 240 & 74 & 47 & 173 & 43 \\ 155 & 23 & 132 & 113 & 30 & 94 & 4 & 23 & 31 \end{bmatrix}.$$

Using C_2, the attacker calculates S_2 as $S_2 = C_2 \oplus K$. S_2 is given as

$$S_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Inspecting S_2, the attacker finds that $x_1' = 8$, $y_1' = 0$, $x_2' = 0$, $y_2' = 8$. Substituting these into (5. 6), the attacker finds M.

In the second simulation, we illustrate the known-plaintext attack. Assume that the attacker knows the two 256×256 plaintext images in Figure 5. 2(a) and 5. 2(b). The difference images $\Delta P = P_1 \oplus P_2$ and $\Delta C = C_1 \oplus C_2$ are also shown in Figure 5. 2(c) and Figure 5. 2(d). For this case, we have

$$M = \begin{bmatrix} 34 & 55 \\ 55 & 89 \end{bmatrix} \bmod 256.$$

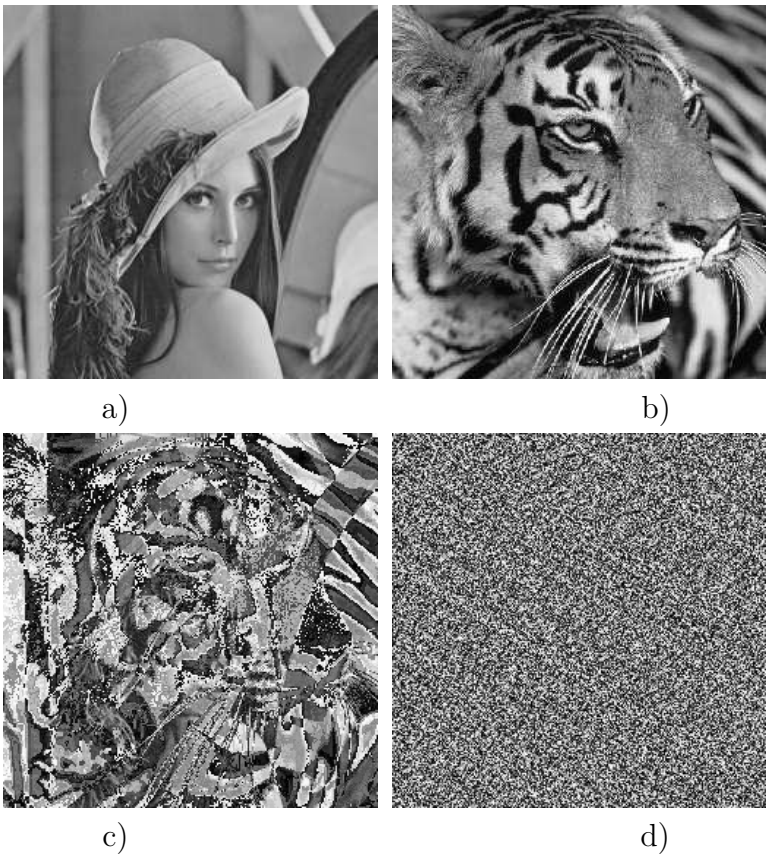

Figure 5. 2 a) Plaintext P_1 b) Plaintext P_2 c) ΔP d) ΔC

The two rarest pixel values in ΔP are $v_1 = 131$ and $v_2 = 140$. Following are procedure in Section (5.4.1), we find 25718 candidates for M. Next rarest pair of pixel values are $\bar{v}_1 = 140$ and $\bar{v}_2 = 120$. This time, there are 29058 candidates. Intersecting the two sets of candidates, we find only one candidate. The attack takes less than one minute.

Finally, after calculating S_1 using (5. 3), the attacker reveals K as $K = C_1 \oplus S_1$. K is given in Figure 5. 3.

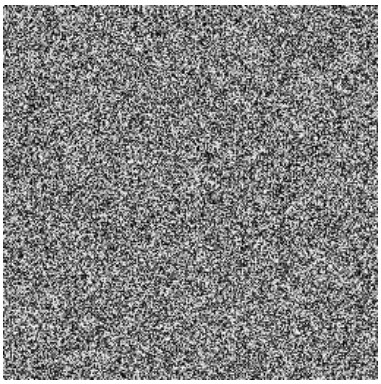

Figure 5. 3 Chen key K

5.7 Concluding Remarks

In this chapter, first I described a proposed chaotic encryption scheme [23]. Then we gave a complete break of the chaos-based image encryption algorithm. We demonstrated that the secret keys can be revealed using chosen and known plaintext attacks.

Chapter 6
Cryptanalysis of a Chaotic Cryptosystem Based on Two Dimensional Chaotic Maps

There was a chaotic cryptosystem based on two-dimensional chaotic maps proposed by Jiri Fridrich in 1998 [16]. The cryptosystem involves a two dimensional chaotic map Baker's Map. The cryptosystem is expanded to three dimensions using some other mapping methods. The cryptosystem is based on a mode of operation stated in the description of the encryption scheme. In this chapter, I give a complete break [26] for the proposed cryptosystem.

In Section 6.1, I give some details for Baker's two-dimensional chaotic map. Then, in Section 6.2, I give the proposed cryptosystem with our point of view, that is we described the cryptosystem with very simple modifications. In Section 6.3 I give our chosen ciphertext attack to the two-dimensional chaotic map. Finally, I give some simulation results to illustrate our attacks.

6.1 Baker's Map

We already described Baker's Map in Chapter 3, so we give the description of the proposed cryptosystem. The map has its own formula. Furthermore, the article gives the generalized version and discretized versions for the chaotic map.

6.1.1 Two-Dimensional Baker's Map

The Baker map, B, is described in [16] with the following formulas

$$\begin{aligned} B(x,y) &= (2x, y/2) \quad , \ when \quad 0 \le x < 1/2 \\ B(x,y) &= (2x-1, y/2+1/2) \quad , \ when \quad 1/2 \le x \le 1. \end{aligned}$$

The map acts on the unit square as depicted in Figure 6. 1. The left vertical column $[0;1=2) \times [0;1)$ is stretched horizontally and contracted vertically into the rectangle $[0;1) \times [0;1=2)$, and the right vertical column $[1=2;1) \times [0;1)$ is similarly mapped onto $[0;1) \times [1=2;1)$. The Baker map is a chaotic bijection of the unit square $I \times I$ onto itself.

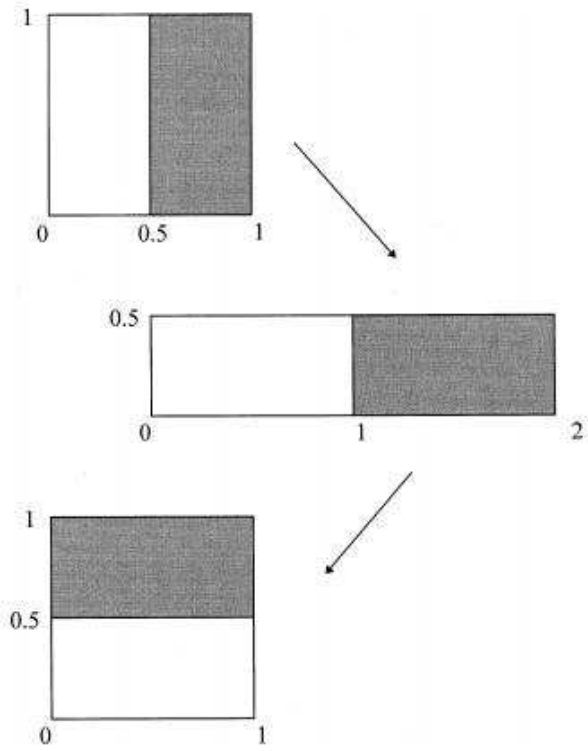

Figure 6. 1 Baker's Map

6.1.2 Discretized Baker's Map

The discretized version of Baker's Map is given in [11] as

$$B_{(n_1,\ldots,n_k)}(r,s) = \left(\frac{N}{n_j}(r - N_j) + s \mod N/n_j,\ \frac{n_j}{N}(s - s \mod N/n_j) + N_j\right).$$

where $n_0+n_1+\ldots+n_t = N$, $n_0+n_1+\ldots+n_j = N_j$, $0 \leq s \leq N$, $N_j \leq r \leq N_j+n_{j+1}$, $0 \leq j \leq t-1$, $n_0 = 0$.

6.2 Description of the Algorithm

Encryption algorithm takes $M \times N$ grayscale images as input. Each pixel of grayscale image is represented using a byte. Preliminarily, the image is vectorized before the encryption starts using the usual row-scan method shown in Chapter 4. Let $p \in S^n$ represent this vectorized image, where $S = \{0, 1, \ldots 255\}$ and $n = NM$. Hence, the plaintext is the vector $p = [p_1\ p_2\ \cdots p_n]$.

The encryption algorithm involves another preliminary process, which is the preparation of a key dependent permutation. The permutation is used, as usual, to shuffle the image pixels in encryption.

After these preliminaries, the round operations start to be applied. Each round of the encryption algorithm consists of two steps. In the first step, p is shuffled using the key dependent secret permutation. The shuffling is done using the discretized version of Baker's Map. Let b denote this key dependent permutation defined on the set $\{1, 2, \ldots, n\}$. Let us denote the shuffled vector by f. Then the relation between the shuffled vector f and the vectorized plaintext p becomes

$$f_i = p_{b(i)},\ 1 \leq i \leq n. \tag{6. 1}$$

In the second step, f is passed through a nonlinear function. The nonlinear function is given as

$$c_i = f_i + g(c_{i-1}) + h_i \bmod 255, \tag{6. 2}$$

where $g : S \to S$ is a fixed nonlinear function and $h \in S^n$ is a fixed vector. The nonlinear function g is used as an S-box since each input $c_i \in S$ is only substituted with another value $g(c_{i-1}) \in S$. Here, we need an initial value c_0, which may also be treated as a secret key.

The number of rounds is denoted by R, so these two steps are repeated R times. In [16], $R = 10$ is suggested for good diffusion and confusion properties.

The encryption process is the combination of the processes described above. Combining Eq.(6. 1) and Eq.(6. 2), we obtain the single round encryption as

$$c_i = p_{b(i)} + g(c_{i-1}) + h_i \mod 256. \tag{6. 3}$$

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

$$j = b(i) \ \Leftrightarrow\ i = u(j). \tag{6. 4}$$

Using Eq.(6. 4) in Eq.(6. 3), we obtain

$$p_j = c_{u(j)} - g(c_{u(j)-1}) - h_{u(j)} \mod 256. \tag{6. 5}$$

In Eq.(6. 3), c_0 is taken to be a system parameter. Hence, for $i = 1$, we have

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

There is only one secret parameter of the proposed algorithm since none of the other operations involves a secret parameter. The secret parameter is only the key dependent permutation b. The size of the secret key of the permutation b differs according to the chosen chaotic map. For example, in the scheme proposed in [16],

the original image P is partitioned using the set of keys, which are the parameters of the discretized version of Baker map. In this case, the set of keys are the partition boundaries. The image is partitioned from those boundaries and shuffled. The discretized Baker's Map is described in Section 6.1 in detail. We may prefer to use another two-dimensional chaotic maps or any other schemes, such as Cat Map or Standard Map, to generate the permutation b. Our attack is general and applicable to all of these cases.

6.3 Chosen Ciphertext Attack

One approach may be extracting the secret parameters of the chaos-based permutation b. This approach is very hard to apply since the number of secret parameters and so the key size is changing when the permutation scheme changes. Even in some cases, such as Baker's Map, the key size is already variable. However, instead of extracting the secret parameters, one can break the system if he reveals the permutation b or the reverse permutation u, since the underlying key is the secret permutation. In our cryptanalysis, we developed some methods to reveal the inverse permutation u. Using such an approach is more general due to the independence of the chosen chaotic map. The attack is applicable to any case where different chaotic maps are used to generate the permutation.

We used the relations between ciphertext and plaintext blocks to reveal the permutation. The function g in Eq.(6. 3) forms a chain that relates consecutive ciphertext pixels. Therefore, in a single round encryption, a change in a plaintext pixel affects many ciphertext pixels. Indeed, if we change $p_{b(i)}$, by Eq.(6. 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}. Therefore, for a single round encryption, a change in $p_{b(i)}$ affects $c_i, c_{i+1}, \ldots, c_n$. In other words, the consecutive ciphertext pixels pass into a chain reaction due to this change. 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 [5].

Even though the encryption process has the desirable properties, the situation is quite different in decryption. For a single round encryption, using Eq.(6. 5) we obtain that only two ciphertext pixels, $c_{u(j)}$ and $c_{u(j)-1}$, affect the plaintext pixel p_j. For two round encryption, we see that 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.(6. 5) with c_k as the plaintext pixel that is input to second

round, we obtain

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

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

$$c_{u(j)} = d_{u^2(j)} - g(d_{u^2(j)-1}) - h_{u^2(j)}. \tag{6. 7}$$

Here, we denote the i times composition of u with itself by u^i, . Similarly, for $k = u(j) - 1$, we have

$$c_{u(j)-1} = d_{u(u(j)-1)} - g(d_{u(u(j)-1)-1}) - h_{u(u(j)-1)}. \tag{6. 8}$$

Thus, we see from Eq.(6. 5), Eq.(6. 7) and Eq.(6. 8) that, for two rounds of decryption, p_j is affected only by the ciphertext pixels

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

Due to the mappings in the reverse permutation u, there may be some overlaps in these four pixels.

Since c_0 is a fixed system parameter, the plaintext pixel $p_{u^{-1}(1)}$ is affected by only c_1. Therefore, in a two round encryption, $p_{u^{-1}(1)}$ is affected by

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

Let's illustrate the causal relations in two round decryption by giving an example.

Example 6.3.1 *Let $n = 6$, and the inverse permutation u be given as*

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

By using this inverse permutation we obtain the causality paths, which are given in Figure 6.3.1. The directed arrows of the figure indicate the nodes, which affect the computation of the destination node. For example, the arrows going from c_5 and c_4 to p_4 means that p_4 is affected by c_5 and c_4.

Then we obtain the causality chain from the ciphertext d to the plaintext p as follows

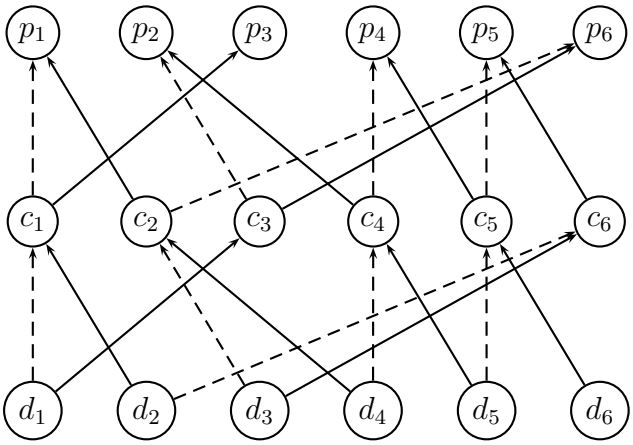

Figure 6. 2 The causality paths for the permutation given in Eq.(6. 9)

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

Note that p_3 is affected by only two ciphertext pixels d_1 and d_2 because $u(3) = 1$. That is p_3 is affected also by the system parameter c_0. Also note that p_4 is affected by three ciphertext pixels rather than four because $u(u(4)-1) = u^2(4)-1 = 5$. There is an overlap in the paths of p_4. This also means that there are two distinct causality paths going from d_5 to p_4.

In an R round decryption, a particular plaintext pixel p_j is generally affected by at most 2^R ciphertext pixels. We can see this fact in Example 6.3.1. For a 256×256 image, $n = 65536$ and $2^R = 1024$. Therefore, about $\frac{1024}{65536} \approx 2\%$ of ciphertext pixels affect any given plaintext pixel.

Let z denote the output, ciphertext image, of an R rounds of encryption. The attacker desires 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 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 causality path from z_i to p_j so that z_i affects p_j. Repeating this 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 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 zeros.

The exception is the $u^{-1}(1)$. For $p_{u^{-1}(1)}$, we have

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

Hence, for one round, $p_{u^{-1}(1)}$ 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 $u^{-1}(1)$ of T contains at most 2^{R-1} 1's.

Example 6.3.2 *The matrix T for the permutation u used in Example 6.3.1 is given as*

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

Here we can extract the same causality paths extracted in Example 6.3.1. For example, from the first column we can say that $p_1 is affected by d_1, d_2, d_3, d_4$ since the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} elements of the first column are 1.

Note that the column of the matrix T with the least number of 1s gives the attacker a starting point for the attack. Once an attacker constructs the matrix T, he can reveal $u^{-1}(1)$ by choosing the column k with the least column sum. Then he knows $u(k) = 1$.

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

Define the operation L on a set A as follows.

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

For an integer $k \in \{1, 2, \ldots, n\}$, $L(\{k\})$ is the set of indices of ciphertext pixels that affect the plaintext pixel p_k in a one round of decryption. So, in general we have

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

We can naturally compose L with itself to define its higher powers. $L^i(\{k\})$ 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 6.3.3 *For the permutation given in Example 6.3.1, we have*

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

Using the definition of L, we have

$$|L^R(\{k\})| \leq 2^R.$$

The operation L naturally defines a graph on the set $\{1,2,\ldots,n\}$.

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 $L^R(\{k\})$, $\forall k \in \{1,2,\ldots,n\}$. The attacker uses this knowledge to reveal the secret permutation u.

Assume that the attacker knows $L^R(\{x\})$ and $L^R(\{y\})$ where $u(x)+1 = u(y)$. By the construction of L, we have

$$L(x) \cap L(y) = u(x).$$

We use this to pin down a z such that $u(z) = u(y) + 1$.

For this z, we have

$$L^R(\{y\}) \backslash L^R(\{x\}) \subset L^R(\{z\}).$$

The sets are illustrated in Figure 6.3.

For a random graph, we expect $L^{R-1}(\{w\})$ to contain about 2^{R-1} elements when $u(w) \neq 1$, $1 \leq w \leq n$. Allowing for overlaps, $L^R(\{y\}) \backslash L^R(\{x\}) \subset L^R(\{z\})$ may contain less than 2^{R-1} elements. Searching through the matrix T, the attacker determines the columns $z_1, z_2, \ldots, z_v$ whose sets of nonzero indices contain $L^R(\{y\}) \backslash L^R(\{x\}) \subset L^R(\{z\})$. So, the attacker knows that

$$u^{-1}(u(y)+1) \in \{z_1, z_2, \ldots, z_v\}. \tag{6. 10}$$

For random graphs, the set on the right hand side of Eq.(6. 10) is likely to contain a single element. If this is the case, we know z such that $u^{-1}(u(y)+1) = z$. Hence, the attacker reveals a new point on the permutation u.

The attack continues with the new pair y and z as we now have $u(z) = u(y)+1$.

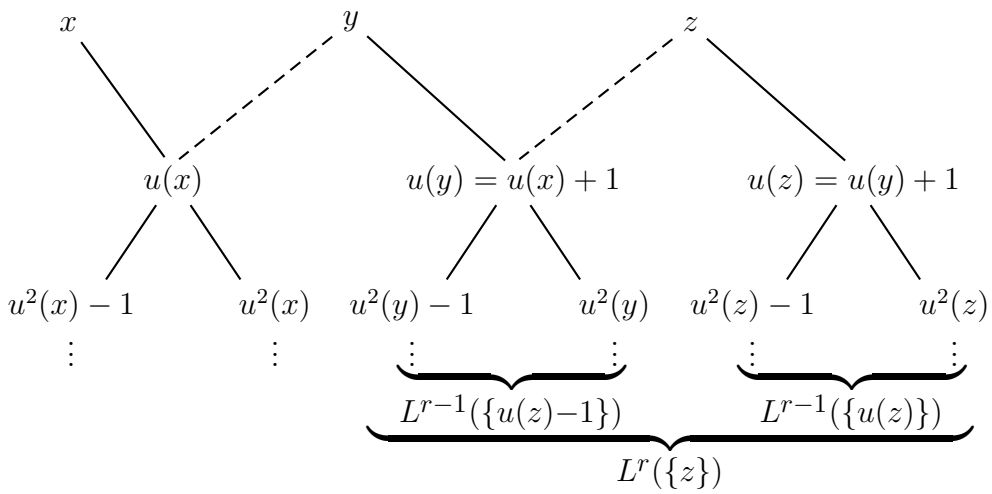

Figure 6. 3 The sets obtained after operation L

In cases when the RHS of Eq.(6. 10) contains more than one candidates, the attacker applies the procedure for each z_m, $1 \leq m \leq v$, each time assuming that $u(z_m) = u(y) + 1$.

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

6.4 Implementation Details

The attack is expressed as an algorithm in Eq.(1)

Data: $L^R(k),\ \forall k \in \{1, 2, \ldots, n\}$
Result: u
Global Variable: k

```
1  Initialize k = 2
2  FindNext()
3  begin
4      Z ← {j|L^R(u^{-1}(k)) \ L^R(u^{-1}(k − 1)) ⊂ L^R(j)} \ u^{-1}({1, 2, ..., k})
5      k ← k + 1
6      if Z ≠ ∅ then
7          foreach z ∈ Z do
8              u^{-1}(k) ← z
9              if k = n then
10                 exit
11             end
12             FindNext()
13         end
14         k ← k − 1
15     end
16     else
17         return
18     end
19 end
```

Algorithm 1: Attack

Instead of the usage of a double array for the matrix T, I implemented a sparce matrix data structure to represent the coordinates of 1s. There are only 0s and 1s in T and the 1s are %2 of the total matrix, so the usage of a double array which holds all 0s and 1s would be inefficient in memory. The data structure of our sparce matrix is given as

```
struct SparceMatrice{
int row;
int col;
struct SparceMatrice *left;
struct SparceMatrice *down;
};
```

Here *row* and *col* denote the row number and column number of 1s, *left* and *down* pointers point the nest 1 of the specified direction.

For example, representing each 0 or 1 by an integer matrix of a 256×256 image requires $65536^2 \times 4 = 2^{34}$ bytes $= 4$ GB which is a huge requirement. By the usage of a sparce matrix, on the other hand, the implementation requires $1024 \times 65536 \times 16 = 2^{30}$ bytes $= 1$ GB, which is more useful. Here the last 16 is the size of the data structure.

6.5 Simulation Results

We give simulation results for two examples that illustrate our method of cryptanalysis. 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.

We first illustrate the attack with an artificially small image size. We chose $R = 2$, a 4×4 random vector image of

$$P = [88 \;\; 216 \;\; 41 \;\; 40 \;\; 130 \;\; 154 \;\; 41 \;\; 162 \;\; 215 \;\; 199 \;\; 67 \;\; 80 \;\; 47 \;\; 114 \;\; 83 \;\; 71],$$

the initial parameter $C_0 = 25$, and a random permutation

$$b = [16 \;\; 7 \;\; 6 \;\; 13 \;\; 1 \;\; 5 \;\; 14 \;\; 12 \;\; 4 \;\; 15 \;\; 8 \;\; 11 \;\; 10 \;\; 3 \;\; 2 \;\; 9],$$

so the inverse permutation

$$u = [5 \;\; 15 \;\; 14 \;\; 9 \;\; 6 \;\; 3 \;\; 2 \;\; 11 \;\; 16 \;\; 13 \;\; 12 \;\; 8 \;\; 4 \;\; 7 \;\; 10 \;\; 1].$$

The other fixed functions g and h are chosen randomly. Here, The matrix T is

calculated for two rounds as

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

Here, you see the least total sum column is 5^{th} which has only two 1s. Hence, we see that

$$u(1) = 5 \Rightarrow L^2(\{5\}) = \{15, 16\}.$$

Then, by construction, we see that the other column comprising 5^{th} column is 15^{th}. Hence,

$$u(2) = 15 \Rightarrow L^2(\{15\}) \backslash L^2(\{5\}) = \{6, 7\}.$$

The column comprising $L^2(\{15\}) \backslash L^2(\{5\})$ is the 14^{th} column. Hence,

$$u(3) = 14 \Rightarrow L^2(\{14\}) \backslash L^2(\{15\}) = \{5\}.$$

The columns comprising the $L^2(\{14\}) \backslash L^2(\{15\})$ are the 2^{nd}, 3^{rd} and 9^{th} columns. Here the recursion starts branching, where each branch continues in this fashion. For example, the branch, where 2^{nd} column is chosen as true cannot continue. Because we obtain

$$u(4) = 2 \Rightarrow L^2(\{2\}) \backslash L^2(\{14\}) = \{4, 13, 14\}$$

and there is no such a column comprising $L^2(\{2\}) \backslash L^2(\{14\})$. Same situation occurs when we try the 3^{rd} column as the true one.

By continuing with 9^{th} column and continuing in this fashion we will end the attack successfully and reveal the inverse permutation u.

In the second example, we used a random vector of a 256×256 image. The image is chosen to be random since our attack does not depend on the input image. We used $R = 10$, the key of the chaotic Baker's Map as $K = [2 \;\; 4 \;\; 2 \;\; 8 \;\; 4 \;\; 8 \;\; 4]$ and the initial parameter $C_0 = 25$. The other fixed functions g and h are also chosen randomly. In this simulation the whole map is recovered successfully in total of 1171 seconds, which is less than 20 minutes. Moreover, the total memory requirement for the whole implementation was 1065171408 bytes which is approximately 1 GB. Note that we used sparse matrix data structure in our implementations.

6.6 Concluding Remarks

In this chapter, we give a complete break of an image encryption algorithm based on two-dimensional chaotic maps. We use the algebraic properties of the permutations and reveal some data about some points in the permutation. After gathering the necessary data we reveal all the permutation recursively.

Chapter 7
Conclusion

In this thesis, we cryptanalyze and show the weaknesses four different chaotic cryptosystems. Moreover, we gave breaks of each cryptosystem in detail by using simulations.

The first cryptosystem was based on two-dimensional chaotic maps. We break the whole system using the weaknesses of the key schedule. We first show that we can reveal a portion of the secret key using a chosen-ciphertext attack. In the secon part of the attack we used the revealed portion and some algebraic properties of the permutations. We used three different attacks, which we developed, in combination. The attacks yields the secret permutation successfully.

The second cryptosystem uses chaotic behavior of the Logistic Map to encrypt images. We first showed that the cryptosystem is not invertible for some cases. We give remedies to correct some of them, however due to the use of finite precision arithmetic we had no suggestions. We then improved the algorithm with some proposed suggestions and modified the cryptosystem. Finally, we showed a break for the modified algorithm. We used chosen plaintext attack which yiields the secret parameters successfully.

The third cryptosystem was based on a two-dimensional and a three-dimensional chaotic maps. The two-dimensional map used for shuffling the image pixels. The three-dimensional map is used to change the gray levels of the shuffled pixels. We broke the algorithm using two different attack. We first showed that the secret parameters can be revealed using a chosen-plaintext attack. Then we showed a known-plaintext attack, which yield the whole key.

The last cryptosystem was based on two-dimensional chaotic maps. The algorithm is used the encrypt images. We gave a chosen-plaintext attack to reveal the secret chaotic map. The attack yields the secret permutation successfully.

References

[1] Ruselli, J., "Cryptology Home Page", *http://home.cogeco.ca/ cipher/* (2008).

[2] "Cryptology", *http://www.resonancepub.com/homecrypto.htm*, Resonance Publications, Inc. (2008).

[3] Stamp, M., Low, R. M., *Applied Cryptanalysis*, John Wiley & Sons, Inc. (2007).

[4] Katz, J., Lindell, Y., *Introduction to Modern Cryptography*, CRC Press (2007).

[5] Menezes, A., van Oorschot, P., Vanstone, S., *Handbook of Applied Cryptography*, CRC Press (1997).

[6] Stinson, D. R., *Cryptography Theory and Practise*, 3rd ed., CRC Press (2006).

[7] Shannon, C. E., "Communication Theory of Secrecy Systems", *Bell System Technical Journal*, vol.28-4, pp. 656-715 (1949).

[8] "Cryptography", *http://www.giac.org/resources/whitepaper/ cryptography/57.php*, Global Information Assurance Certification (2009).

[9] Kocarev, L., "Chaos-Based Cryptography: A Brief Overview", *IEEE CAS Newsletter*, pp. 18-19 (2001).

[10] Baptista, M. S., "Cryptography with Chaos", *Physics Letters*, pp. 1-3 (1998).

[11] Xiang, T., Wong, K.-W., Liao, X., "A Novel Symmetrical Cryptosystem Based on Discretized Two-Dimensional Chaotic Map", *Physics Letters A*, 364 252 (2007).

[12] Solak, E., Çokal, C., "Cryptanalysis of a Cryptosystem Based on Discretized Two-Dimensional Chaotic Maps", *Physics Letters A*, vol. 372 issue. 46, pp. 6922-6924 (2008).

[13] Solak, E., Çokal, C., "Algebraic Break of a Cryptosystem Based on Discretized Two-Dimensional Chaotic Maps", *Physics Letters A* (2009).

[14] Weisstein, E. W., "Standard Map", MathWorld, *http://mathworld.wolfram.com/StandardMap.html* (2009).

[15] Peterson, G., "Arnold's Cat Map", *http://online.redwoods.cc.ca.us/instruct /darnold/laproj/Fall97/Gabe/ catmap.pdf* (1997).

[16] Fridrich, J., "Symmetric Ciphers Based on Two-Dimensional Chaotic Map", *Int. J. of Bifurcation and Chaos*, vol. 8(6), pp. 12591284 (1998).

[17] Kocarev, L., Jakimoski, G., "Logistic Map as a Block Encryption Algorithm", *Physics Letters A*, 289 199 (2001).

[18] Fraleigh, J. B., *A First Course in Abstract Algebra*, 5th ed., Addison-Wesley (1993).

[19] Solak, E., Çokal, C., "Comment on Encryption and Decryption of Images with Chaotic Map Lattices.", *Chaos*, vol.18, pp. 038101, DOI:10.1063/1.2966114 (2008).

[20] Pisarchik, A. N., Flores-Carmona, N. J., Carpio-Valadez, M., "Encryption and Decryption of Images with Chaotic Map Lattices", *Chaos*, vol. 16, pp. 033118 (2006).

[21] Weisstein, E. W., "Logistic Equation", MathWorld, *http://mathworld.wolfram.com/LogisticEquation.html* (2009).

[22] Arroyo, D., Rhouma, R., Alvarez, G., Li, S., Fernandez, V., "On the Security of a New Image Encryption Scheme Based on Chaotic Map Lattices", *Chaos*, vol. 18, Issue 3, pp. 033112-033112-7 (2008).

[23] Guan, Z.-H., Huang, F., Guan, W., "Chaos-Based Image Encryption Algorithm", *Physics Letters A*, vol. 346, pp. 153 (2005).

[24] Chen, G., Ueta, T., "Yet Another Chaotic Attractor", *Int. J. Bifurcation and Chaos* vol. 9(7), pp. 1465-1466 (1999).

[25] Çokal, C., Solak, E., "Cryptanalysis of a Chaos-Based Image Encryption Algorithm", *Physics Letters A* (2009).

[26] Solak, E., Çokal, C., "Cryptanalysis of a Symmetric Cipher Based on Two-Dimensional Chaotic Map", (in progress).

Curriculum Vitae

Publications

[1] Solak, E., Çokal, C., Cryptanalysis of a Cryptosystem Based on Discretized Two-Dimensional Chaotic Maps, Physics Letters A, vol. 372 issue. 46, pp. 6922-6924 (2008).

[2] Solak, E., Çokal, C., Algebraic Break of a Cryptosystem Based on Discretized Two-Dimensional Chaotic Maps, Physics Letters A (2009).

[3] Solak, E., Çokal, C., Comment on Encryption and Decryption of Images with Chaotic Map Lattices., Chaos, vol.18, pp. 038101, DOI:10.1063/1.2966114 (2008).

[4] Çokal, C., Solak, E., Cryptanalysis of a Chaos-Based Image Encryption Algorithm, Physics Letters A (2009).