Cryptanalyses of Some Multimedia Encryption Schemes

By

Chengqing Li

A Thesis Presented to

Graduate School of Zhejiang University

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

Thesis Supervisors: Prof. GuangChang Dong

Dr. Shujun Li

Department of Mathematics

Zhejiang University,

Hangzhou, P.R.China

May, 2005

Abstract

Since early 1990s, chaos has been widely investigated to construct multimedia encryption scheme for its good cryptography-like characteristics, such as the ergodicity, mixing and exactness property and the sensitivity to initial conditions. This thesis is concerned with the cryptanalyses of some recently-proposed chaos related multimedia encryption schemes. The security of the schemes against some familiar attack methods, such as brute-force attack, known/chosen-plaintext attack, is investigated in detail with theoretical analyses and experimental results. The main achievements are as follows:

1. Based on a normalized encryption/decryption model, from a general perspective this thesis analyzes the security of permutation-only multimedia ciphers. It is pointed out that all permutation-only image ciphers are insecure against known/chosen-plaintext attacks in the sense that only $O\left(\log_L(MN)\right)$ known/chosen plain-images are enough to break the ciphers, where MN is the size of the image and L is the number of all possible different pixel values. Also, it is found that the attack complexity is only $O(n\cdot(MN)^2)$, where n is the number of known/chosen plain-images used. A recently proposed permutation-only image cipher called hierarchical chaotic image encryption (HCIE) is served as a concretized example to show how the attack work. Experiments are shown to verify the feasibility of the known/chosen-plaintext attacks.

2. The security of a recently proposed chaos-based image encryption scheme called RCES (also called RSES) was analyzed and we found that it can be broken with only one or two known/chosen-plaintexts. In addition, the security of RCES against the brute-force attack was overestimated. Both theoretical and experimental analyses are given to show the performance of the suggested known/chosen-plaintext attacks.

3. This thesis analyzes the security of a new multistage encryption system (MES) recently proposed in ISCAS'2004. It is found that MES is insecure against a differential chosen-plaintext/ciphertext attack. Experiments are given to support the proposed attack. It is also pointed out that the security of MES against brute-force attacks is not sufficiently high.

4. This thesis analyzes the security of a new domino signal encryption algorithm (DSEA), and points out the following weaknesses: 1) its security against the brute-force attack was overestimated; 2) it is not sufficiently secure against ciphertext-only attacks, and only one ciphertext is enough to get some information about the plaintext and to break the value of a sub-key; 3) it is

insecure against known/chosen-plaintext attacks, in the sense that the secret key can be recovered from a number of continuous bytes of only one known/chosen plaintext and the corresponding ciphertext. Experimental results are given to show the performance of the proposed attacks.

5. A comprehensive analysis on the security of two-dimensional circulation encryption algorithm (TDCEA) is presented. The following security problems are found: 1) there exist some essential security defects in TDCEA; 2) two known-plaintext attacks can break TDCEA; 3) the chosen-plaintext versions of the aforementioned two known-plaintext attacks can break TDCEA even with a smaller complexity and a better performance. Some experiments are given to show the security defects of TDCEA and the feasibility of the proposed known-plaintext attacks.

6. The security of two neural-network-based encryption schemes, which are proposed by Yen et al. and Zhou et al. respectively, are analyzed in detail. It is found that the former can be easily broken by known/chosen-plaintext attacks and the latter can be broken by a chosen-plaintext attack. Experimental analyses are given to support the feasibility of the proposed attacks.

7. Some insecure properties of a VoIP encryption scheme named hierarchical data security protection (HDSP) are pointed out, which are then used to develop known/chosen-plaintext attacks. The following facts are found: 1) given n known plaintexts, only about $(50/2^n)\%$ of secret chaotic bits cannot be uniquely determined; 2) given only one specially-chosen plaintext, all secret chaotic bits can be uniquely derived; 3) the secret key can be derived with a practically small complexity even when only one plaintext is known (or chosen). Experiments are given to show the feasibility of the proposed attacks. In addition, it is found that the security of HDSP against the brute-force attack is not practically strong.

Keywords: chaos, cryptanalysis, multimedia encryption, brute-force attack, known-plaintext attack, chosen-plaintext attack

Dedication

To everyone who faces hardship courageously and optimistically.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

§1.1 Motivation

With the rapid development of multimedia and network technologies, the security of multimedia becomes more and more important, since multimedia data are transmitted over open networks more and more frequently. Typically, reliable security in storage and transmission of digital speech data, images and videos is needed in many real applications, such as pay-TV, medical imaging systems, military image databases as well as confidential video conferences. In recent years, some consumer electronic devices, such as mobile phones, have also started to provide the function of saving and exchanging digital speech/music data, images and video clips under the support of multimedia messaging services over wireless networks, which is urgently demand for multimedia security.

To meet the above needs in practice, some encryption algorithms are required to offer a sufficient level of security for different multimedia applications. Apparently, the simplest way to encrypt multimedia data is to consider the 1-D, 2-D or 3-D) multimedia bit-stream as a 1-D signal, and then to encrypt it with any available cipher [1, 2]. In some multimedia applications, such a simple way may be enough. However, in many applications, especially when digital images and videos are involved, encryption schemes considering special features of the multimedia data, such as bulky size and large redundancy in uncompressed images/videos, are still required to achieve a better overall performance and to make the integration of the encryption scheme into the whole processing procedure easier. Since the 1990s, many different algorithms have been proposed to provide solutions to image encryption [3–44], video encryption [45–62] and speech encryption [63–67]. Meanwhile, some cryptanalysis work has also been published and a number of multimedia encryption schemes have been found to be insecure from the cryptographical point of view [3, 13, 23, 66–81]. On the whole, the cryptanalytic work is still not enough now compared with cryptographic one, which is the main reason for publication of so much insecure encryption schemes.

Due to the tight relationship between chaos theory and cryptography, a great number of multimedia encryption schemes use chaos as a mechanism to realize secret permutations of digital images/frames [7, 11, 16–18, 21], or as a source to generate pseudo-random bits to control secret encryption operations [24, 27–44, 54]. As we well-known, cryptanalysis and cryptography are the two sides of cryptology which promote each other mutually. To accelerate the development of designing secure multimedia ciphers, we choose choose the security analysis (i.e., cryptanalysis) of some chaos related multimedia encryption schemes as the research topic of this thesis.

§1.2 Preliminaries of Cryptography and Cryptanalysis

To facilitate the following discussions, this section gives a brief introduction to the basic theory of the modern cryptography and cryptanalysis, which compose the technology of encryption — cryptology [1]. Simply speaking, cryptography studies how to design good (secure and fast) encryption algorithms, and cryptanalysis tries to find security weaknesses of existing algorithms and studies whether or not they are vulnerable to some attacks.

An encryption/decryption system is also called a *cipher*, or a *cryptosystem*. Accordingly, the encryption machine is called an *encipher*, and the decryption machine is called a *decipher*. The message for encryption is called the *plaintext*, and the encrypted message is called the *ciphertext*. Assuming that the plaintext and the ciphertext are denoted by P and C, respectively, the encryption procedure in a cipher can be described as $C = E_{K_e}(P)$, where K_e is the encryption key and $E(\cdot)$ is the encryption function. Similarly, the decryption procedure is $P = D_{K_d}(C)$, where K_d is the decryption key and $D(\cdot)$ is the decryption function. When $K_e = K_d$, the cipher is called a *private-key* cipher or a *symmetric* cipher. For private-key ciphers, the encryption-decryption key must be transmitted from the sender to the receiver via a separate secret channel. When $K_e \neq K_d$, the cipher is called a *public-key* cipher or an *asymmetric* cipher. For public-key ciphers, the encryption key K_e is published, and the decryption key K_d is kept private, for which no additional secret channel is needed for key transfer.

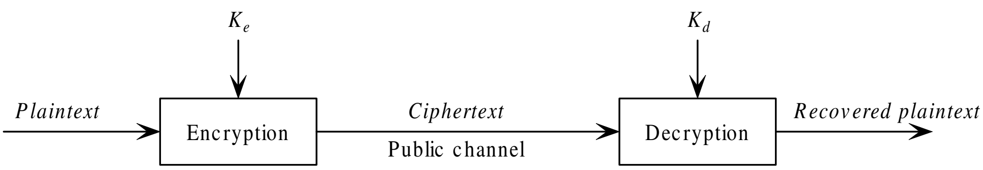


Figure 1.1: The encryption and decryption procedures of a cipher.

Following the widely-acknowledged Kerckhoffs' principle in the cryptology community [1], it is assumed that all details of the encryption/decryption algorithms are known to attackers. This means that the security of a cipher relies on the decryption key K_d only. Thus, the main task of cryptanalysis is to reconstruct the key, or its equivalent form that can successfully decrypt all or partial contents of any plaintext encrypted by the cipher.

From the cryptographical point of view, a cryptographically strong cipher should be secure enough against all kinds of attacks. For most ciphers, the following four attacks corresponding to different scenarios should be checked (from the hardest to the easiest):

- *the ciphertext-only attack* - attackers can only observe part of the ciphertexts;
- *the known-plaintext attack* - attackers can get some plaintexts and the corresponding ciphertexts;
- *the chosen-plaintext attack* - attackers can choose some plaintexts and get the corresponding ciphertexts;
- *the chosen-ciphertext attack* - attackers can choose some ciphertexts and get the corresponding plaintexts.

In the four kinds of attacks, ciphertext-only attack is the easiest and the most common attack, due to the fact that the communication channel is generally accessible for attackers. Known/chosen-plaintext attacks are possible when an attacker can temporarily access the encryption machine, or he can successfully guess the plaintexts or some segments. Chosen-ciphertext attack is possible when an attacker can have a temporary access to the decryption machine. The last three kinds of attacks, which seem to seldom occur in practice, are feasible in some real applications [1, Sec. 1.1] and have become more and more common in the digital world today. This thesis mainly focuses on known-plaintext and chosen-plaintext attacks.

§1.3 Organization of This Thesis

The main contents of this thesis is divided into 6 chapters. Chapter 2 is about cryptanalysis of permutation-only encryption algorithms. Chapter 3 discusses cryptanalysis of four chaos-based schemes. Chapter 4 is about cryptanalysis of two neural-network-based encryption schemes. Chapter 5 is about cryptanalysis of a data security protection scheme for VoIP. Chapter 6 gives the conclusion and future works. A relative detailed introduction of all chapters are as follows.

Chapter 2 focuses a general cryptanalysis of permutation-only multimedia encryption algorithms with two typical examples. Based on a normalized encryption/decryption model, from a general perspective this chapter first provides a quantitative security analysis of permutation-only image ciphers working in the spatial domain. Then two typical encryption algorithms, HCIE and TDCEA, are presented to show how the attacks work. Finally, the cryptanalysis result is generalized to permutation-only image ciphers working in the frequency domain, as well as video ciphers and speech ciphers.

Chapter 3 discusses how to break four chaos-based encryption schemes with a similar method. Three schemes, RCES, MES and DSEA, are introduced first. Then the special security properties of the three schemes are analyzed respectively. The fourth scheme, TDCEA is also cryptanalyzed with a point of view different from that used in Chap. 2. Both theoretical and experimental analyses are given to show the performance of the proposed attacks.

Chapter 4 analyzes the security properties of two neural-network-based encryption schemes in detail. At first, the two schemes, proposed by Yen et al. and Zhou et al. respectively, are introduced briefly. Then the analysis shows that the two schemes can be successfully broken with one known/chosen-plaintext attack and two chosen-plaintexts attacks respectively. The theoretical analyses are supported by the experimental results.

Chapter 5 shows a new hierarchical data security protection (HDSP) scheme for VoIP technique is very weak against brute-force attack and known/chosen-plaintext attacks. After the HDSP scheme is introduced, the detailed cryptanalysis is given. It is found that the security against the brute-force attack is not practically strong, and that it can be broken with some known-plaintexts and/or one chosen-plaintext, due to some insecure properties of HDSP. Experiments are given to show the feasibility of the proposed attacks. Finally, some countermeasures are suggested for enhancing the security of HDSP.

Chapter 6 draws some lessons for designing securer multimedia encryption scheme, summaries our works in this thesis and gives some remarks on future research.

Chapter 2
Cryptanalyses of Permutation-Only Encryption Algorithms

§2.1 Introduction

In image encryption, secret permutations are widely used to shuffle the positions of pixels (and/or pixel bits) [4, 7, 15], which is an effective and easy way to make the cipher-image look "chaotic". Similarly, in video encryption, secret permutations are widely used to shuffle the DCT/wavelet coefficients, blocks or macroblocks [13, 25, 47, 51, 53, 60, 61]. The same idea has also been used in speech data encryption, by permuting the samples within each frame[65–67]. There are many image/video/speech encryption algorithms that are based only on secret permutations [4, 5, 7, 8, 10, 12, 13, 13, 15, 27, 28, 30, 33, 38, 43, 47, 50, 51, 53, 65–67], which are called *permutation-only* (image/video/speech) ciphers in this thesis. Note that some ciphers can be formalized as permutation-only ciphers, even though some other encryption techniques are used together with secret permutations. As typical examples, the video ciphers proposed in [25, 60, 61] become permutation-only ciphers, if the sign bits of all encrypted data elements are neglected. The main advantages of using only secret permutations in a cipher include easy implementation and the universality for most multimedia data formats.

In most *permutation-only* ciphers, the security is analyzed only for ciphertext-only attacks, i.e., brute-force attacks of exhaustively searching the secret key. However, from the cryptographical point of view, such a security analysis is not enough, since there exist other more powerful attacks, such as known/chosen-plaintext attacks and chosen-ciphertext attacks. In fact, it has been pointed out that permutation-only multimedia ciphers are not secure against known/chosen-plaintext attacks [3, 13, 23, 66, 67, 70–77], but most previous cryptanalysis results are proposed for some specific permutation-only image/video ciphers and the essential security defects of these encryption algorithms have not been quantitatively clarified in a general way. The lack of a general cryptanalysis makes it ambiguous in understanding whether or not the security of permutation-only multimedia encryption algorithms can be effectively enhanced by designing new methods to generate better secret permutations.

This chapter gives a general cryptanalysis of most (if not all) permutation-only multimedia encryption algorithms, mainly focusing on the quantitative relation between the breaking performance and the number of required known/chosen plaintexts, as well as the estimation of the attack complexity. It will be pointed out that secret permutations alone cannot provide sufficient security against known/chosen-plaintext attacks, from both theoretical and experimental points of view. The cryptanalysis is performed on a general model of permutation-only image ciphers working in spatial domain, which then is generalized to permutation-only image working in frequency domain and also to permutation-only video/speech ciphers. As a typical example of permutation-only image ciphers, a recently-proposed image encryption scheme called HCIE [27, 30, 33]* is investigated in detail, to show how the known/chosen-plaintext attacks work. It is shown that all permutation-only image ciphers are not secure against known/chosen-plaintext attacks, in the sense that only $O\left(\log_L(MN)\right)$ known/chosen plain-images are enough to break the ciphers, where MN is the size of the image (i.e., the number of pixels) and L is the number of different pixel values. An upper bound of the attack complexity is also derived to be $O(n \cdot (MN)^2)$, where n is the number of known/chosen plain-images. What's more, it is found that the hierarchical encryption structure suggested in HCIE cannot provide any higher security against known/chosen-plaintext attacks, but actually make the security weaker. As a conclusion, secure permutations must be used together with other encryption mechanisms to design a secure multimedia encryption scheme, as in some compound image/video ciphers [9, 16–18, 21, 48].

Recently, a new signal security system called TDCEA was proposed for real-time multimedia data transmission in [38, 43]. In fact, TDCEA is an enhanced version of a previous image encryption scheme proposed by the same authors in [29, 32], named BRIE (bit recirculation image encryption), which is the one-dimensional counterpart of TDCEA. The original BRIE scheme has been successfully cryptanalyzed in [81], showing its insecurity against known/chosen-plaintext attacks. Essentially, TDCEA is secret permutations of all 64 bits of each 8-pixel block, so TDCEA can be easily broken by a general method proposed in this chapter (In §3.6, we will proposed another method to break it). In addition, some special security flaws are also discussed in detail.

This chapter is organized as follows. §2.2 gives cryptanalysis on common permutation-only image ciphers working in spatial domain. §2.3 presents two typical permutation-only encryption algorithm, HCIE and TDCEA. §2.4 and §2.5 discuss cryptanalyses of HCIE and TDCEA, respectively. §2.6 discusses the generalization of the cryptanalysis results to permutation-only image ciphers working in frequency domain and permutation-only video/speech ciphers. The last section concludes the chapter.

*The chaotic image encryption (CIE) scheme proposed in [28] is an initial version of HCIE.

§2.2 Cryptanalysis of Permutation-Only Encryption Algorithms Working in Spatial Domain

§2.2.1 A Normalized Model for Encryption and Decryption

When working in spatial domain, just as its name implies, *permutation-only* image ciphers encrypt images by permuting the positions of all pixels in a secret way. The secret permutations have to be invertible to make the decryption possible. This means that all permutation-only ciphers belong to symmetry ciphers, i.e., $K_e = K_d = K$, which is used to generate the secret permutations. Although many different methods have been proposed to realize secret key-dependent pixel permutations, for a given plain-image of size $M \times N$ ("height×width"), a permutation-only image cipher can be normalized with an *invertible key-dependent permutation matrix* of size $M \times N$, denoted by

$$\boldsymbol{W} = [w(i,j) = (i',j') \in \mathbb{M} \times \mathbb{N}]_{M\times N}\,, \tag{2.1}$$

where $\mathbb{M} = \{0, \cdots, M-1\}$ and $\mathbb{N} = \{0, \cdots, N-1\}$. With the permutation matrix $\boldsymbol{W}$ and its inverse $\boldsymbol{W}^{-1} = [w^{-1}(i,j)]_{M\times N}$, for a plain-image $f = [f(i,j)]_{M\times N}$ and its corresponding cipher-image $f' = [f'(i,j)]_{M\times N}$, the encryption and decryption procedures of a permutation-only image cipher can always be described as follows:

- *the encryption procedure*: for $i = 0 \sim (M-1)$ and $j = 0 \sim (N-1)$, $f'(w(i,j)) = f(i,j)$;
- *the decryption procedure*: for $i = 0 \sim (M-1)$ and $j = 0 \sim (N-1)$, $f(w^{-1}(i,j)) = f'(i,j)$.

In a short form, one can express the encryption procedure as $f'(\boldsymbol{W}(\boldsymbol{I})) = f(\boldsymbol{I})$ and the decryption procedure as $f(\boldsymbol{W}^{-1}(\boldsymbol{I})) = f'(\boldsymbol{I})$, where

$$\boldsymbol{I} = \begin{bmatrix} (0,0) & \cdots & (0,N-1) \\ \vdots & \ddots & \vdots \\ (M-1,0) & \cdots & (M-1,N-1) \end{bmatrix}_{M\times N}.$$

To ensure the invertibility of the permutation matrix, i.e., to make the decryption possible, the following point should be satisfied: $\forall (i_1,j_1) \neq (i_2,j_2)$, $w(i_1,j_1) \neq w(i_2,j_2)$. This means that $\boldsymbol{W}$ determines a bijective (i.e., one-to-one) permutation mapping, $F_{\boldsymbol{W}} : \mathbb{M} \times \mathbb{N} \to \mathbb{M} \times \mathbb{N}$.

From the above description, one can see that the design of a permutation-only image cipher focuses on two points: 1) what the secret key K is; 2) how the permutation matrix $\boldsymbol{W}$ and $\boldsymbol{W}^{-1}$ are derived from the secret key K. Generally speaking, each key defines a permutation matrix, and each permutation-only image cipher defines a finite set containing a number of permutation matrices selected from $(MN)!$ possible permutation matrices. In literature, many different methods have been proposed to derive a permutation matrix from a key, some of which are listed as follows:

- *SCAN language* based methods [4, 5, 7–9, 48]: define some different scan patterns of the 2-D image and combine these patterns to define a permutation matrix by scanning the whole image pixel by pixel;
- *quadtree* based methods [5, 8]: divide the image into multi-level quadtree and shuffle the order of four nodes in each level to realize a permutation matrix;
- *2-D chaotic maps* based methods [16–18, 21]: iterate a discretized 2-D chaotic map over the $M \times N$ image lattice for many times to realize a permutation matrix;
- *Fractal curves* based methods [15, 50]: use a fractal(-like) curve to replace the normal scan order to realize a permutation matrix;
- *pseudo-random rotations* based methods [27, 28, 30, 33]: pseudo-randomly rotate pixels along some straight lines for many times to realize a permutation matrix;
- *matrix transformation based methods* [10]: use (integer) transformations of matrix, such as n-dimensional Arnold transformation and Fabonacci-Q transformation, to define permutation matrices;
- *composite methods* [12]: combine different methods to realize more complicated permutation matrices.

Although different types of secret keys are used in different permutation-only image ciphers to generate the permutation matrix, it is reasonable to consider the permutation matrix $\boldsymbol{W}$ itself as the equivalent encryption key and $\boldsymbol{W}^{-1}$ as the equivalent decryption key. From such a point of view, all permutation-only image ciphers can be considered the same. This is the base for the security analysis to be carried out below in this chapter.

§2.2.2 The Known-Plaintext Attack

As shown above, when a *permutation-only* image cipher is used to encrypt images in spatial domain, a pixel at the position (i, j) will be secretly permuted to another fixed position (i', j') while the pixel value is unchanged. Therefore, by comparing a number of known plain-images and the corresponding cipher-images, it is possible for an attacker to (partially or even totally) reconstruct the secret permutations of all pixels, i.e., to derive the encryption/decryption keys – the permutation matrix $\boldsymbol{W}$ and its inverse $\boldsymbol{W}^{-1}$.

Given n known plain-images $f_1 \sim f_n$ and their cipher-images $f'_1 \sim f'_n$, the deduction procedure of $\boldsymbol{W}$ and $\boldsymbol{W}^{-1}$ can be shown in the following `Get_Permutation_Matrix` function. With the input parameters $(f_1 \sim f_n, f'_1 \sim f'_n, M, N)$, this function returns an estimation of the permutation matrix $\boldsymbol{W}$ and its inverse $\boldsymbol{W}^{-1}$. Assuming the value of each pixel ranges in $\{0, \cdots, L-1\}$, the `Get_Permutation_Matrix` function is described as follows.

- *Step 1: compare pixel values within the* n *cipher-images* $f'_1 \sim f'_n$ *to get* $(n \cdot L)$ *sets of pixel positions*:

  $$\Lambda'_1(0) \sim \Lambda'_1(L-1), \cdots, \Lambda'_n(0) \sim \Lambda'_n(L-1),$$

  where $\Lambda'_m(l) \subseteq \mathbb{M} \times \mathbb{N}$ denotes a set containing positions of all pixels in f'_m $(m = 1 \sim n)$ whose values are equal to $l \in \{0, \cdots, L-1\}$, i.e., $\forall (i', j') \in \Lambda'_m(l)$, $f'_m(i', j') = l$. Note that $\Lambda'_m(0) \sim \Lambda'_m(L-1)$ actually compose a partition of the set of all pixel positions: $\bigcup_{l=0}^{L-1} \Lambda'_m(l) = \mathbb{M} \times \mathbb{N} = \{(0,0), \cdots, (M-1, N-1)\}$, and $\forall l_1 \neq l_2$, $\Lambda'_m(l_1) \cap \Lambda'_m(l_2) = \varnothing$;

- *Step 2: get a multi-valued permutation matrix,* $\widehat{\boldsymbol{W}} = [\widehat{\boldsymbol{w}}(i,j)]_{M \times N}$, where $\widehat{\boldsymbol{w}}(i,j) = \bigcap_{m=1}^{n} \Lambda'_m(f_m(i,j))$. Here, note that $\widehat{\boldsymbol{w}}(0,0) \sim \widehat{\boldsymbol{w}}(M-1, N-1)$ actually composes a new partition of the position set $\mathbb{M} \times \mathbb{N}$;

- *Step 3: determine a single-valued permutation matrix,* $\widetilde{\boldsymbol{W}} = [\widetilde{w}(i,j)]_{M \times N}$ *from* $\widehat{\boldsymbol{W}}$, where $\widetilde{w}(i,j) \in \widehat{\boldsymbol{w}}(i,j)$ and $\forall (i_1, j_1) \neq (i_2, j_2)$, $\widetilde{w}(i_1, j_1) \neq \widetilde{w}(i_2, j_2)$;

- *Step 4: output* $\widetilde{\boldsymbol{W}}$ *and its inverse* $\widetilde{\boldsymbol{W}}^{-1} = [\widetilde{w}^{-1}(i,j)]_{M \times N}$ *as the estimations of* $\boldsymbol{W}$ *and* $\boldsymbol{W}^{-1}$.

Apparently, if and only if $\#(\widehat{\boldsymbol{w}}(0,0)) = \cdots = \#(\widehat{\boldsymbol{w}}(S_M - 1, S_N - 1)) = 1$, i.e., each element of $\widehat{\boldsymbol{W}}$ contains only one pixel position, it is true that $\widetilde{\boldsymbol{W}} = \boldsymbol{W}$ and the cipher is totally broken. However, because some elements of $\widehat{\boldsymbol{W}}$ contain more than one pixel position, generally $\widetilde{\boldsymbol{W}}$ is not an exact estimation of $\boldsymbol{W}$. Assume that there are $(\widehat{N} \leq MN)$ different elements in $\widehat{\boldsymbol{W}}$, and that the $\widehat{N}$ different elements are $\widehat{\boldsymbol{w}}_1 \sim \widehat{\boldsymbol{w}}_{\widehat{N}}$. Then, it can be easily verified that there are $\prod_{k=1}^{\widehat{N}} \#(\widehat{\boldsymbol{w}}_k)!$ possibilities of $\widetilde{\boldsymbol{W}}$. To make the estimation of $\widetilde{\boldsymbol{W}}$ as accurate as possible, some specific optimization algorithms can be used to choose a better position from $\widehat{\boldsymbol{w}}(i,j)$ as the value of $\widetilde{w}(i,j)$, such as genetic and simulated annealing algorithms. Our experiments show that even a simple algorithm can achieve a rather good estimation when $n \geq 3$ for 256×256 gray-scale images. The simple algorithm is called "taking-the-first" algorithm, which sets $\widetilde{w}(i,j)$ to be the first available element in $\widehat{\boldsymbol{w}}(i,j)$, where the term "available" refers to the constraint that $\forall (i_1,j_1) \neq (i_2,j_2)$, $\widetilde{w}(i_1,j_1) \neq \widetilde{w}(i_2,j_2)$.

Now, let us consider the decryption performance of the estimated permutation matrix $\widetilde{\boldsymbol{W}}$ when $\widetilde{\boldsymbol{W}} \neq \boldsymbol{W}$. Generally speaking, due to the large information redundancy existing in a digital image, only partially-recovered pixels are enough to reveal most visual information. Therefore, if there are enough correct elements in $\widetilde{\boldsymbol{W}}$, the decryption performance may be acceptable from a practical point of view. From the above discussions, one can see that correctly-recovered elements in $\widetilde{\boldsymbol{W}}$ belong to two different classes:

- *the absolutely correct elements*: derived from the single-valued elements of $\widehat{\boldsymbol{W}}$;
- *the probabilistically correct elements*: derived from the multi-valued elements of $\widehat{\boldsymbol{W}}$, and are correctly guessed by an optimization algorithm of selecting a proper position from each $\widehat{\boldsymbol{w}}(i,j)$.

Assuming that the number of single-valued elements of $\widehat{\boldsymbol{W}}$ is n_c and the successful probability of the optimization algorithm is p_s, the average number of correct elements in $\widetilde{\boldsymbol{W}}$ will be $n_c + p_s \cdot (MN - n_c)$. Because p_s is generally not fixed (tightly dependent on the employed optimization algorithm), only the absolutely correct elements are considered here (i.e., $p_s = 0$ is assumed) to perform a qualitative analysis. Now the problem of correct elements in $\widetilde{\boldsymbol{W}}$ is simplified to be the problem of singe-value elements in $\widehat{\boldsymbol{W}}$. Observing the `Get_Permutation_Matrix` function, one can see that the cardinality of $\widehat{\boldsymbol{w}}(i,j)$ is uniquely determined by $\Lambda'_1(f_1(i,j)) \sim \Lambda'_n(f_n(i,j))$. To further simplify the analysis, assume that the value of each pixel distributes uniformly in $\{0, \cdots, L-1\}$, and that the values of any two pixels (within the same image or in two different cipher-images*) are independent of each other. Then, one can consider the following two types of positions in $\widehat{\boldsymbol{w}}(i,j)$:

- *the only one real position* $w(i,j)$, which absolutely occurs in $\widehat{\boldsymbol{w}}(i,j)$;
- *other fake positions*, each of which occurs in each $\Lambda'_m(f_m(i,j))$ with a probability of $\frac{1}{L}$, i.e., each of which occurs in all the n sets, $\Lambda'_1(f_1(i,j)) \sim \Lambda'_n(f_n(i,j))$, with a probability of $\frac{1}{L^n}$.

*Note that a plain-image and its cipher-image are totally related via the secret permutation matrix.

Based on the above results, one can qualitatively deduce that the average cardinality of $\widehat{\boldsymbol{w}}(i,j)$ is $\overline{\#\left(\widehat{\boldsymbol{w}}(i,j)\right)} = \left(1 + \frac{MN-1}{L^n}\right)$, which approaches 1 exponentially as n increases. Generally speaking, when $1 + \frac{MN-1}{L^n} < 1.5$, i.e., about half elements in $\widetilde{\boldsymbol{W}}$ are correct, the decryption performance will be acceptable. Solving this inequality, one has

$$n \geq \lceil \log_L(2(MN-1)) \rceil. \tag{2.2}$$

As an example, for 256×256 gray-scale images, $M = N = L = 256$, one has $n \geq \lceil \log_L(2(MN-1)) \rceil = \lceil 2.125 \rceil = 3$. The average cardinality is about 1.0039 when $n = 3$, so it is expected that the decryption performance for $n \geq 3$ will be rather good, which is verified by the experiments given in §2.4.4. Here, note that the actual decryption performance is generally better than the above theoretical expectation for the following two reasons:

- human eyes have a powerful capability of suppressing image noises and extracting significant features: 10% noisy pixels cannot make much influence on the visual quality of a digital image, and it only needs 50% of pixels to reveal most visual information of the original image;
- due to the short-distance and long-distance correlations in natural images, two pixel values are close to each other with a non-negligible probability larger than the average probability; as a result, the wrongly-decrypted pixel are close to the right value with a probability larger than the average probability.

The second point implies that the decryption performance of natural images will be better than the performance of noise-like images, from the point of view of decryption error ratio. For experimental verification and more explanations, see §2.4.4, Figs. 2.3 and 2.4.

Next, let us consider the time complexity of the above-discussed known-plaintext attack, i.e., the time complexity of the `Get_Permutation_Matrix` function. Note that the time complexity depends on the implementation details of this function. This chapter only gives a conservative estimation, i.e., an upper bound, of the time complexity. The time complexity of each step is as follows:

- *Step 1*: The L sets of each cipher-image f'_l are obtained by scanning f'_l once: for $i = 0 \sim (M-1)$ and $j = 0 \sim (N-1)$, add (i,j) into the set $\Lambda'_m(f'_l(i,j))$. Thus, the time complexity of this step is $O(n \cdot MN)$.
- *Step 2*: Specially, assume all cipher-pixels satisfy uniform distributions. Then, the average cardinality of $\Lambda_m(l)$ is $\frac{MN}{L}$ and an upper bound of the time

complexity of this step is $MN\cdot\left(\frac{MN}{L}\cdot\left(\frac{1}{2}\cdot\frac{MN}{L}\right)^{n-1}\right)=2MN\cdot\left(\frac{MN}{2L}\right)^{n}$, which exponentially increase as n increases if $MN>2L$. However, in practice, the real complexity is much smaller due to the optimization of the calculation process. Here, we consider the so-called halving algorithm, which calculates the intersection of n sets $A_1\sim A_n$ by dividing them into multi-level groups of $(2,4,\cdots,2^i,\cdots)$ sets. For example, when $n=11$, the calculation process is described by

$$\begin{aligned}((A_1 \overset{1}{\cap} A_2) \overset{3}{\cap} (A_3 \overset{2}{\cap} A_4)) \overset{7}{\cap} ((A_5 \overset{4}{\cap} A_6) \overset{6}{\cap} (A_7 \overset{5}{\cap} A_8))\\ \overset{10}{\cap}((A_9 \overset{8}{\cap} A_{10}) \overset{9}{\cap} A_{11}),\end{aligned} \tag{2.3}$$

where $\overset{i}{\cap}$ denotes the i-th intersection operation. The goal of this halving algorithm is to minimize the cardinalities of the two sets involved in each intersection operation so as to reduce the global complexity. To make the estimation of the complexity easier, let us consider the case of $n=2^d$, where d is an integer. In this case, the overall complexity is shown in Eq. (2.4). As two typical examples, when $M=N=256$ and $L=2$ (monotonic images), the complexity is about $\left(2^{29.2}\cdot n\right)$; when $M=N=256$ and $L=256$ (gray-scale images), the complexity is only $\left(2^{15}\cdot n\right)$. One can see that now the complexity is always much smaller than $2MN\cdot\left(\frac{MN}{L}\right)^{n}$. When n is not a power of 2, the complexity will be smaller than $\frac{2^{\lceil\log_2 n\rceil}}{2L^2-1}\cdot(MN)^2\leq\frac{2n}{2L^2-1}\cdot(MN)^2$.

$$\begin{aligned}\sum_{k=d-1}^{0}2^k\cdot\left(\frac{MN}{L^{d-k}}\right)^2 &= \sum_{k'=1}^{d}2^{d-k'}\cdot\left(\frac{MN}{L^{k'}}\right)^2=2^d\cdot(MN)^2\cdot\left(\sum_{k'=0}^{d}\frac{1}{(2\cdot L^2)^{k'}}-1\right)\\ &= n\cdot(MN)^2\cdot\left(\frac{1-((2L^2)^{-1})^{d+1}}{1-(2L^2)^{-1}}-1\right)\\ &< n\cdot(MN)^2\cdot\left(\frac{1}{1-(2L^2)^{-1}}-1\right)=\frac{n}{2L^2-1}\cdot(MN)^2.\end{aligned} \tag{2.4}$$

- *Step 3*: The time complexity of this step is determined by the details of the involved optimization algorithm. For the "taking-the-first" algorithm, the complexity is $MN\cdot\left(1+\frac{MN-1}{L^n}\right)\approx MN+\frac{(MN)^2}{L^n}$.
- *Step 4*: The time complexity of this step is MN.

Combining the above discussions, the final time complexity of the `Get_Permutation_Matrix` function is always of order $n\cdot(MN)^2$, which is practically small even for a PC.

From the above analysis, one can see that the time complexity is mainly determined by Step 2. When the "taking-the-first" algorithm is adopted in the

`Get_Permutation_Matrix` function, Step 2 can be skipped so that the total complexity will still be of order $O\left(n \cdot (MN)^2\right)$, even without using the halving algorithm to calculate the intersections. In this case, Step 3 can be described as follows:

- *Step 3'*: For $i = 0 \sim (M-1)$ and $j = 0 \sim (N-1)$, do the following operations:
  - *Step 3'a*: find the first element satisfying $f_1(i,j) = f_1'(i',j'), \cdots, f_n(i,j) = f_n'(i',j')$ by searching each element in $\Lambda_1'(f_1(i,j))$ and checking whether it occurs in $\Lambda_2'(f_2(i,j)) \sim \Lambda_n'(f_n(i,j))$;
  - *Step 3'b*: set $\widetilde{w}(i,j) = (i',j')$ and then delete (i',j') from $\Lambda_1'(f_1(i,j)) \sim \Lambda_n'(f_m(i,j))$.

It is obvious that the time complexity of Step 3'a is always less than $n \cdot (MN)$ and averagely is $O\left(n \cdot \frac{MN}{L}\right)$, so the time complexity of Step 3' is always less than $n \cdot (MN)^2$ and averagely is $O\left(n \cdot \frac{(MN)^2}{L}\right)$.

§2.2.3 The Chosen-Plaintext Attack

The chosen-plaintext attack works in the same way as the known-plaintext attack, but the plain-images can be deliberately chosen to optimize the estimation of $\widetilde{\boldsymbol{W}}$ (i.e., to maximize the decryption performance). The following two rules are useful in the creation of the n chosen plain-images $f_1 \sim f_n$:

- the histogram of each chosen plain-image should be as uniform as possible;
- the i-dimensional ($2 \leq i \leq n$) histogram of any i chosen plain-images should be as uniform as possible, which is a generalization of the above rule.

The goal of the above two rules is to minimize the average cardinality of the elements in $\widehat{\boldsymbol{W}}$, and then to maximize the number of correct elements in the estimated permutation matrix $\widetilde{\boldsymbol{W}}$.

As an example of the two rules, consider the condition when $M = N = L = 256$ (256-valued gray-scale images of size 256×256). In this case, the following two chosen plain-images are enough to ensure a perfect estimation of the permutation matrix $\boldsymbol{W}$: $f_1 = [f_1(i,j) = i]_{256\times 256}$ and $f_2 = [f_2(i,j) = j]_{256\times 256}$, i.e.,

$$f_1 = f_2^T = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ i & \cdots & i \\ \vdots & \ddots & \vdots \\ 255 & \cdots & 255 \end{bmatrix}_{256\times 256} \tag{2.6}$$

and

$$f_2 = f_1^T = \begin{bmatrix} 0 & \cdots & j & \cdots & 255 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & j & \cdots & 255 \end{bmatrix}_{256\times 256} . \tag{2.7}$$

For the above two chosen plain-images, it is true that $\forall(i_1, j_1) \neq (i_2, j_2)$, $(f_1(i_1, j_1), f_1(i_2, j_2)) \neq (f_2(i_1, j_1), f_2(i_2, j_2))$. This can ensure that $\forall l_1, l_2 \in \{0, \cdots, L-1\}$, $\#\left(\Lambda_1'(l_1) \cap \Lambda_2'(l_2)\right) = 1$. For n images satisfying this constraint, we say that they compose *an orthogonal image set*. This concept is introduced to facilitate the following discussion on the chosen-plaintext attack to HCIE.

In general cases, it can be easily deduced that $n = \lceil \log_L(MN) \rceil$ orthogonal images* have to be created to carry out a successful chosen-plaintext attack. Apparently, it will never be larger than $\lceil \log_L(2(MN-1)) \rceil$ – the number of required plain-images in the known-plaintext attack with a good breaking performance (recall the above subsection). This means the chosen-plaintext attack is a little (but not so much) stronger than the chosen-plaintext attack in the present case of discussion.

§2.3 Two Permutation-Only Encryption Algorithms

§2.3.1 Hierarchical Chaotic Image Encryption (HCIE)

HCIE is a two-level hierarchical permutation-only image cipher, and all involved permutation matrices are defined by pseudo-random combinations of four rotation mappings with pseudo-random parameters. For an image, $f = [f(i,j)]_{M\times N}$, the four mapping operations are described as follows, where $p < \min(M, N)$ holds for each mapping.

*When $MN \leq L$, only one chosen plain-image is enough, if each pixel value is different from the others.

Definition 2.1: *The mapping* $f' = ROLR_b^{i,p}(f)$ $(0 \le i \le M-1)$ *is defined to rotate the i-th row of* f*, in the left (when* $b=0$*) or right (when* $b=1$*) direction by p pixels.*

Definition 2.2: *The mapping* $f' = ROUD_b^{j,p}(f)$ $(0 \le j \le N-1)$ *is defined to rotate the j-th column of* f*, in the up (when* $b=0$*) or down (when* $b=1$*) direction by p pixels.*

Definition 2.3: *The mapping* $f' = ROUR_b^{k,p}(f)$ $(0 \le k \le M+N-2)$ *is defined to rotate all pixels satisfying* $i+j=k$*, in the lower-left (when* $b=0$*) or upper-right (when* $b=1$*) direction by p pixels.*

Definition 2.4: *The mapping* $f' = ROUL_b^{l,p}(f)$ $(1-N \le l \le M-1)$ *is defined to rotate all pixels satisfying* $i-j=l$*, in the upper-left (when* $b=0$*) or lower-right (when* $b=1$*) direction by p pixels.*

Given a pseudo-random bit sequence $\{b(i)\}$ starting from i_0, the following `Sub_HCIE` function is used to permute an $S_M \times S_N$ image f_{sub} to be another $S_M \times S_N$ image f'_{sub}, where $(\alpha, \beta, \gamma, no)$ are control parameters. Note that all codes in this chapter is described in C-language style.

$$
\begin{array}{l}
\text{for } (ite=0;\ ite<no;\ ite++)\ \{ \\
\quad q = i_0 + (3S_M + 3S_N - 2)\cdot ite; \\
\quad p = \alpha + \beta\cdot b(q+0) + \gamma\cdot b(q+1); \\
\quad \text{for } (i=0;\ i\le (S_M-1);\ i++) \\
\quad\quad f'_{sub} = ROLR_{b(i+q)}^{i,p}(f_{sub}); \\
\quad \text{for } (j=0;\ j\le (S_N-1);\ j++) \\
\quad\quad f'_{sub} = ROUD_{b(j+q+S_M)}^{j,p}(f'_{sub}); \\
\quad \text{for } (k=0;\ k\le (S_M+S_N-2);\ k++) \\
\quad\quad f'_{sub} = ROUR_{b(k+q+S_M+S_N)}^{k,p}(f'_{sub}); \\
\quad \text{for } (l=(1-S_N);\ l\le (S_M-1);\ l++) \\
\quad\quad f'_{sub} = ROUL_{b(l+q+2\cdot S_M+3\cdot S_N-2)}^{l,p}(f'_{sub}); \\
\} \\
i_0 = i_0 + (3S_M + 3S_N - 2)\cdot no;
\end{array}
$$

One can see that the above `Sub_HCIE` function actually defines an $S_M \times S_N$ permutation matrix pseudo-randomly controlled by $(3S_M + 3S_N - 2)\cdot no$ bits in the bit sequence $\{b(i)\}$ from i_0. Based on this function, for an $M \times N$ image $f = [f(i,j)]_{M\times N}$, the encryption procedure of HCIE can be briefly described in two levels.

- *The secret key* is the initial condition $x(0)$ and the control parameter μ of the following chaotic Logistic map[82]:

$$x(k+1) = \mu \cdot x(k) \cdot (1 - x(k)), \tag{2.8}$$

which is realized in L-bit finite precision.

- *Some public parameters*: S_M, S_N, α, β, γ and no, where $\sqrt{M} \leq S_M \leq M$, $M \bmod S_M = 0$, $\sqrt{N} \leq S_N \leq N$, and $N \bmod S_N = 0$.

  Note: *Although $(S_M, S_N, \alpha, \beta, \gamma, no)$ can be all included in the secret key, they are not suitable for such a use due to the following reasons: 1) S_M, S_N are related to M, N; 2) α, β, γ are related to S_M, S_N (and then related to M, N, too); 3) S_M, S_N can be easily guessed from the mosaic effect of the cipher-image; 4) no cannot be too large to achieve an acceptable encryption speed.*

- *The initialization procedure* of generating the bit sequence used in the `Sub_HCIE` function: run the Logistic map from $x(0)$ to generate a chaotic sequence $\{x(i)\}_{i=0}^{\lceil L_b/8 \rceil - 1}$, and then extract the 8 bits following the decimal point of each chaotic state $x(i)$ to yield a pseudo-random binary sequence (PRBS) $\{b(i)\}_{i=0}^{L_b-1}$, where $L_b = \left(1 + \frac{M}{S_M} \cdot \frac{N}{S_N}\right) \cdot (3S_M + 3S_N - 2) \cdot no$; finally, set $i_0 = 0$ to let the `Sub_HCIE` function run from $b(0)$.

- *The two-level hierarchical encryption (permutation) procedure*:

  1. *The high-level encryption – permuting image blocks*: divide the plain-image f into $S_M \times S_N$ blocks, which compose an $\frac{M}{S_M} \times \frac{N}{S_N}$ block-image P_f as follows

$$P_f = [P_f(i,j)]_{\frac{M}{S_M} \times \frac{N}{S_N}} = \begin{bmatrix} P_f(0,0) & \cdots & P_f\left(0, \frac{N}{S_N} - 1\right) \\ \vdots & \ddots & \vdots \\ P_f\left(\frac{M}{S_M} - 1, 0\right) & \cdots & P_f\left(\frac{M}{S_M} - 1, \frac{N}{S_N} - 1\right) \end{bmatrix}_{\frac{M}{S_M} \times \frac{N}{S_N}}, \tag{2.9}$$

  where

  $P_f(i,j) =$

$$\begin{bmatrix} f(i \cdot S_M, j \cdot S_N) & \cdots & f(i \cdot S_M, j \cdot S_N + (S_N - 1)) \\ \vdots & \ddots & \vdots \\ f((i+1) \cdot S_M - 1, j \cdot S_N) & \cdots & f((i+1) \cdot S_M - 1, j \cdot S_N + (S_N - 1)) \end{bmatrix}_{S_M \times S_N}, \tag{2.10}$$

  and then permute the positions of all blocks with the `Sub_HCIE` function in the following way:

(a) create a pseudo-image $f_p = [f_p(i,j)]_{S_M \times S_N}$ as follows:
$f_p = [f_p(i,j)]_{S_M \times S_N} =$

$$\left[\begin{array}{cccc:c} 1 & \cdots & \frac{N}{S_N} & \\ \frac{N}{S_N}+1 & \cdots & 2\cdot\frac{N}{S_N} & \\ \vdots & \ddots & \vdots & \mathbf{0} \\ \left(\frac{M}{S_M}-1\right)\cdot\frac{N}{S_N}+1 & \cdots & \frac{M}{S_M}\cdot\frac{N}{S_N} & \\ \hdashline & \mathbf{0} & & & \mathbf{0} \end{array}\right]_{S_M \times S_N}, \tag{2.11}$$

where the $\frac{M}{S_M}\cdot\frac{N}{S_N}$ non-zero elements of f_p means the 1-based indices of all image blocks, and permute f_p with the `Sub_HCIE` function to get a shuffled pseudo-image f_p^*;

(b) generate a permuted block-image P_{f^*} from P_f (i.e., permute f blockwise) using the shuffled indices contained in f_p^*:

$order = 0$;
for $(i = 0;\ i \le (S_M - 1);\ i{+}{+})$
  for $(j = 0;\ j \le (S_N - 1);\ j{+}{+})$
    if $(f_p^*(i,j) \neq 0)$ {
      $f_p^*(i,j){-}{-}$; // 1-based index $\Rightarrow$ 0-based one
      $i^* = \left\lfloor \frac{order}{N/S_N} \right\rfloor$, $j^* = order \bmod (N/S_N)$;
      $ii = \left\lfloor \frac{f_p^*(i,j)}{N/S_N} \right\rfloor$, $jj = f_p^*(i,j) \bmod (N/S_N)$;
      $P_{f^*}(i^*, j^*) = P_f(ii, jj)$;
      $order{+}{+}$;
    }

The above high-level encryption procedure can be considered as the permutation of the block-image: $P_f \xrightarrow{f_p^* = \texttt{Sub\_HCIE}(f_p)} P_{f^*}$, where f_p^* actually is an $\frac{M}{S_M} \times \frac{N}{S_N}$ permutation matrix.

2. *The low-level encryption – permuting pixels in each image block*: for $i = 0 \sim \left(\frac{M}{S_M} - 1\right)$ and $j = 0 \sim \left(\frac{N}{S_N} - 1\right)$, call the `Sub_HCIE` function to permute each block $P_{f^*}(i,j)$ to get the corresponding block of the cipher-image f': $P_{f'}(i,j) = \texttt{Sub\_HCIE}\left(P_{f^*}(i,j)\right)$.

As described in §2.2.1, for permutation-only encryption algorithm, decryption procedure is just like encryption one beside replacing permutation matrix with its inverse. For HCIE, the inverse of permutation matrix can be easily generated by reversing orders of some parameters and procedures of function `Sub_HCIE`.

In HCIE, a total of $\left(1+\frac{M}{S_M}\cdot\frac{N}{S_N}\right)$ permutation matrices are involved: 1) one high-level permutation matrix of size $\frac{M}{S_M}\times\frac{N}{S_N}$; 2) $\left(\frac{M}{S_M}\cdot\frac{N}{S_N}\right)$ low-level permutation matrices of size $S_M\times S_N$. With the above-mentioned representation of permutation-only image ciphers, the secret key $(\mu, x(0))$ of HCIE is equivalent to the $\left(1+\frac{M}{S_M}\cdot\frac{N}{S_N}\right)$ permutation matrices. To facilitate the following discussions, we use $\boldsymbol{W}_0=[w_0(i,j)]_{\frac{M}{S_M}\times\frac{N}{S_N}}$ to denote the high-level permutation matrix, and use $\left\{\boldsymbol{W}_{(i,j)}\right\}_{i=0,j=0}^{\frac{M}{S_M}-1,\frac{N}{S_N}-1}$ to denote the $\left(\frac{M}{S_M}\times\frac{N}{S_N}\right)$ low-level permutation matrices, where $\boldsymbol{W}_{(i,j)}=\left[w_{(i,j)}(i',j')\right]_{S_M\times S_N}$. Apparently, the $\left(1+\frac{M}{S_M}\cdot\frac{N}{S_N}\right)$ permutation matrices can be easily transformed to an equivalent permutation matrix of size $M\times N$: $\boldsymbol{W}=[w(i,j)]_{M\times N}$.

When $S_M=M$ and $S_N=N$ (or $S_M=S_N=1$), the two hierarchical encryption levels merge a single layer; the $\left(1+\frac{M}{S_M}\cdot\frac{N}{S_N}\right)$ permutation matrices become one permutation matrix of size $M\times N$; and HCIE is simplified to be CIE [28] – a typical permutation-only image cipher in which each pixel can be freely permuted to be any other positions in the whole image by a single $M\times N$ permutation matrix $\boldsymbol{W}$.

§2.3.2 Two-Dimensional Circulation Encryption Algorithm (TDCEA)

The basic idea used in TDCEA is to rotate secret bits of every 64 consecutive bits (of 8 consecutive pixels), which are controlled by a chaotic PRBS. First, some definitions and notations are given in order to introduce TDCEA. Assuming two matrices $\boldsymbol{M}$ and $\boldsymbol{M}'$ of size $m\times n$, where m is the height and n is the width, two mapping operations are defined as follows.

Definition 2.5: *The horizontal rotation mapping,* $\mathrm{Rotate}X_i^{p,r}:\boldsymbol{M}\to\boldsymbol{M}'$ $(0\leq i\leq m-1)$*, is defined to circularly rotate the i-th row of $\boldsymbol{M}$, in the left (when $p=1$) or right (when $p=0$) direction, by r elements.*

Definition 2.6: *The vertical rotation mapping,* $\mathrm{Rotate}Y_j^{q,s}:\boldsymbol{M}\to\boldsymbol{M}'$ $(0\leq j\leq n-1)$*, is defined to circularly rotate the j-th column of $\boldsymbol{M}$, in the up (when $q=1$) or down (when $q=0$) direction, by s elements.*

Definition 2.7: *The cyclical shift operation,* $\mathrm{ROLR}_p^q:\boldsymbol{M_1}\to\boldsymbol{M_1'}$*, is defined to circularly rotate the elements of $\boldsymbol{M_1}$, in the left (when $p=1$) or right (when $p=0$) direction, by q elements, where $\boldsymbol{M_1}$ is a matrix of size $1\times n$. If $\boldsymbol{M_1}$ is a number, we replace it with its binary representation (from LSB to MSB).*

Note that actually mapping $\text{Rotate}X_i^{p,r}$ *and* $\text{Rotate}Y_j^{q,s}$ *are the same with* $ROLR_b^{i,p}$ *(Def. 2.1) and* $ROUD_b^{j,p}$ *(Def. 2.2) respectively, we keep the difference to make them consistent with corresponding original literature.*

TDCEA encrypts a plain-image block by block, where each block contains 8 consecutive pixels. To simplify the following description, without loss of generality, assume that the plain-image is of size $M \times N$ and MN can be exactly divided by 8, where M is the height and N is the width of the image. Consider the 2-D plain-image $\{f(x,y)\}_{x=0,y=0}^{M-1,N-1}$ as a 1-D signal $\{f(l)\}_{l=0}^{MN-1}$ by scanning it in raster order*. Then, the plain-image can be divided into $MN/8$ blocks:

$$\{f^{(8)}(0), \cdots, f^{(8)}(k), \cdots, f^{(8)}(MN/8-1)\},$$

where

$$f^{(8)}(k) = \{f(8k+0), \cdots, f(8k+i), \cdots, f(8k+7)\}.$$

Rewrite each block $f^{(8)}(k)$ as an 8×8 bit matrix $\boldsymbol{M}_k = [M_k(i,j)]_{i=0,j=0}^{7,7}$, by assigning the 64 bits in the current block in the raster order: $f(8k+i) = \sum_{j=0}^{7} M_k(i,j) \cdot 2^j$. In the same way, 8 pixels of each block of the cipher-image can be obtained from the transformation of $\boldsymbol{M}_k$, $\boldsymbol{M}'_k = [M'_k(i,j)]_{i=0,j=0}^{7,7}$, by $f'(8k+i) = \sum_{j=0}^{7} M'_k(i,j) \cdot 2^j$. Based on the matrix-representations of the plain/cipher-images, the working mechanism of TDCEA can be described as follows.

- *The secret key*: two integers α, β, the initial condition $x(0)$, and the control parameter μ of the Logistic map Eq. (2.8), where $0 < \alpha < 8$, $0 \leq \beta < 8$ and $0 < \alpha + \beta < 8$.
- *The initialization procedure*: run the Logistic map starting from $x(0)$ to generate a chaotic sequence, $\{x(k)\}_{k=0}^{MN/8-1}$, and then extract the 17-bit representation of $x(k)$ to yield a PRBS, $\{b(i)\}_{i=0}^{17MN/8-1}$. In the hardware implementation given in [38, 43], the Logistic map is realized in 17-bit fixed-point arithmetic.
- *The encryption procedure*:
  - *Step 1 – horizontal rotations*: for $i = 0 \sim 7$ do $\boldsymbol{M}_k^* = \text{Rotate}X_i^{p,r}(\boldsymbol{M}_k)$, where $p = b(17k+i)$ and $r = \alpha + \beta \cdot b(17k+i+1)$;
  - *Step 2 – vertical rotations*: for $j = 0 \sim 7$ do $\boldsymbol{M}'_k = \text{Rotate}Y_j^{q,s}(\boldsymbol{M}_k^*)$, where $q = b(17k+8+j)$ and $s = \alpha + \beta \cdot b(17k+9+j)$.

*Note that in [38, 43] TDCEA is described directly for 1-D signals. In this chapter, we prefer to explicitly mention the transform from 2-D images to 1-D signals, so as to emphasize the relation between BRIE and TDCEA (which is not mentioned in [38, 43]).

- *The decryption procedure* is a simple reversion of the above encryption procedure, as follows:
  - *Step 1 – vertical rotations*: for $j = 0 \sim 7$ do $\boldsymbol{M}_k^* = \text{RotateY}_j^{q,s}(\boldsymbol{M}_k')$, where $q = \overline{b(17k + 8 + j)}$ and $s = \alpha + \beta \cdot b(17k + 9 + j)$;
  - *Step 2 – horizontal rotations*: for $i = 0 \sim 7$ do $\boldsymbol{M}_k = \text{Rotate}X_i^{p,r}(\boldsymbol{M}_k^*)$, where $p = \overline{b(17k + i)}$ and $r = \alpha + \beta \cdot b(17k + i + 1)$.

§2.4 Cryptanalysis of HCIE

In this section, we discuss how to utilize the general known/chosen-plaintext attacks to the normalized permutation-only image ciphers in §2.2 to break HCIE. Also, we will point out in passing that the security of HCIE against brute-force attacks was much over-estimated in [27, 30, 33]. Note that HCIE has not been cryptanalyzed by others yet till now.

§2.4.1 The Known-Plaintext Attack

Since HCIE is a permutation-only image cipher, given n known plain-images $f_1 \sim f_n$ of size $M \times N$ and the corresponding cipher-images $f_1' \sim f_n'$, one can simply call the `Get_Permutation_Matrix` function with the input parameter $(f_1 \sim f_n, f_1' \sim f_n', M, N)$ to estimate an $M \times N$ permutation matrix $\boldsymbol{W}$, which is equivalent to the $\left(1 + \frac{M}{S_M} \cdot \frac{N}{S_N}\right)$ smaller permutation matrices. However, if the hierarchical structure of HCIE is considered, the known-plaintext attack may be quicker and the estimation will be more effective, as demonstrated later in §2.4.4. Thus, the following hierarchical procedure of known-plaintext attacks to HCIE is suggested*:

- *Reconstruct the high-level permutation matrix* $\boldsymbol{W}_0$:
  - for $i = 0 \sim \left(\frac{M}{S_M} - 1\right)$ and $j = 0 \sim \left(\frac{N}{S_N} - 1\right)$, calculate the mean values of the $2n$ blocks $P_{f_1}(i,j) \sim P_{f_n}(i,j)$, $P_{f_1'}(i,j) \sim P_{f_n'}(i,j)$ and denote them by $\overline{P_{f_1}(i,j)} \sim \overline{P_{f_n}(i,j)}$ and $\overline{P_{f_1'}(i,j)} \sim \overline{P_{f_n'}(i,j)}$;
  - generate $2n$ images $\overline{P}_{f_1} \sim \overline{P}_{f_n}$ and $\overline{P}_{f_1'} \sim \overline{P}_{f_n'}$ of size $\frac{M}{S_M} \times \frac{N}{S_N}$ as follows: $\forall m = 1 \sim n$,

$$\overline{P}_{f_m} = \left[\overline{P_{f_m}(i,j)}\right]_{\frac{M}{S_M} \times \frac{N}{S_N}} \tag{2.12}$$

*For HCIE, the permutation matrices also depend on the values of the public parameters. To simplify the following description, specifically, it is assumed that all public parameters are fixed for all known plain-images.

and

$$\overline{P}_{f'_m} = \left[\overline{P_{f'_m}(i,j)}\right]_{\frac{M}{S_M}\times\frac{N}{S_N}}, \tag{2.13}$$

and call the `Get_Permutation_Matrix` function with the input parameters

$$\left(\overline{P}_{f_1} \sim \overline{P}_{f_n}, \overline{P}_{f'_1} \sim \overline{P}_{f'_n}, \frac{M}{S_M}, \frac{N}{S_N}\right)$$

to get an estimated permutation matrix $\widetilde{\boldsymbol{W}}_0 = [\widetilde{w}_0(i,j)]_{\frac{M}{S_M}\times\frac{N}{S_N}}$ and its inverse $\widetilde{\boldsymbol{W}}_0^{-1} = \left[\widetilde{w}_0^{-1}(i,j)\right]_{\frac{M}{S_M}\times\frac{N}{S_N}}$.

- *Reconstruct the* $\left(\frac{M}{S_M}\cdot\frac{N}{S_N}\right)$ *low-level permutation matrices* $\left\{\boldsymbol{W}_{(i,j)}\right\}_{i=0,j=0}^{\frac{M}{S_M}-1,\frac{N}{S_N}-1}$:
  - for $i = 0 \sim \left(\frac{M}{S_M}-1\right)$ and $j = 0 \sim \left(\frac{N}{S_N}-1\right)$, call the `Get_Permutation_Matrix` function with the input parameters $(P_{f_1}(i,j) \sim P_{f_n}(i,j), P_{f'_1}(i',j') \sim P_{f'_n}(i',j'), S_M, S_N)$, where $(i',j') = W_0(i,j)$, to determine an estimated permutation matrix $\widetilde{\boldsymbol{W}}_{(i,j)}$ and its inverse $\widetilde{\boldsymbol{W}}_{(i,j)}^{-1}$.

With the $\left(1+\frac{M}{S_M}\cdot\frac{N}{S_N}\right)$ inverse matrices $\boldsymbol{W}_0^{-1}$ and $\left\{\boldsymbol{W}_{(i,j)}\right\}_{i=0,j=0}^{\frac{M}{S_M}-1,\frac{N}{S_N}-1}$, one can decrypt a new cipher-image f'_{n+1} as follows to get an estimated plain-image f^*_{n+1}:

for $(i=0;\ i\le(M/S_M)-1;\ i++)$
    for $(j=0;\ j\le(N/S_N)-1;\ j++)$ {
        $f_{temp} = P_{f'_{n+1}}(w_0^{-1}(i,j))$;
        for $(ii=0;\ ii\le S_M-1;\ ii++)$
            for $(jj=0;\ jj\le S_N-1;\ jj++)$
                $f^*_{temp}(ii,jj) = f_{temp}\left(w_{(i,j)}^{-1}(ii,jj)\right)$;
        $P_{f^*_{n+1}}(i,j) = f^*_{temp}$;
    }

In fact, in the above procedure, any measure keeping invariant in the block permutations can be used instead of the mean value. A typical measure is the histogram of each $S_M\times S_N$ block. Although the mean value is less precise than the histogram, it works well in most cases and is useful to reduce the time complexity. When L and $S_M\times S_N$ are both too small, the efficiency of the mean value will become low, and the histogram or the array of all pixel values can be used as a replacement. Apparently, in most cases it is easier to get the high-level permutation matrix $\boldsymbol{W}_0$ than the low-level permutation matrices.

Finally, let us see whether the hierarchical structure used in HCIE is helpful to enhance the security against the known-plaintext attack to the common permutation image ciphers. As discussed above, $n \geq \lceil \log_L(2(MN-1)) \rceil$ known plain-images are needed to achieve an acceptable breaking performance. Since the hierarchical structure makes it possible for an attacker to work on permutation matrices of size $S_M \times S_N$ or $\frac{M}{S_M} \times \frac{N}{S_N}$ (both smaller than $M \times N$), it is obvious that for HCIE the number of required known plain-image will be smaller than $\lceil \log_L(2(MN-1)) \rceil$. Also, the attack complexity will become less, since it is proportional to the square of the matrix sizes. In such a sense, hierarchical permutation-only image ciphers are less secure than non-hierarchical ones, which discourages the use of HCIE. This result has been confirmed by our experiments (see §2.4.4).

§2.4.2 The Chosen-Plaintext Attack

Following the same way introduced in the chosen-plaintext attack to common permutation-only image ciphers, one can choose $n = \lceil \log_L(MN) \rceil$ plain-images to carry out a chosen-plaintext attack to HCIE. Similar to the known-plaintext attack, the use of a hierarchical structure in HCIE can also make the construction of chosen plain-images easier. Accordingly, an attacker can also work hierarchically to construct n chosen plain-images, $f_1, \cdots, f_n$, as follows:

- *high-level*: $\overline{P}_{f_1} \sim \overline{P}_{f_n}$, which are defined in Eq. (2.12), compose an orthogonal image set;
- *low-level*: $\forall (i,j)$, $P_{f_1}(i,j) \sim P_{f_n}(i,j)$ compose an orthogonal image set.

In this case, the minimal number of required chosen plain-image becomes

$$\begin{aligned} n &= \max\left(\lceil \log_L(S_M \cdot S_N) \rceil, \left\lceil \log_L\left(\frac{M}{S_M} \cdot \frac{N}{S_N} \right) \right\rceil \right) \\ &\leq \lceil \log_L(MN) \rceil, \end{aligned} \tag{2.14}$$

where the equality holds if and only if the hierarchical encryption structure is disabled, i.e., when $(S_M = M, S_N = N)$ or $(S_M = S_N = 1)$.

§2.4.3 The Brute-Force Attack

The brute-force attack is the attack of exhaustively searching the secret key from the set of all possible keys [1]. Apparently, the attack complexity is determined by the size of the key space and the complexity of verifying each key. In [27, 30, 33], it was claimed that the complexity of brute-force attacks to HCIE is $O\left(2^{L_b}\right)$, since there are $L_b = \left(1 + \frac{M}{S_M} \cdot \frac{N}{S_N}\right) \cdot (3S_M + 3S_N - 2) \cdot no$ secret chaotic bits in $\{b(i)\}_{i=0}^{L_b-1}$ that are unknown to attackers. However, this statement is not true due to the following fact: the L_b bits are uniquely determined by the secret key, i.e., the initial condition $x(0)$ and the control parameter μ, which have only $2L$ secret bits. This means that there are only 2^{2L} different chaotic bit sequences. Now, let us study the real complexity of brute-force attacks. For each pair of guessed values of $x(0)$ and μ, the following operations are needed:

- generating the chaotic bit sequence: $L_b/8$ chaotic iterations;
- creating the pseudo-image f_p: the complexity is $S_M \cdot S_N$;
- shuffling the pseudo-image f_p: running the `Sub_HCIE` function once;
- generating P_{f^*}: the complexity is $M \cdot N$;
- shuffling the partition image P_{f^*}: running the `Sub_HCIE` function for $\left(\frac{M}{S_M} \cdot \frac{N}{S_N}\right)$ times.

Assume that the computing complexity of the `Sub_HCIE` function is $(4S_M + 4S_N) \cdot no$. Then, the total complexity of brute-force attacks to HCIE will be

$$O\left(2^{2L} \cdot \left(\frac{L_b}{8} + S_M \cdot S_N + MN + \left(\frac{M}{S_M} \cdot \frac{N}{S_N} + 1\right) \cdot (3S_M + 3S_N) \cdot no\right)\right) \approx O\left(2^{2L} \cdot (L_b + MN)\right), \tag{2.15}$$

which is much smaller than $O\left(2^{L_b/8}\right)$ when L_b is not too small. Additionally, considering the fact that the Logistic map can exhibit a sufficiently strong chaotic behavior only when μ is close to 4 [82], the complexity should be even smaller. The above analysis shows that the security of HCIE was much over-estimated by the authors in [27, 30, 33], even under brute-force attacks.

§2.4.4 Experiments

To verify the decryption performance of the above-discussed known-plaintext attack to general permutation-only image ciphers working in spatial domain and particularly to HCIE, some experiments are performed using the six 256×256 test images with 256 gray scales shown in Fig. 2.1. Assume that the first $n = 1 \sim 5$ test images are known to an attacker, the cipher-image of the last test image is decrypted with the estimated permutation matrices to see the breaking performance. In the experiments, the "taking-the-first" algorithm is used to generate $\widetilde{\boldsymbol{W}}$ from $\widehat{\boldsymbol{W}}$ in the `Get_Permutation_Matrix` function. It turns out that such a simple algorithm is enough to achieve a considerable performance in real attacks.

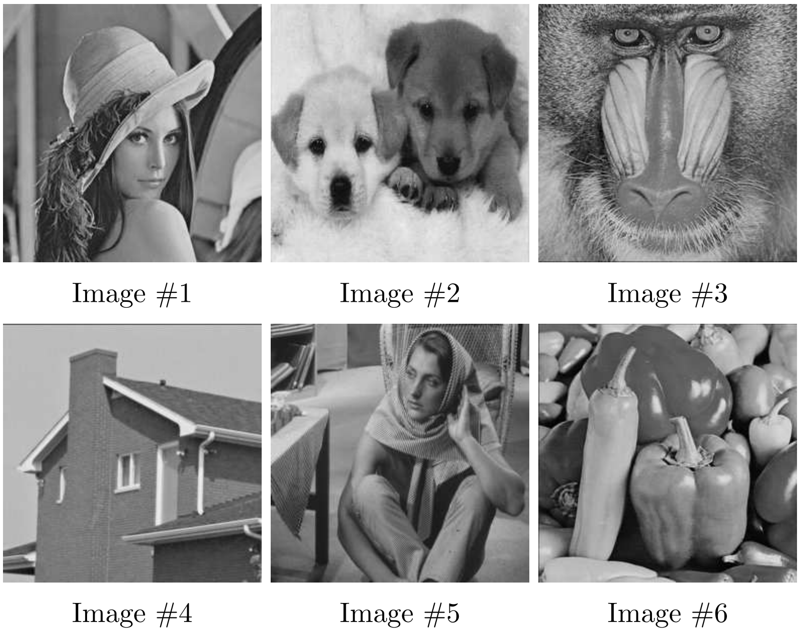


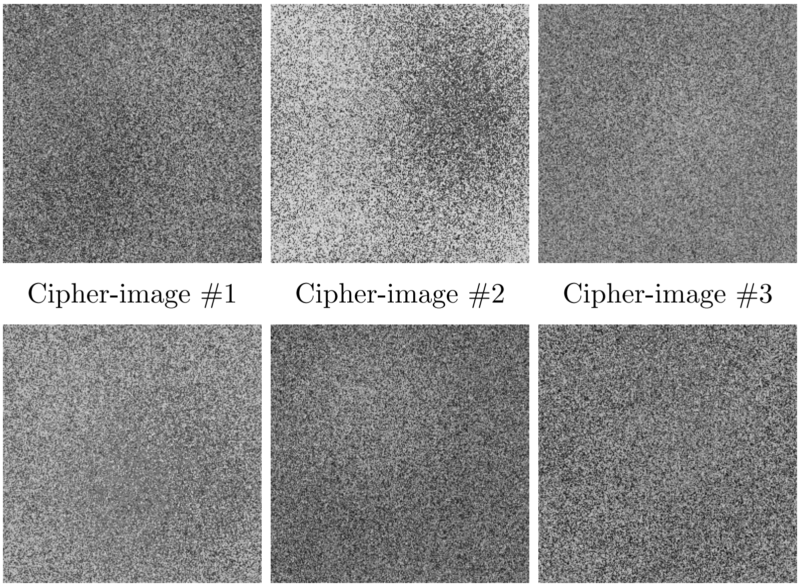
Figure 2.1: The six 256×256 test images used in the experiments.

In the experiments, three different configurations of HCIE are used: $S_M = S_N = 256$, $S_M = S_N = 32$, $S_M = S_N = 16$. As mentioned above, the configuration of $S_M = S_N = 256$ corresponds to general permutation-only image ciphers working in spatial domain (without using hierarchical structures). It is shown that three known plain-images are always enough to achieve a good breaking performance, and that an almost perfect breaking performance can be achieved with four plain-images. Thus, the theoretical analysis given in the last section is verified. Also, it has been confirmed that the security of the two-level hierarchical encryption structure is weaker than the security of the non-hierarchical structure. As a result, the security of HCIE against known-plaintext attack is even weaker than the security of other common permutation-only image ciphers.

The chosen-plaintext attack is omitted in this sub-section, since one can absolutely break the permutation matrix by choosing two plain-images f_1 and f_2 as shown in Eqs. (2.6) and (2.7). Of course, some experiments have been completed to verify the theoretical result and the correctness of the uniquely-determined permutation matrix.

§2.4.4.1 The experimental results with $S_M = S_N = 256$ (non-hierarchical)

The public parameters are $\alpha = 6$, $\beta = 3$, $\gamma = 3$ and $no = 9$. The cipher-images of the six test images are shown in Fig. 2.2. When the first $n = 1 \sim 5$ test images and their cipher-images are known to the attacker, the five decrypted images of the sixth cipher-image are shown in Fig. 2.3. As can seen, one known plain-image cannot reveal any visual information, but two is capable to recover a rough view of the sixth test image, and three are enough to obtain a good recovery.



Cipher-image #1 Cipher-image #2 Cipher-image #3

Cipher-image #4 Cipher-image #5 Cipher-image #6

Figure 2.2: The cipher-images of the six test images, when $S_M = S_N = 256$.

To verify the fact that the breaking performance is better than the theoretical prediction based on the correctly-recovered elements in $\widetilde{W}$, let us see the decryption performance with $n = 2$ as an example. For this case, the number of the absolutely correct elements in $\widetilde{W}$ are only 10,600, and the number of all correct elements in $\widetilde{W}$ is 26,631. In comparison, the number of correctly-recovered pixels are 27,210. Although only about $\frac{27210}{65536} \approx 41.52\%$ of the pixels are recovered, most visual information in the plain-image #6 has been revealed successfully. Now, let us consider the correct pixels that are not recovered from the correct elements in $\widetilde{W}$, i.e, the $(27210 - 26631 = 579)$ more correct pixels. These pixels are correctly decrypted with a frequency $\frac{579}{65536-26631} \approx 0.0149$, which is much larger than the average probability $L^{-1} \approx 0.0039$. If we also count those pixels whose values close

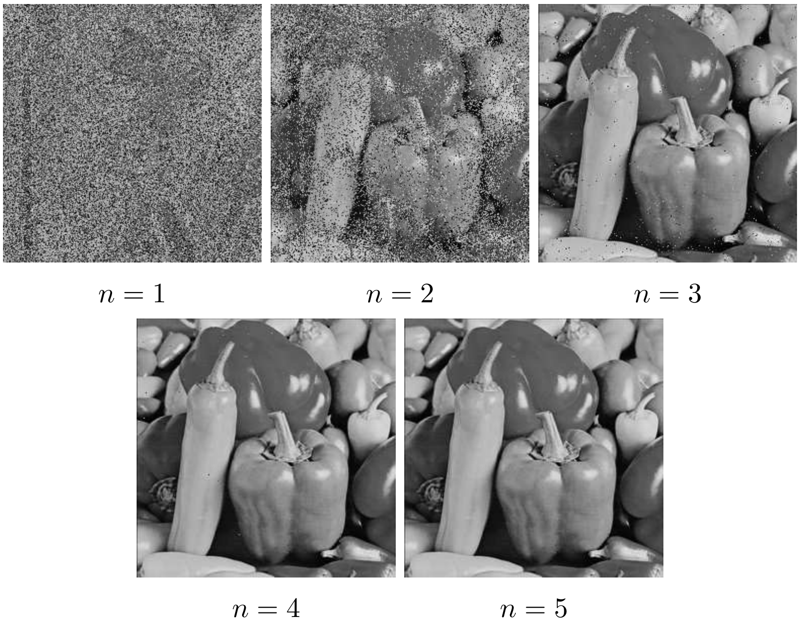


Figure 2.3: The decrypted images of Cipher-Image #6 when the first n test images are known to the attacker, when $S_M = S_N = 256$.

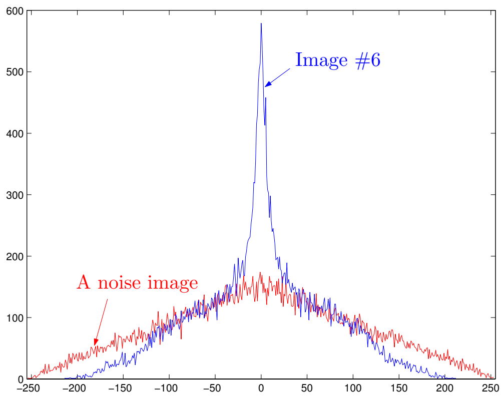


Figure 2.4: The histogram of the difference image between the recovered image and the original plain-image, when the plain-image is Image #6 (the blue line) or a randomly-generated noise image (the red line).

to the right ones, this frequency will be even larger. In fact, excluding the pixels correctly determined by the 26,631 correct elements in $\widetilde{W}$, the histogram of the other $(65536-26631 = 38905)$ pixels of the difference image between the recovered image and the original plain-image #6 is a Gaussian-like function as shown in Fig.

2.4. In comparison, the histogram of the difference image corresponding to a randomly-generated noise image of the same size 256 × 256 is also shown. It is clear that the Gaussian-like histogram corresponding to Image #6 is caused by the correlation information existing in natural images. Note that the triangular histogram of the noise image can be easily deduced under the assumption that the pixels of the two involved images (i.e., the noise image and the corresponding cipher-image) are independent of each other and have a uniform histogram: $\forall i = -255 \sim 255$, the occurrence probability of the difference value i in the histogram is: $\frac{256-|i|}{65536} = \frac{1}{256} - \frac{|i|}{65536}$.

§2.4.4.2 The experimental results with $S_M = S_N = 32$

The public parameters are $\alpha = 4$, $\beta = 2$, $\gamma = 1$ and $no = 2$. The cipher-images of the six test images are all shown in Fig. 2.5. When the first $n = 1 \sim 5$ test images are known to the attacker, the five decrypted images of the sixth cipher-image are shown in Fig. 2.6. As can be seen, one known plain-image cannot reveal much useful visual information, but two is enough to obtain a good performance.

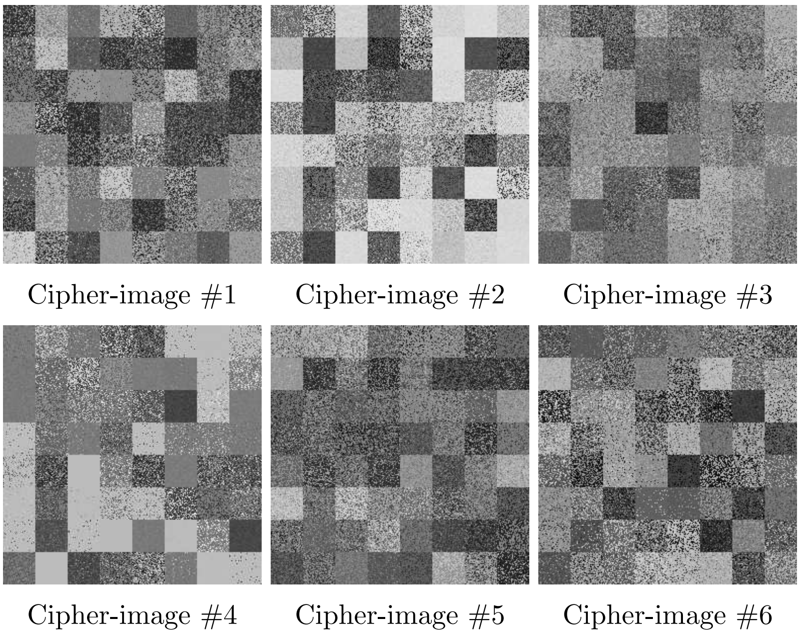


Figure 2.5: The cipher-images of the six 256 × 256 test images, when $S_M = S_N = 32$.

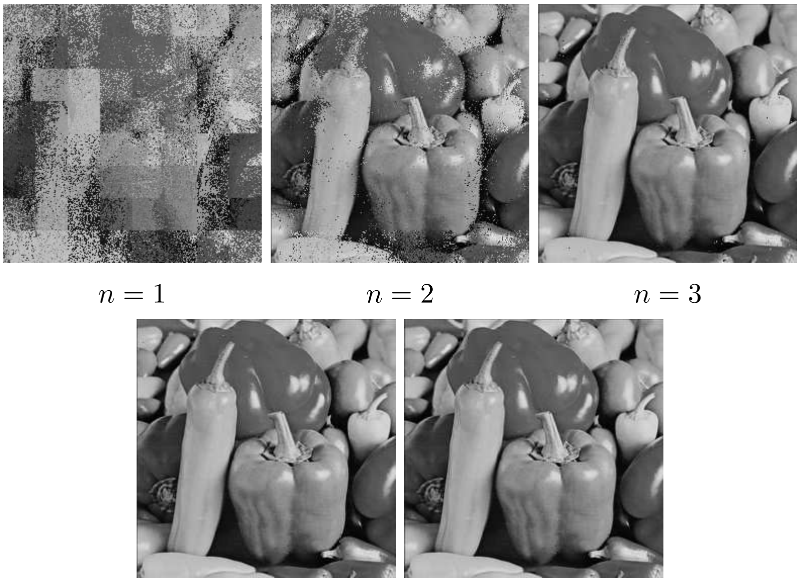


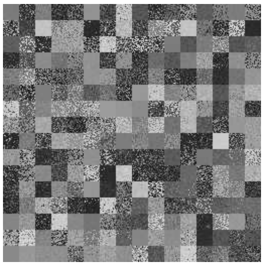
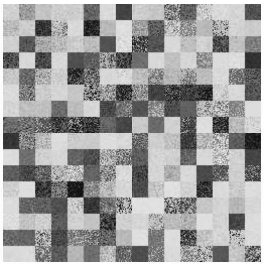
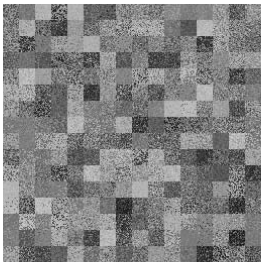
Figure 2.6: The decrypted image of Cipher-Image #6 when the first n test images are known to the attacker, when $S_M = S_N = 32$.

§2.4.4.3 The experimental results with $S_M = S_N = 16$

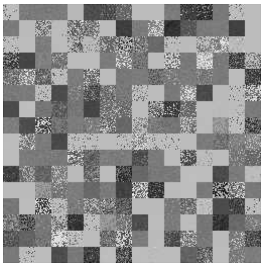
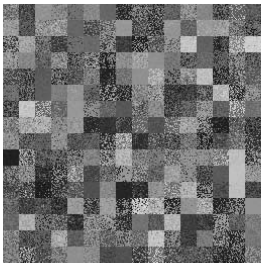
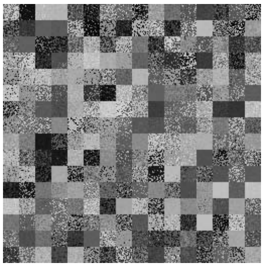
The public parameters are $\alpha = 4$, $\beta = 2$, $\gamma = 1$ and $no = 2$. The cipher-images of the six test images are all shown in Fig. 2.7. When the first $n = 1 \sim 5$ test images are known to the attacker, the five decrypt images of the sixth cipher-image are shown in Fig. 2.8. As can be seen, even one known plain-image can reveal a rough view of the plain-image, and two is enough to obtain a nearly-perfect recovery.

§2.4.4.4 A comparison of the performances

This sub-subsection gives a performance comparison of the known-plaintext attack to HCIE with the above three different configurations. Figure 2.9a shows the quantitative relation between the number of known plain-images and the decryption quality (represented by the decryption error ratio). It can be seen that three known plain-images are enough for all three configurations to achieve an acceptable breaking performance, and two can reveal quite a lot of pixels (which means that most significant visual information is revealed). Also, it is shown that the breaking performance is dependent on the configuration: when $S_M = S_N = 16$, the best performance is achieved, which coincides with the expectation from Eq. (2.2): n is minimized when $S_M = S_N = \sqrt{256} = 16$.

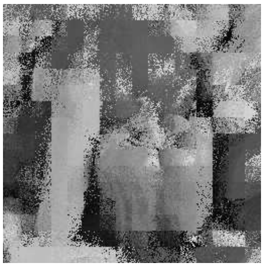




Cipher-image #1 Cipher-image #2 Cipher-image #3

Cipher-image #4 Cipher-image #5 Cipher-image #6

Figure 2.7: The cipher-images of the six 256×256 test images, when $S_M = S_N = 16$.



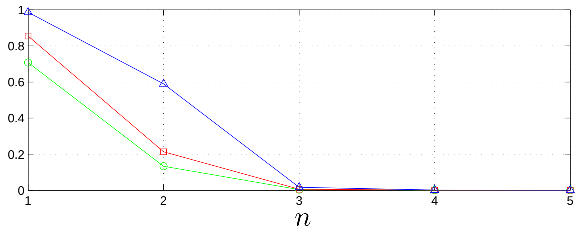
$n = 1$ $n = 2$ $n = 3$


$n = 4$ $n = 5$

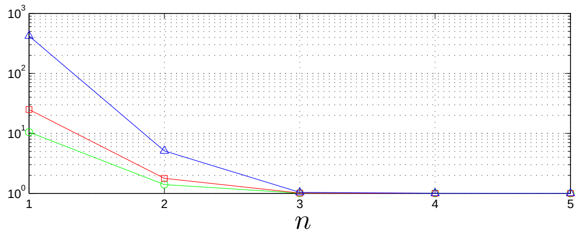
Figure 2.8: The decrypted images of Cipher-Image #6 when the first n test images are known to an attacker, when $S_M = S_N = 16$.

Figure 2.9b shows the average cardinality of the elements in $\widehat{\boldsymbol{W}}$, which is an indicator of the probability of getting correct permutation elements in $\widetilde{\boldsymbol{W}}$ and an indicator of the time complexity as analyzed above. Comparing Figures 2.9a and 2.9b, one can see that the occurrence probability of decryption errors has a good correspondence with the average cardinality.

From the above comparison, it is true that the security of HCIE with a hierarchical structure is even weaker than the security of general permutation-only image ciphers without hierarchical structures: when $S_M = S_N = 32$ and $S_M = S_N = 16$, two known plain-images are enough to achieve an acceptable breaking performance; while when $S_M = S_N = 256$, the breaking performance with two known plain-images is not satisfactory, and three plain-images are needed to achieve an acceptable performance. Therefore, from the viewpoint of security against known/chosen-plaintext attacks, the hierarchical idea proposed in HCIE has no technical merits. This verifies the theoretical analyses given in §2.4.1.



a) decryption error ratio



b) the average cardinality $\overline{\#(\widehat{w}(i,j))}$

(Legend: △ – $S_M = S_N = 256$, □ – $S_M = S_N = 32$, ○ – $S_M = S_N = 16$)

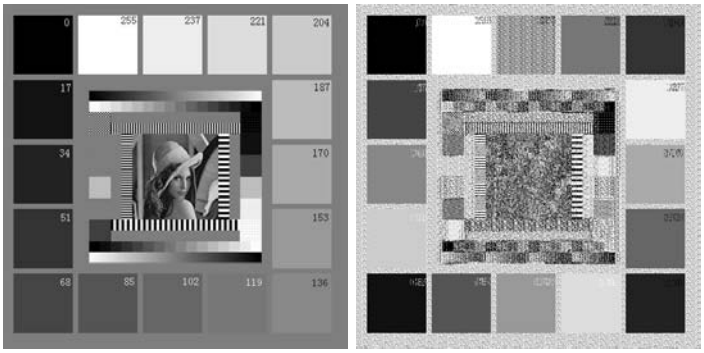
Figure 2.9: A performance comparison of the known-plaintext attack to HCIE

§2.5 Cryptanalysis of TDCEA

§2.5.1 Essential Defects of Circulations

In [81], some essential defects of the *ROLR* operation were found: 1) some plain-pixels may keep unchanged after encryption, so the plain-image will roughly emerge if there are too many such pixels; 2) for a sub-region in the plain-image with a fixed gray value, at most eight gray values* will be contained in the corresponding sub-region of the cipher-image, which will lead the edge of this sub-region to appear in the cipher-image. The second fact is also true for sub-regions with close pixel values.

Although TDCEA extends the shift operation to two dimensions, the above defects of *ROLR* cannot be completely removed. As an extreme example, when all elements in $\boldsymbol{M}_k$ are 0-bits or 1-bits, it is obvious that $\boldsymbol{M}'_k \equiv \boldsymbol{M}_k$, which means TDCEA cannot encrypt blocks with fixed pixel value 0 (black) or 255 (white) at all. To test the performance of TDCEA compared with scheme BRIE[32], we have encrypted the same test image used in [81] for BRIE, with the following parameters: $(\alpha, \beta) = (2, 4)$, $x(0) = 34816/2^{17} \approx 0.2656$, $\mu = 128317/2^{15} \approx 3.9159$. The encryption result is shown in Fig. 2.10, from which one can see that the 16 squares in the plain-image remain fixed in the cipher-image, though the fixed gray values have been changed for most squares. Comparing this result with those given in [81, Fig. 1], it is obvious that the security defects of BRIE is not enhanced by TDCEA.



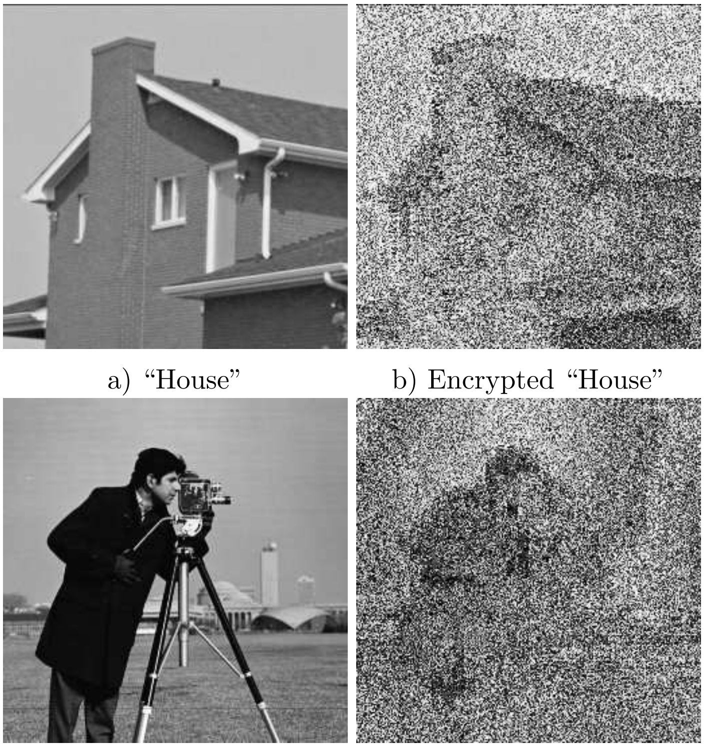
a) the plain-image b) the cipher-image

Figure 2.10: A special test image, "Test_pattern", encrypted by TDCEA.

As a second example to test the possible enhancement of TDCEA on the BRIE security, we also tested the encryption performance of TDCEA on some general natural images containing many smooth areas. As known, the pixels within a smooth area generally have close pixel values, which is found similar to the squares with fixed gray values shown in Fig. 2.10 when TDCEA is applied for encryption. Two images, "House" and "Cameraman", are selected for testing. The experimental results are shown in Fig. 2.11, from which one can see many important

*For some pixel values, the number of different cipher pixel-values is even smaller, which may be 1, 2, or 4 [81, Sec. 3.1].

edges of the plain-images emerging in the cipher-images. In this experiment, the parameters of TDCEA are as follows: $(\alpha, \beta) = (5, 1)$, $x(0) = 33578/2^{17} \approx 0.2562$ and $\mu = 129518/2^{15} \approx 3.9526$.



a) "House" b) Encrypted "House" c) "Cameraman" d) Encrypted "Cameraman"

Figure 2.11: Two natural images, "House" and "Cameraman", encrypted by TDCEA, with $(\alpha, \beta) = (5, 1)$, $x(0) = 33578/2^{17} \approx 0.2562$ and $\mu = 129518/2^{15} \approx 3.9526$.

§2.5.2 Security Problem of α, β

In [38, 43], the values of α and β are constrained by $0 < \alpha < 8$, $0 \leq \beta < 8$ and $0 < \alpha+\beta < 8$. Thus, the number of all possible values of (α, β) is $7+6+\cdots+2+1 = 28$. However, similar to the case of BRIE, α and β should also obey the following rule pointed out in [81]: $\alpha \neq 1, 7$ or $\alpha + \beta \neq 1, 7$. If this rule is not satisfied, then there only exist 1-bit circular rotation operations, since $\text{Rotate}X_i^{p,1} = \text{Rotate}X_i^{p,7}$ and $\text{Rotate}Y_j^{q,1} = \text{Rotate}Y_j^{q,7}$. Generally speaking, 1-bit circular rotations are not good enough to effectively encrypt the plain-image, and some visual information may leak from the cipher-image. When $(\alpha, \beta) = (1, 6)$, $x(0) = 33578/2^{17} \approx 0.2562$,

$\mu = 129518/2^{15} \approx 3.9526$, the encryption results of two plain-images, "House" and "Cameraman", are shown in Fig. 2.12. It can be seen that the visual information containing in the cipher-images is so much (even more than that in Fig. 2.11) that the plain-images can be obviously guessed. Excluding the three values of (α, β) that violate the above rule, $(1, 0), (1, 6), (7, 0)$, the number of all "good" values of (α, β) is only 25 (= 28 − 3).

a) Encrypted "House"   b) Encrypted "Cameraman"

Figure 2.12: Two natural images, "House" and "Cameraman", encrypted by TDCEA, when $(\alpha, \beta) = (1, 6)$, $x(0) = 33578/2^{17} \approx 0.2562$ and $\mu = 129518/2^{15} \approx 3.9526$.

§2.5.3 The Brute-Force Attack

In [38, 43], it was claimed that the complexity of TDCEA against brute-force attack is $O\left(2^{17MN/8}\right)$ since $17MN/8$ secret bits are used in the encryption/decryption procedures. However, this statement is not true due to the following reason: all $17MN/8$ bits are uniquely determined by the initial condition $x(0)$ and the control parameter μ of the Logistic map Eq. (2.8), which have only 34 secret bits. Moreover, not all values of μ can ensure the chaoticity of the Logistic map [82], so we can assure that the number of possible different chaotic bit sequences is smaller than 2^{34}.

Considering that the computational complexity of TDCEA is proportional to $O(MN)$, i.e. $49MN$ operations of all kinds [43, Sec.2.5], and the number of all possible values of (α, β) is less than 25, the total complexity against the brute-force attack is $O(2^{34} \cdot 25 \cdot 49MN) \approx O(2^{44}MN)$. For a typical image of size 256×256, the complexity is about $O(2^{60})$, which is much smaller than $O(2^{17MN/8}) = O(2^{139264})$, the complexity claimed in [38, 43]. Obviously, the security of TDCEA against brute-force attacks was over-estimated too much in [38, 43].

Although it was claimed that TDCEA can efficiently resist known/chosen-plaintext attack [43, Sec.2.6], we propose two different attack methods in this section and §3.6 to effectively break TDCEA. One attack requires a few known plaintexts, and the other requires only one.

§2.5.4 Known-Plaintext Attack: Getting Permutation Matrices as an Equivalent Key

The insecurity of BRIE against known/chosen-plaintext attacks are caused by the fact that the *ROLR* operation is actually composed of secret permutations of all 8 bits of each pixel value. As shown in §2.2, all permutation-only ciphers are not secure against known/chosen-plaintext attacks. Apparently, TDCEA falls into the category of permutation-only ciphers, since the circulation rotations are actually secret permutations of all 64 bits of each 8-pixel block. As a result, if an attacker knows (or chooses) a number of plain-blocks and cipher-blocks at the same position, k, it is possible for him to partially (or even completely) reconstruct the bit permutation by comparing $\boldsymbol{M}_k$ and $\boldsymbol{M}'_k$. This is the basic principle of known/chosen-plaintext attacks to be discussed below.

Apparently, for the k-th pixel-block $f^{(8)}(k)$ and its cipher-block $f'^{(8)}(k)$, the encryption transformation can be represented by an 8×8 permutation matrix, $\boldsymbol{W}_k = [W_k(i,j)]_{i=0,j=0}^{7,7}$, where $W_k(i,j) = (i',j')$ denotes the secret position of the plain-bit $M_k(i,j)$ in $\boldsymbol{M}'_k$. Since there are $MN/8$ different blocks, the encryption of f can be represented by $MN/8$ permutation matrices: $\{\boldsymbol{W}_k\}_{k=0}^{MN/8-1}$. Once the attacker gets the $MN/8$ permutation matrices and their inverses, $\{\boldsymbol{W}_k^{-1}\}_{k=0}^{MN/8-1}$, he can use these matrices as an equivalent key to decrypt any cipher-image encrypted with the same key.

In §2.2.2, a general algorithm was proposed for deriving the secret permutations (i.e., the permutation matrices) from a number of known plain-images and the corresponding cipher-images. This algorithm depends on the fact that the secret permutations do not change the values of the permuted elements. As a result, one can compare the values of the elements of the plain-images and the cipher-images to reveal the secret permutations. Here, we show how to optimally realize the general algorithm for TDCEA and discuss the breaking performance.

Given n known plain-images $f_0 \sim f_{n-1}$ and the corresponding cipher-images $f'_0 \sim f'_{n-1}$, denoting the k-th 8×8 bit matrix of the l-th plain-image and cipher-image by $\boldsymbol{M}_{l,k} = [M_{l,k}(i,j)]_{i=0,j=0}^{7,7}$, $\boldsymbol{M}'_{l,k} = [M'_{l,k}(i,j)]_{i=0,j=0}^{7,7}$, respectively, the algorithm of deriving the permutation matrix $\boldsymbol{W}_k$ is described as follows.

- *Step 1a – calculate a generalized bit matrix* $\widetilde{\boldsymbol{M}}_k = \left[\widetilde{M}_k(i,j)\right]_{i=0,j=0}^{7,7}$, where

$\widetilde{M}_k(i,j) = \sum_{l=0}^{n-1} M_{l,k}(i,j) \cdot 2^l$. Apparently, $\widetilde{M}_k(i,j)$ is an n-bit integer.

Note: when n is larger than the word-length of the longest integer (which is 32 or 64 for most computers), it may be impossible to store $\widetilde{M}_k(i,j)$ as a normal integer in a computer. In this case, one has to divide $\widetilde{M}_k(i,j)$ into multiple short integers for storage and computation (i.e., to use long-integer techniques). Since the long-integer technique is easy for implementations and n is generally smaller than 32 in most attacking scenarios, here we do not pay special attention on this issue.*

- *Step 1b – calculate a generalized bit matrix* $\widetilde{\boldsymbol{M}}'_k = \left[\widetilde{M}'_k(i,j)\right]_{i=0,j=0}^{7,7}$, in the same way as *Step 1a*.
- *Step 2 – get multi-valued permutation matrix,* $\widehat{\boldsymbol{W}}_k = \left[\widehat{W}_k(i,j)\right]_{i=0,j=0}^{7,7}$, where $\widehat{W}_k(i,j) = \left\{(i',j') \mid \widetilde{M}_k(i,j) = \widetilde{M}'_k(i',j')\right\}$.
- *Step 3 – derive an estimation of the permutation matrix* $\boldsymbol{W}_k$ *from* $\widehat{\boldsymbol{W}}_k$.

Apparently, if and only if each element of $\widehat{\boldsymbol{W}}_k$ contains only one pixel position, i.e., the measure of every element of $\widehat{\boldsymbol{W}}_k$ is 1, one can uniquely get the permutation matrix $\boldsymbol{W}_k$; otherwise, only an estimated version, $\widetilde{\boldsymbol{W}}_k$, can be derived. In other words, $\widetilde{\boldsymbol{W}}_k = \boldsymbol{W}_k$ holds if and only if the cardinality of $\widehat{\mathbb{W}}_k = \left\{\widehat{W}_k(0,0), \cdots, \widehat{W}_k(7,7)\right\}$ is 64, i.e., $\#\left(\widehat{\mathbb{W}}_k\right) = 64$. When $\#\left(\widehat{\mathbb{W}}_k\right) = P < 64$, with n_i ($i = 1 \sim P$) denoting the measure of the P different elements in $\widehat{\boldsymbol{W}}_k$, one can easily deduce that there are $\prod_{i=1}^{P}(n_i!)$ possible estimations of $\boldsymbol{W}_k$ in total. Thus, the task of *Step 3* is to determine one estimated permutation matrix from all $\prod_{i=1}^{P}(n_i!)$ possible ones. Although many different methods can be used to realize *Step 3*, the following simple algorithm is enough in most cases to achieve an acceptable performance:

- Initialize all elements of an 8×8 flag matrix, $\boldsymbol{F}_k = [F_k(i,j)]_{i=0,j=0}^{7,7}$, to zeros.
- For $i = 0 \sim 7$ and $j = 0 \sim 7$, determine the value of $\widetilde{W}_k(i,j)$ as follows:
    1. find the first position (i',j') satisfying $M_k(i,j) = M'_k(i',j')$ and $F_k(i',j') = 0$;
    2. set $\widetilde{W}_k(i,j) = (i',j')$ and $F_k(i',j') = 1$.

*As discussed below, the breaking performance is rather good when $n \leq 32$ (see Fig. 2.14), so one can simply set $n = 32$ even when $n > 32$.

Note that *Step 2* is also incorporated into the above algorithm, which is very useful in reducing the total complexity.

From Eq. (2.2), we can assure that the attacker can achieve an acceptable breaking performance when $n \geq 1 + \lceil \log_2 63 \rceil = 1 + \lceil 5.9773 \rceil = 7$ plain-images are known. Although this result is deduced under the assumption that $\{\boldsymbol{M}_{l,k}\}$ is an independent and identically distributed sequence, it can be qualitatively generalized to other distributions of $\{\boldsymbol{M}_{l,k}\}$. Our experiments show that the theoretical result essentially holds for most natural images.

For a randomly selected key, $(\alpha, \beta) = (2, 2)$, $x(0) = 33578/2^{17} \approx 0.2562$, $\mu = 129518/2^{15} \approx 3.9526$, a set of known plain-images (all natural images) are randomly selected for testing. When $n = 8$, the plain-image "Peppers" (Fig. 2.13a) and its cipher-image (Fig. 2.13b) are used to verify the breaking performance based on $MN/8$ estimated permutation matrices, $\{\widetilde{\boldsymbol{W}}_k\}_{k=0}^{MN/8-1}$. The recovered plain-image is shown in Fig. 2.13c. It is found that almost all visual information contained in the original plain-image has been successfully recovered, though only $38012/65536 = 58\%$ of plain-pixels are correct in value. With some noise reduction algorithms, one can further enhance the recovered plain-image. One enhanced result with a 3×3 median filter is shown in Fig. 2.13d.

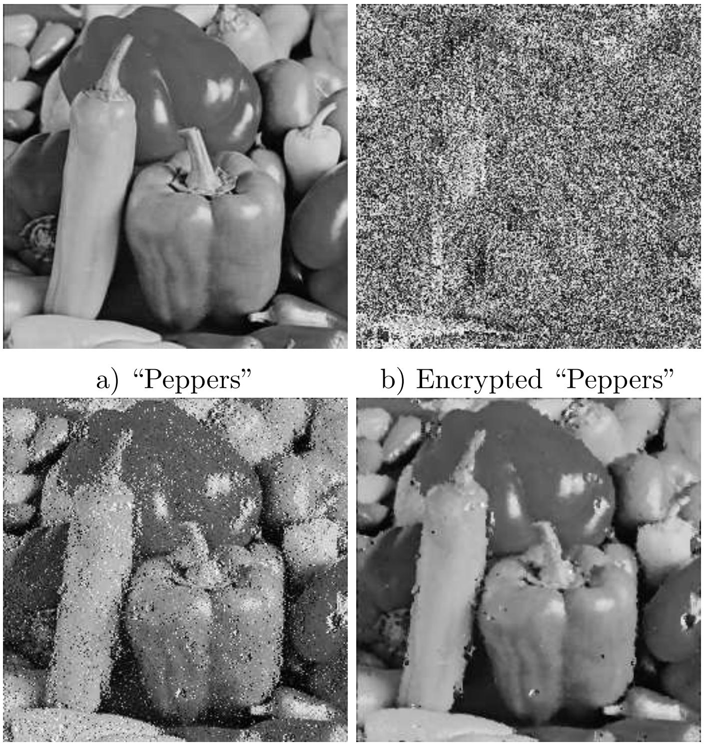
Figure 2.14 shows the percentage of correctly-recovered plain-pixels with respect to n, the number of known plain-images. One can see that the breaking performance is good when $n \geq 8$. Also, it is found that the breaking performance of the natural image is better than the noisy image under the same condition, which is attributed to the correlation existing in the natural image for decryption as discussed in §2.2.2. It can also be observed that the slope of the two lines in Fig. 2.14 are very flat when $n \geq 16$, this is also due to the correlation of the known-images (e.g., the MSBs of adjacent pixels are the same with a high probability).

The complexity of this attack is rather small. For each block, the time complexity consumed in *Step 1a* and *Step 1b* is $O(2 \cdot 64 \cdot (n-1))$, and the average complexity in *Step 2* is $O(64 \cdot 32)$, so the total attack complexity is only $O((2 \cdot 64 \cdot (n-1) + 64 \cdot 32) \cdot MN/8) = O(16(n+15)MN)$.

§2.5.5 Chosen-Plaintext Attack: Getting Permutation Matrices as an Equivalent Key

As discussed in above subsection, if $\#(\widetilde{\mathbb{W}}_k) = 64$, the permutation matrix $\boldsymbol{W}_k$ can be uniquely determined. Apparently, it is easy to ensure $\#(\widetilde{\mathbb{W}}_k) = 64$ by choosing the following six plain-images: $\forall\ k = 0 \sim MN/8-1, i = 0 \sim 7, j = 0 \sim 7$,

$$f_0 \ :\ M_{0,k}(i,j) = \lfloor (8i+j)/32 \rfloor \bmod 2;$$
$$f_1 \ :\ M_{1,k}(i,j) = \lfloor (8i+j)/16 \rfloor \bmod 2;$$



a) "Peppers" b) Encrypted "Peppers"

c) Recovered "Peppers" via $\{\widetilde{\boldsymbol{W}}_k\}_{k=0}^{MN/8-1}$ d) Enhanced "Peppers" by a 3×3 median filter

Figure 2.13: The image "Peppers" recovered by the first known-plaintext attack

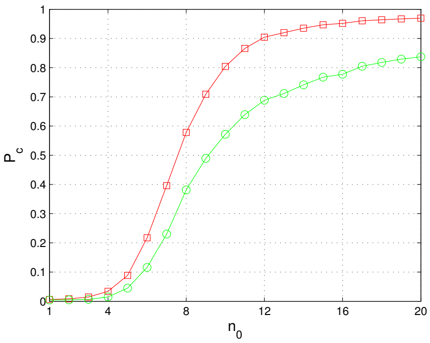


Figure 2.14: The percentage of correctly-recovered pixels, P_c, with respect to the number of known plain-images, n_0.

$$
\begin{aligned}
f_2 &: \quad M_{2,k}(i,j) = \lfloor (8i+j)/8 \rfloor \bmod 2; \\
f_3 &: \quad M_{3,k}(i,j) = \lfloor (8i+j)/4 \rfloor \bmod 2;
\end{aligned}
$$

$$f_4 \quad : \quad M_{4,k}(i,j) = \lfloor (8i+j)/2 \rfloor \bmod 2;$$
$$f_5 \quad : \quad M_{5,k}(i,j) = (8i+j) \bmod 2.$$

With the above six chosen plain-images, $\#(\widetilde{\mathbb{W}}_k) = 64$ holds so all $MN/8$ permutation matrices can be uniquely determined, which can then be used to decrypt any cipher-image of size not greater than MN.

The time complexity of such an attack is of the same order as the known-plaintext attack with $n = 6$ known plain-images, i.e., $O(16(6+15)MN) = O(336MN)$.

In fact, due to a special weakness of TDCEA, even two chosen plain-images are enough to completely reconstruct each 8×8 permutation matrix. Recalling the encryption procedure of TDCEA, one can see that 2-D secret rotations are merely a simple combination of 1-D rotations in two directions: 8 horizontal rotations followed by 8 vertical rotations. Such a property makes the division of the 2-D secret rotations possible in chosen-plaintext attacks with only two plain-images. In cryptanalysis, we call such attacks *divide-and-conquer* (DAC) attacks. The DAC chosen-plaintext attack can be described as follows.

- *Break the 8 vertical secret rotations*: Choose a plain-image f_0 as follows: $\forall\, k = 0 \sim MN/8-1$, $f_0^{(8)}(k) = \{255, 0, 0, 0, 0, 0, 0, 0\}$, i.e.,

$$\boldsymbol{M}_{0,k} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

  It is obvious that the 8 horizontal secret rotations have no influence on the above plain-image. That is, the 2-D TDCEA is reduced to the 1-D BRIE in the vertical direction. Since each column of $\boldsymbol{M}_{0,k}$ has only one 1-bit, by comparing $\boldsymbol{M}_{0,k}$ and $\boldsymbol{M}'_{0,k}$ one can uniquely get 8 values, $s_k(j)$ $(j = 0 \sim 7)$, which satisfy $\boldsymbol{M}'_{0,k} = \mathrm{Rotate}Y_j^{0,s_k(j)}(\boldsymbol{M}_{0,k})$ and serves as the equivalent rotation parameter of the j-th column.

- *Break the 8 horizontal secret rotations*: Choose a plain-image f_1 as follows:

$\forall\, k = 0 \sim MN/8-1$, $f_1^{(8)}(k) = \{1,1,1,1,1,1,1,1\}$, i.e.,

$$\boldsymbol{M}_{1,k} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Since the 8 vertical secret rotations have been obtained via f_0, one can remove all the 8 vertical rotations from $\boldsymbol{M}'_{1,k}$ to get the intermediate bit matrix $\boldsymbol{M}^*_{1,k}$. Then, by comparing $\boldsymbol{M}^*_{1,k}$ and $\boldsymbol{M}_{1,k}$, one can similarly get another 8 values, $r_k(i)$ $(i = 0 \sim 7)$, where $\boldsymbol{M}^*_{1,k} = \text{Rotate}X_i^{0,r_k(i)}(\boldsymbol{M}_{1,k})$. Here, $r_k(0) \sim r_k(7)$ are the equivalent rotation parameter of the i-th line.

Apparently, after revealing the horizontal and vertical secret rotations, the permutation matrix $\boldsymbol{W}_k$ can be immediately reconstructed by simply combining the 16 rotations. In this case, the time complexity is only $O((4+1+4+8)MN) = O(17MN)$.

§2.6 Generalization of The Above Cryptanalyses

In previous sections, it has been shown that permutation-only image cipher working in spatial domain are not secure against known/chosen-plaintext attacks. In this section, the above cryptanalysis results are generalized to permutation-only image ciphers working in frequency domain and also to permutation-only video ciphers and permutation-only speech data ciphers. Since the cryptanalysis procedure is almost identical except for the format of plaintexts and ciphertexts, the following discussions only focus on a rough comparison of the breaking performances in different situations.

§2.6.1 Cryptanalysis of Permutation-Only Image Ciphers Working in Frequency Domain

Many digital images are stored by lossy compression techniques, which generally work in frequency domain, especially in DCT or wavelet domain. Accordingly, when permutation-only image ciphers are used to encrypt such images, the secret permutations are exerted on the transformation coefficients in frequency domain, not on the pixels in the spatial domain. In most transformation-based compression formats, the image is divided into many blocks of smaller size to reduce the time complexity of compression. For example, in DCT-based formats, the image is generally divided into 8×8 blocks; and in wavelet-based formats, the image is generally divided into a quadtree. In this case, the secret permutations can also be exerted on the blocks or the nodes of the tree, i.e., there may exist a hierarchical encryption structure.

Generally speaking, it is easy to directly generalize the above known/chosen-plaintext cryptanalysis, by considering the transformed image $\mathfrak{T}(f)$ as the plain-image f, i.e., considering the transform coefficients as the pixels in spatial domain. The only difference between the two cases are that there exists energy concentration in $\mathfrak{T}(f)$ – generally most significant transform coefficients distribute within low-frequency band. What does this mean for cryptanalysis? Apparently, to achieve an acceptable breaking performance, one can only reconstruct the elements in $\boldsymbol{W}$ and $\boldsymbol{W}^{-1}$ that correspond to low-frequency coefficients. This implies that the reduction of the image size, which immediately leads to a smaller number of required known/chosen plain-images and to the decline of the security against known/chosen-plaintext attacks. In fact, in [74, Sec. 3.4.2], it has been pointed out that the non-uniform distribution of DCT coefficients in MPEG videos (also for JPEG images) can even be used to partially break the secret permutations in *ciphertext-only attacks*. For example, one can correctly locate the DC coefficient of each 8×8 block with a large probability since the DC coefficient generally has the largest amplitude among all 64 DCT coefficients.

As shown in previous sections, the existence of hierarchical structures in compression techniques further reduces the security. What's more, some elements in $\boldsymbol{W}$ and $\boldsymbol{W}^{-1}$ that correspond to high-frequency coefficients can also be determined in the attacks, which can further help refine the visual quality of the recovered plain-image.

As a result, generally permutation-only image ciphers working in frequency domain are less secure against known/chosen-plaintext attack than those working in spatial domain. If it is possible to avoid the energy-concentration property and the hierarchical structure, the security at best will be equivalent to that in the spatial-domain case.

§2.6.2 Cryptanalysis of Permutation-Only Video Ciphers

A video stream is composed of a series of 2-D consecutive images, which are called *frames* of the video. Essentially, permutation-only video ciphers work in the same way on each frame as permutation-only image ciphers. Due to the bulky size of most videos, transform-based lossy compression techniques are widely used for storage and transmission of videos. Also, the block-based or quadtree-based hierarchical structure is widely used in various video formats. So, despite the details of different video formats, the security of most permutation-only video ciphers against known/chosen-plaintext attacks is in the same order as that of permutation-only image ciphers working in frequency domain.

As a result, the security of a video cipher can be evaluated by considering it as an image cipher encrypting the following two types of plain-images: 1) independent frames, such as I-frames in MPEG videos; 2) frames dependent on others, such as B/P-frames in MPEG videos. In such a way, the security analysis of the video cipher becomes simpler and clearer. The major extra consideration in the design of a video cipher is how to make the cipher faster and easier for implementation in the whole video processing system.

Here, note the following fact: if the permutation matrix is fixed for all frames, then only one partially-known/chosen plain-video is enough to reveal the secret permutation matrix. From such a point of view, the security of a permutation-only video cipher may be even weaker than its image counterpart. However, if the permutation matrix has to be changed from frame to frame, it will be more difficult to maintain the fast speed of the video cipher. This is another consideration in the design of a good video cipher.

§2.6.3 Cryptanalysis of Permutation-Only Speech Data Ciphers

The general cryptanalysis of permutation-only image and video ciphers given in this chapter can be easily applied to permutation-only speech data ciphers. In this case, the permutation matrix is of size $1 \times N$. Apparently, permutation-only speech ciphers are just 1-D special cases of permutation-only image/video ciphers, so the above-discussed cryptanalysis still works with the same breaking performance. Also, if the encryption is made in the frequency domain, the energy-concentration effect will expedite the attack in the same way as in the case of permutation-only image/video ciphers working in the frequency domain. Some other existing cryptanalysis work on permutation-only speech data ciphers can be found in, for example, [66, 67].

§2.7 Conclusion

By surveying most permutation-only image ciphers working in the spatial domain and normalizing the encryption and decryption procedures of them, from a general perspective this chapter analyzes the security of such image ciphers against known/chosen-plaintext attacks. When the plain-images have size $M \times N$ with

L possible pixel values, it is found that only $O(\log_L(MN))$ known/chosen plain-images are enough for an attacker to achieve a rather good breaking performance, leading to the conclusion that all permutation-only ciphers are not secure enough against known/chosen-plaintext attacks. Also, it has been found that the attack complexity is practically small – only $O(n\cdot(MN)^2)$, when n plain-images are known or chosen to use. The generalization of above cryptanalyses results to permutation-only image ciphers working in the frequency domain, as well as permutation-only video ciphers and permutation-only speech data ciphers is also discussed.

A recently-proposed permutation-only image cipher, named HCIE, has been studied as a typical example for illustrating the cryptanalysis. Some experiments have been shown to support the cryptanalysis of the general permutation-only multimedia ciphers as well as the specific HCIE. Another multimedia encryption scheme called TDCEA have also been analyzed carefully. It is found that some defects exist in TDCEA and the scheme belongs in the category of permutation-only cipher. The above general attacks are deducted to break it. Experimental results have been given to demonstrate the defects and the feasibility of the proposed attacks. In addition, the security of the two schemes against brute-force attack are also found not sufficient strong.

In summary, secret permutations alone are incapable of providing a sufficiently high level of security against known/chosen-plaintext attacks, so they must be used together with other encryption techniques in the design of highly secure multimedia encryption algorithms. To the best of our knowledge, this is the first time in the literature to quantitatively clarify the security principle on multimedia encryption algorithms, from both theoretical and experimental points of view.

Chapter 3
Cryptanalyses of Some Chaos-Based Encryption Algorithms

§3.1 Introduction

Due to the tight relationship between chaos and cryptography [83, Chap. 2], chaotic systems have been widely used in image encryption to realize diffusion and confusion[11, 16, 18, 21, 32, 33, 35, 37, 41]. For a comprehensive survey of the state-of-the-art of image encryption schemes, see Sec. 4.3 and Sec. 4.4 of [84].

As surveyed in Sec. 4.4.3, J.-C. Yen and J.-I. Guo (et al.) proposed many chaotic multimedia encryption schemes in recent years*. Most of them are based on the following basic idea: using a chaotic map (Logistic map) to generate a secret chaotic PRBS, and the PRBS is then used to control the running and combination of simple permutation and substitution operations. From a strict cryptographical point of view, most Yen-Guo cryptosystems are insecure since known/chosen-plaintext attack can break them with much less complexity than brute force attack. Among all Yen-Guo's schemes, BRIE and CKBA, have been broken successfully in [80] and [81] respectively. In this chapter, we'll give cryptanalyses of another four schemes, RCES(also called RSES), MES, DSEA and TDCEA. Note that TDCEA have been cryptanalyzed in §2.5 as a permutation-only cipher, here it will be broken for another time with a different method.

This chapter is organized as follows. §3.2 briefly introduces three chaos-based encryption schemes, which are all proposed by Yen et al. §3.3, §3.4 and §3.5 give detailed cryptanalyses of RCES, MES, DSEA respectively. §3.6 present another kind of cryptanalysis of TDCEA. The last section concludes the chapter.

§3.2 Some Chaos-Based Encryption Algorithms

§3.2.1 Random Control Encryption System (RCES)[41](or RSES [37])

RCES encrypts plain-images block by block, where each block contains 16 consecutive pixels. To simplify the following description, without loss of generality, assume that the sizes of plain-images are all $M \times N$ (M is the width and N is

*In original literature, the schemes were presented for encrypting image and then recommended for encrypting MPEG video.

the height), and that MN can be exactly divided by 16. Consider a plain-image $\{f(x,y)\}_{x=0,y=0}^{x=M-1,y=N-1}$ as a 1-D pixel-sequence $\{f(l)\}_{l=0}^{MN-1}$ by scanning it line by line from bottom to top. The plain-image can be divided into $MN/16$ blocks:

$$\{f^{(16)}(0),\cdots,f^{(16)}(k),\cdots,f^{(16)}(MN/16-1)\},$$

where

$$f^{(16)}(k)=\{f(16k+0),\cdots,f(16k+i),\cdots,f(16k+15)\}.$$

For the k-th pixel-block $f^{(16)}(k)$, the work mechanism of RCES can be described as follows.

- *The secret key:* the control parameter μ and the initial condition $x(0)$ of the Logistic map Eq. (2.8).
- *Initialization:* run the Logistic map to generate a chaotic sequence, $\{x(i)\}_{i=0}^{MN/16-1}$, and then extract the 24-bit representation of $x(i)$ to yield a PRBS $\{b(i)\}_{i=0}^{3MN/2-1}$. Note that the Logistic map is realized in 24-bit fixed-point arithmetic.
- *Encryption:* two pseudo-random seeds,

$$Seed1(k) = \sum_{i=0}^{7} b(24k+i)\cdot 2^{7-i}, \tag{3.1}$$

$$Seed2(k) = \sum_{i=0}^{7} b(24k+8+i)\cdot 2^{7-i}, \tag{3.2}$$

  are calculated to encrypt the current plain-block with the following two steps:

  – pseudo-randomly swapping adjacent pixels
    for $i=0\sim 7$, do

$$Swap_{b(24k+16+i)}(f(16k+2i),f(16k+2i+1)), \tag{3.3}$$

    where $Swap_w(a,b)=\begin{cases}(a,b), & w=0,\\ (b,a), & w=1;\end{cases}$

  – masking the current plain-block with the two pseudo-random seeds
    for $j=0\sim 15$, do

$$f'(16k+j)=f(16k+j)\oplus Seed(16k+j), \tag{3.4}$$

where

$$Seed(16k+j) = \begin{cases} Seed1(k), & B(k,j) = 3, \\ \overline{Seed1(k)}, & B(k,j) = 2, \\ Seed2(k), & B(k,j) = 1, \\ \overline{Seed2(k)}, & B(k,j) = 0, \end{cases} \quad (3.5)$$

and $B(k,j) = 2 \cdot b(24k+j) + b(24k+j+1)$, $\oplus$ denotes the bitwise XOR operation (the same hereinafter).

- *Decryption:* The decryption procedure is similar to the encryption procedure, but the masking operation is exerted before the swapping for each pixel-block.

§3.2.2 The Multistage Encryption System (MES)

MES encrypts the plaintext block by block, where each block contains 7 plain-bytes. Each 7-byte plain-block is firstly expanded to an 8-byte block by adding a secret pseudo-random byte, and then is encrypted by three different operations: byte permutations, value masking, and bit recirculations, which are all controlled by a secret PRBS generated from the chaotic Logistic map Eq. (2.8).

To facilitate the description of MES, without loss of generality, assume that the plaintext is $f = \{f(i)\}_{i=0}^{N-1}$, where $f(i)$ denotes the i-th plain-byte and N can be exactly divided by 7. In this case, the plaintext has $N/7$ blocks: $f = \{f^{(7)}(k)\}_{k=0}^{N/7-1}$, where $f^{(7)}(k) = \{f^{(7)}(k,j)\}_{j=0}^{6} = \{f(7k+j)\}_{j=0}^{6}$. Similarly, assume that the ciphertext is $f' = \{f'(i)\}_{i=0}^{N-1} = \{f'^{(8)}(k)\}_{k=0}^{N/7-1}$, where $f'^{(8)}(k) = \{f'^{(8)}(k,j)\}_{j=0}^{7} = \{f'(8k+j)\}_{j=0}^{7}$ denotes the expanded cipher-block with 8 bytes. With the above notations, MES can be described as follows.

- *The secret key* three integers α, β, *Open*, the control parameter μ and the initial condition $x(0)$ of the chaotic Logistic map, where $\alpha > 0$, $\beta > 0$, $\alpha + \beta < 8$ and $Open \in \{0, \cdots, 255\}$.

- *The initialization procedure* a) in 33-bit fixed-point finite precision, run the Logistic map from $x(0)$ to generate a chaotic sequence, $\{x(k)\}_{k=0}^{N/7-1}$, and then extract the 33 bits of $x(k) = 0.b_{33k+0} \cdots b_{33k+32}$ to yield a chaotic PRBS, $\{b(i)\}_{i=0}^{33N/7-1}$; b) set $temp = Open$.

- *The encryption procedure of each plain-block $f^{(7)}(k)$ is composed of the following four steps:*

  - data expansion

get an 8-byte block, $f^{(8)}(k) = \{f^{(8)}(k,j)\}_{j=0}^{7} = \{temp, f^{(7)}(k,0),\cdots, f^{(7)}(k,6)\}$, and then set $temp = f^{(8)}(k, l(k))$, where $l(k) = \sum_{i=0}^{2} b(33k+i)\cdot 2^i$.

- byte permutation

  do the random swapping operation, $Swap_{b(33k+l)}\left(f^{(8)}(k,i), f^{(8)}(k,j)\right)$, for 12 times with the following parameters in order: $(i,j,l) = (0,4,3)$, $(1,5,4)$, $(2,6,5)$, $(3,7,6)$, $(0,2,7)$, $(1,3,8)$, $(4,6,9)$, $(5,7,10)$, $(0,1,11)$, $(2,3,12)$, $(4,5,13)$, $(6,7,14)$. Denote the permuted 8-byte block by $f^{*(8)}(k)$.

- random masking

  determine two pseudo-random bytes, $Seed1(k) = \sum_{i=0}^{7} b(33k+i)\cdot 2^{7-i}$ and $Seed2(k) = \sum_{i=0}^{7} b(33k+8+i)\cdot 2^{7-i}$, and then do the following masking operations for $j = 0 \sim 7$:

$$f^{**(8)}(k,j) = f^{*(8)}(k,j) \oplus Seed(k,j), \tag{3.6}$$

  where

$$Seed(k,j) = \begin{cases} Seed1(k), & B(k,j) = 3, \\ \overline{Seed1(k)}, & B(k,j) = 2, \\ Seed2(k), & B(k,j) = 1, \\ \overline{Seed2(k)}, & B(k,j) = 0, \end{cases} \tag{3.7}$$

  and $B(k,j) = 2\cdot b(33k+16+j) + b(33k+17+j)$.

- bit recirculation

  for $j = 0 \sim 7$, do

$$f'(8k+j) = f'^{(8)}(k,j) = ROLR_{p(k,j)}^{q(k,j)}\left(f^{**(8)}(k,j)\right), \tag{3.8}$$

  where $p(k,j) = b(33k+24+j)$, $q(k,j) = \alpha + \beta\cdot b(33k+25+j)$.

- *The decryption procedure is the simple inverse of the above encryption procedure*

  for the k-th 8-byte cipher-block $f'^{(8)}(k)$, Step *d)* is first performed by replacing $p(k,j)$ with its complement $\overline{p(k,j)} = 1 - p(k,j)$, then Step *c)* is performed, and then Step *b)* is performed in the reversed order, finally the first byte is discarded to recover the plain-block $f^{(7)}(k)$.

§3.2.3 Domino Signal Encryption Algorithm (DSEA)

Assume that the plaintext is $g = \{g(n)\}_{n=0}^{M-1}$ and that the ciphertext is $g' = \{g'(n)\}_{n=0}^{M-1}$, where $g(n)$ and $g'(n)$ denote the n-th plain-byte and cipher-byte, respectively. Then, the encryption procedure of DSEA can be described as follows.

- *The secret key*: two integers, $L \in \{1, \cdots, M\}$, $initial\_key \in \{0, \cdots, 255\}$, the control parameter μ and the initial condition $x(0)$ of the chaotic Logistic map Eq. (2.8).
- *The initialization procedure*: under 8-bit finite computing precision, run the Logistic map from $x(0)$ to generate a chaotic sequence $\{x(k)\}_{k=0}^{\lceil M/8 \rceil -1}$, and then extract the 8 significant bits of $x(k)$ to yield a PRBS $\{b(n)\}_{n=0}^{M-1}$, where $x(k) = \sum_{i=0}^{7} \left(b_{8k+i} \cdot 2^{-(i+1)} \right) = 0.b_{8k+0} \cdots b_{8k+7}$.
- *The encryption procedure*: for $n = 0 \sim M-1$, do

$$g'(n) = \begin{cases} g(n) \oplus true\_key, & b(n) = 1, \\ g(n) \oplus \overline{true\_key}, & b(n) = 0, \end{cases}$$

  where

$$true\_key = \begin{cases} initial\_key, & n \bmod L = 0. \\ g'(n-1), & n \bmod L \neq 0. \end{cases}$$

- *The decryption procedure* is identical with the above encryption procedure, since XOR is an invertible operation.

§3.3 Cryptanalysis of RCES

In this section, we analyze the security of RCES in detail and the following results are obtained: 1) its security against brute-force attack was over-estimated; 2) it is not secure against known/chosen-plaintext attacks, and the number of required plain-images is only $O(1)$ and, in fact, only one or two; 3) there are two available known/chosen-plaintext attacks, and they can be further combined to make a nearly-perfect attack to RCES; 4) the chosen-plaintext attacks can even achieve much better breaking performance than their known-plaintext versions.

§3.3.1 The Brute-Force Attack

In [37, 41], Chen and Yen claimed that the complexity of RCES against brute-force attack is $O\left(2^{3MN/2}\right)$ since $\{b(i)\}_{i=0}^{3MN/2-1}$ has $3MN/2$ bits. However, such a statement is not true due to the following reason: all $3MN/2$ bits are uniquely determined by the control parameter μ and the initial condition $x(0)$ of the Logistic map Eq. (2.8), which has only 48 secret bits. This means that the key entropy of RCES is only 48. Considering not all values of μ can produce chaoticity in the Logistic map, the key entropy should be even smaller than 48. To simplify the following analysis, assume that the key entropy is $K_\mu < 48$, so the total number of all possible keys for brute-force search is only 2^{K_μ}.

Considering that the complexity of RCES is $O(MN)$ [41, Sec. 2.4], the complexity against the brute-force attack is $O\left(2^{K_\mu} \cdot MN\right)$. Assume $K_\mu = 48$, for a typical image whose size is 256×256, the complexity is about $O\left(2^{64}\right)$, which is much smaller than $O\left(2^{3MN/2}\right) = O\left(2^{98304}\right)$, the claimed complexity in [37, 41]. Apparently, the security of RCES against the brute-force attack was over-estimated by too much.

§3.3.2 Known-Plaintext Attack 1: Breaking RCES with a Mask Image f_m

With only one known plain-image and its corresponding cipher-image, it is very easy to get a mask image f_m, which can be used as an equivalent key of the secret key $(\mu, x(0))$ to decrypt any cipher-image whose size is not larger than the size of f_m. When two or more plain-images are known, a swapping matrix Q can be constructed to enhance the breaking performance of the mask image f_m.

§3.3.2.1 Get f_m from One Known Plain-Image

Assume that an $M \times N$ plain-image f_K and its corresponding cipher-image f'_K have been known to an attacker. Then he can get f_m by simply XORing the plain-image and the cipher-image pixel by pixel: $f_m(l) = f_K(l) \oplus f'_K(l)$, where $l = 0 \sim MN - 1$.

With the mask image f_m, the attacker tries to recover the plain-image by XORing the mask image and the cipher-image pixel by pixel: $f(l) = f'(l) \oplus f_m(l)$. If a pixel $f(l)$ is not swapped, $f(l) = f'(l) \oplus f_m(l)$ holds; otherwise, $f(l) = f'(l) \oplus f_m(l)$ is generally not true. Assume that the bit $b(24k + 16 + i)$ in Eq. (3.3) satisfies the balanced distribution* over $\{0, 1\}$, it is expected that about half of all plain-pixels are not swapped and can be successfully decrypted with $f_m \oplus f'$. Intuitively, half of plain-pixels should be enough to reveal the main content and some details of the plain-image.

*Strictly speaking, the Logistic map cannot guarantee the balance of each generated bit, since its variant density function is not uniform [85]. In this chapter, without loss of generality, it is taken for granted so as to simplify the theoretical analyses.

With the secret key $(\mu, x(0)) = (3.915264, 0.2526438)$, which is randomly chosen with the standard `rand()` function, some experiments are made to show the real performance of the mask image f_m in this attack. One known plain-image f_{Lenna} and its cipher-image f'_{Lenna} are shown in Fig. 3.1. The mask image $f_m = f_{\text{Lenna}} \oplus f'_{\text{Lenna}}$ is given in Fig. 3.2. For an unknown plain-image f_{Peppers} (Fig. 3.3a), the mask image f_m is used to recover it from its cipher-image f'_{Peppers} (Fig. 3.3b). The recovered plain-image $f^*_{\text{Lenna}} = f_m \oplus f'_{\text{Peppers}}$ and the difference image $\left|f^*_{\text{Peppers}} - f_{\text{Peppers}}\right|$ are shown in Fig. 3.4a and 3.4b, respectively. It is surprisingly seen that the decryption performance is much better than expected: most (much more than 50%) pixels are successfully recovered, and almost all subtle details remain.

a) The known plain-image f_{Lenna} b) The cipher-image f'_{Lenna}

Figure 3.1: One known plain-image, 'Lenna' (256×256), and its cipher-image

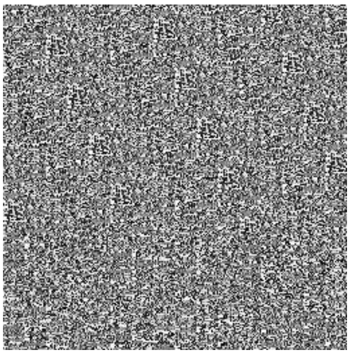
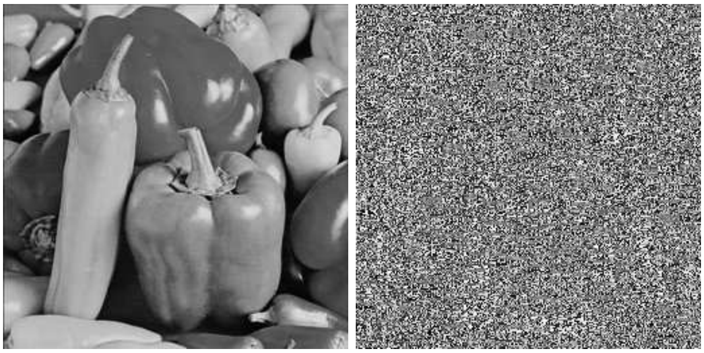


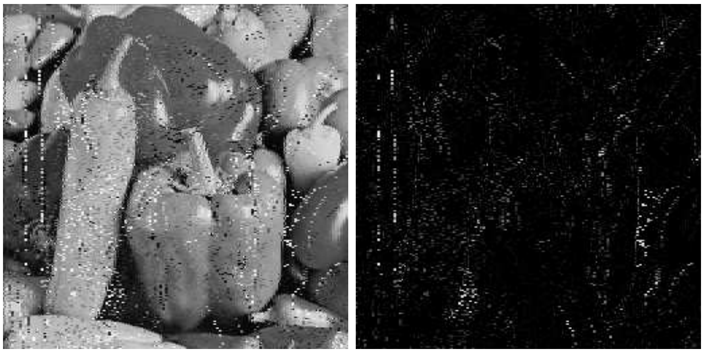
Figure 3.2: The mask image f_m derived from f_{Lenna} and f'_{Lenna}

Although the difference $\left|f^*_{\text{Peppers}} - f_{\text{Peppers}}\right|$ visually shows that most plain-pixels are exactly recovered, statistical data show that 33,834 pixels in $f^*_{\text{Peppers}} -$



a) The plain-image f_{Peppers} b) The cipher-image f'_{Peppers}

Figure 3.3: A plain-image unknown to the attacker, 'Peppers' (256×256), and its cipher-image



a) The recovered plain-image f^*_{Peppers} with f_m b) The recovery difference $\left|f^*_{\text{Peppers}} - f_{\text{Peppers}}\right|$

Figure 3.4: Breaking the plain-image with f_m derived from Lenna.bmp

f_{Peppers} are not zero, i.e., about 51.63% of pixels are not exactly recovered. To explain why f_m is so effective to recover most pixels of the plain-image with only half exactly-recovered pixels, consider two pixels in the known plain-image, $f(2i)$, $f(2i+1)$, and their cipher-pixels, $f'(2i)$, $f'(2i+1)$, where $i = 0 \sim MN/2-1$. Then, the corresponding elements of the two pixels in the mask image f_m will be $f_m(2i) = f(2i) \oplus f'(2i)$ and $f_m(2i+1) = f(2i+1) \oplus f'(2i+1)$. Since all recovery errors are introduced at the positions where the adjacent plain-pixels are swapped, one can theoretically study the recovery performance of the mask image f_m by considering the elements corresponding to the swapped pixels only. Assume that $f(2i)$ and $f(2i+1)$ are swapped in the encryption procedure, $f'(2i) = f(2i+1) \oplus Seed(2i)$ and $f'(2i+1) = f(2i) \oplus Seed(2i+1)$. Therefore,

$$f_m(2i) \quad = \quad f^{(\oplus)}(2i) \oplus Seed(2i), \tag{3.9}$$

$$f_m(2i+1) = f^{(\oplus)}(2i) \oplus Seed(2i+1), \tag{3.10}$$

where $f^{(\oplus)}(2i) = f(2i) \oplus f(2i+1)$.

Consider a cipher-image f_1' and its corresponding plain-image f_1. Assuming that the plain-image recovered from f_m is f_1^*, the recovered plain-pixels, $f_1^*(2i)$ and $f_1^*(2i+1)$, satisfy the following proposition and corollaries.

Proposition 3.1: $f_1^*(2i) \oplus f_1(2i) = f_1^*(2i+1) \oplus f_1(2i+1) = f^{(\oplus)}(2i) \oplus f_1^{(\oplus)}(2i)$.

Proof: From Eq. (3.9) and $f_1'(2i) = f_1(2i+1) \oplus Seed(2i)$,

$$\begin{aligned} f_1^*(2i) &= f_m(2i) \oplus f_1'(2i), \\ &= \left(f^{(\oplus)}(2i) \oplus Seed(2i)\right) \\ &\quad \oplus (f_1(2i+1) \oplus Seed(2i)) \\ &= f^{(\oplus)}(2i) \oplus f_1(2i+1) \end{aligned}$$

Then, one has

$$\begin{aligned} f_1^*(2i) \oplus f_1(2i) &= f^{(\oplus)}(2i) \oplus f_1(2i+1) \oplus f_1(2i) \\ &= f^{(\oplus)}(2i) \oplus f_1^{(\oplus)}(2i). \end{aligned}$$

In a similar way, one can get $f_1^*(2i+1) \oplus f_1(2i+1) = f^{(\oplus)}(2i) \oplus f_1^{(\oplus)}(2i)$. Thus, the proof is completed. ■

Corollary 3.1: *When* $f(2i) = f(2i+1)$, $f_1^*(2i) = f_1(2i+1)$ *and* $f_1^*(2i+1) = f_1(2i)$.

Proof: The results of this corollary are special cases of the above two propositions with $f^{(\oplus)}(2i) = 0$. ■

Based on the above proposition, one can get an upper bound of the recovery errors $|f_1^*(2i) - f_1(2i)|$ and $|f_1^*(2i+1) - f_1(2i+1)|$. Firstly, a lemma should be introduced.

Lemma 3.1: *If* $a \oplus b = c$, *then* $|a - b| \leq c$.

Proof: Represent c in the following binary form:

$$c = (0, \cdots, 0, c_{n-1} = 1, \cdots, c_i, \cdots, c_1, c_0)_2.$$

Similarly, represent a and b as follows:

$$a = (a_{N-1}, \cdots, a_{n-1}, \cdots, a_i, \cdots, a_1, a_0)_2,$$

$$b = (b_{N-1}, \cdots, b_{n-1}, \cdots, b_i, \cdots, b_1, b_0)_2.$$

From $a \oplus b = c$, one have $\forall j = n \sim N-1$, $a_j = b_j$. Therefore,

$$|a-b| = |\sum_{i=0}^{N-1}(a_i - b_i)\cdot 2^i| = |\sum_{i=0}^{n-1}(a_i - b_i)\cdot 2^i| \leq \sum_{i=0}^{n-1}|a_i - b_i|\cdot 2^i.$$

Since $|a_i - b_i| = a_i \oplus b_i = c_i$, one has $|a-b| \leq \sum_{i=0}^{n-1} c_i \cdot 2^i = c$. The lemma is thus proved. ■

Corollary 3.2: *$|f_1^*(2i) - f_1(2i)| \leq f^{(\oplus)}(2i) \oplus f_1^{(\oplus)}(2i)$, and $|f_1^*(2i+1) - f_1(2i+1)| \leq f^{(\oplus)}(2i) \oplus f_1^{(\oplus)}(2i)$.*

Proof: This corollary is an obvious result of Proposition 3.1 and Lemma 3.1. ■

Corollary 3.2 says that the recovery errors of both $f_1^*(2i)$ and $f_1^*(2i+1)$ will not be larger than $f^{(\oplus)}(2i) \oplus f_1^{(\oplus)}(2i) = f(2i) \oplus f(2i+1) \oplus f_1(2i) \oplus f_1(2i+1)$. Due to the strong correlation between adjacent pixels of digital images, the distribution of the difference between two adjacent pixels is Gaussian-like. As a result, $f^{(\oplus)}(2i)$ will also obeys a (positive) single-side Gaussian-like distribution, which means that the recovery error of each plain-pixel recovered from f_m will also obey a Gaussian-like distribution. The Gaussian-like distribution of recovery errors actually implies that most recovered pixels are close to the real values of the original plain-pixels. Therefore, the surprising recovery performance of f_m shown in Fig. 3.4 can be naturally explained.

For the plain-image f_{Peppers}, the histograms of some differential images are plotted to verify the above-mentioned theoretical results. Define two $(M-1) \times N$ differential images $f^{(-)}$ and $f^{(\oplus)}$:

$$f^{(-)}(x, y) = f(x, y) - f(x+1, y), \tag{3.11}$$

$$f^{(\oplus)}(x, y) = f(x, y) \oplus f(x+1, y), \tag{3.12}$$

where $x = 0 \sim M-2$, $y = 0 \sim N-1$. The histograms of the above two differential images of f_{Peppers} are shown in Fig. 3.5. When $f = f_{\text{Lenna}}$, $f_1 = f_{\text{Peppers}}$, the histograms of $f^{(\oplus)} \oplus f_1^{(\oplus)}$ and $\left|f_{\text{Peppers}}^* - f_{\text{Peppers}}\right|$ are shown in Fig. 3.6. Apparently, Figure 3.6 agrees with Corollary 3.2 very well. Note that only the swapped pixels are enumerated for the histogram of $\left|f_{\text{Peppers}}^* - f_{\text{Peppers}}\right|$, since the above theoretical analysis on the recovery errors is only focused on the swapped pixels.

Since all recovery errors are introduced by swapped pixels, the recovery performance will be better if some swapped pixels can be distinguished. In the following, it is shown that an attacker can manage to do so by manually detecting visible noises in cipher-images, and by intersecting multiple mask images generated from different known plain-images.

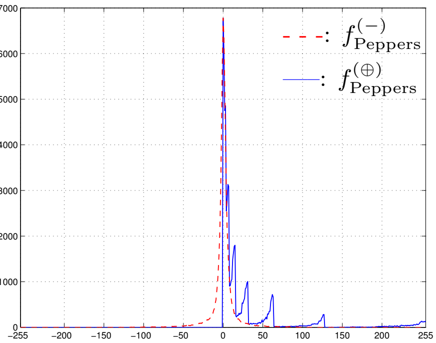


Figure 3.5: The histograms of $f^{(-)}_{\text{Peppers}}$ and $f^{(\oplus)}_{\text{Peppers}}$

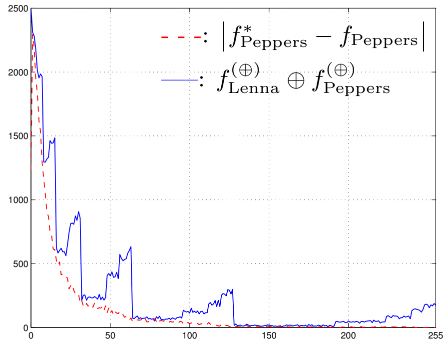


Figure 3.6: The histograms of $f^{(\oplus)}_{\text{Lenna}} \oplus f^{(\oplus)}_{\text{Peppers}}$ and $\left|f^{*}_{\text{Peppers}} - f_{\text{Peppers}}\right|$

§3.3.2.2 Amending f_m with More Cipher-Images

Assume that the corresponding plain-image of a cipher-image does not contain salt-pepper impulsive noises. Then, one can assert that all such noises in the recovered plain-image indicates the positions of swapped pixels. Observing the recovered plain-image f^{*}_{Lenna} shown in Fig. 3.4a, one can find many distinguishable noises by naked eyes, which correspond to the strong edges of the known plain-image f_{Lenna} (see Fig. 3.4b). Following Proposition 3.1, strong edges means large values of $f^{(\oplus)}(x)$, and so generates salt-pepper noises.

Once some swapped pixels are distinguished, one can generate a swapping (0, 1)-matrix $Q = [q_{i,j}]_{M \times N}$, where $q_{i,j} = 1$ for swapped pixels and $q_{i,j} = 0$ for others. Similarly, Q can be represented in 1-D form: $Q = \{q(l)\}_{i=0}^{MN-1}$. With the swapping matrix, the mask image f_m is amended as follows: for $i = 0 \sim MN/2-1$, if $q(2i) = 1$ or $q(2i+1) = 1$, the values of $f_m(2i)$ and $f_m(2i+1)$ are re-calculated as follows: $f_m(2i) = f(2i) \oplus f'(2i+1)$ and $f_m(2i+1) = f(2i+1) \oplus f'(2i)$; otherwise, $f_m(2i)$ and $f_m(2i+1)$ are left untouched. With the amended f_m and the swapping matrix Q, one can decrypt the cipher-images in the following two steps:

- use f_m to XOR the cipher-image to get an initial recovered plain-image f^*;
- $\forall i = 0 \sim MN/2 - 1$, if $q(2i) = 1$ or $q(2i+1) = 1$, swap the two adjacent pixels $f^*(2i)$ and $f^*(2i+1)$.

If an attacker can get more cipher-images encrypted with the same key, he can distinguish more swapped pixels, and gets better recovery performance with f_m and Q. This implies that more and more knowledge on how to purify the attack can be learned from the cipher-images, which is a very interesting and useful feature from an attacker's point of view.

§3.3.2.3 Amending f_m with More Known Plain-Images

With two or more known plain-images and their cipher-images encrypted with the same secret key, it is possible to successfully distinguish most swapped pixels, achieving nearly perfect recovery performance. Given $n \geq 2$ known plain-images, $f_1, \cdots, f_n$, and their cipher-images, $f_1', \cdots, f_n'$, one can get n mask images $f_m^{(i)} = f_i \oplus f_i'$ ($i = 1 \sim n$). Apparently, if the l-th pixel is not swapped, $\forall i \neq j$, $f_m^{(i)}(l) = f_m^{(j)}(l)$. That is, if $f_m^{(i)}(l) \neq f_m^{(j)}(l)$, it can be asserted that the pixel at this position is swapped. Therefore, by comparing the elements of n mask images, some positions corresponding to the swapped pixels can be distinguished. With the swapping information, following the same way described above, a swapping matrix Q can be constructed, and then f_m is amended with Q with the way mentioned above. Using the amended f_m and the swapping matrix Q, the cipher-image is decrypted with XOR and swapping operations.

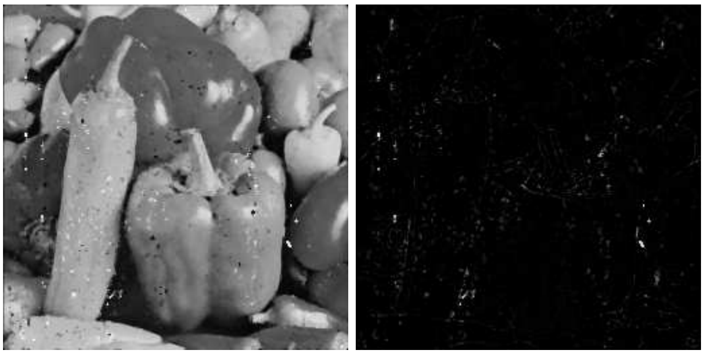
From Eqs. (3.9) and (3.10), the probability of $f_m^{(i)}(l) \neq f_m^{(j)}(l)$ is the probability of $f_i^{(\oplus)}(2i) \neq f_j^{(\oplus)}(2i)$, where $l = 2i$ or $2i+1$. Assume the n mask images are independent of each other and the value of each element distributes uniformly over $\{0, \cdots, 255\}$. The probability of $f_m^{(i)}(l) \neq f_m^{(j)}(l)$ will be $1 - 256^{-1} \approx 0.996$. This means that only two mask images are enough to distinguish almost all swapped pixels. However, since the mask images are generally not independent of each other and $f_m(l)$ does not obey uniform distribution, the real probability will be less than $1 - 256^{-1}$. Fortunately, for most natural images, this probability is still sufficiently close to $1 - 256^{-1}$, so that two known plain-images are still enough to distinguish most swapped pixels. Given two known plain-images, 'Lenna' (see Fig. 3.1a) and 'Babarra' (see Fig. 3.7a), the recovery performance of the attack to 'Peppers' is shown in Fig. 3.7b. It can be seen that the recovered plain-image is almost perfect, and only 952 (about 1.45% of all) pixels are not exactly recovered.

a) Another known plain-image f_{Babarra} b) The recovered plain-image f^{**}_{Peppers}

Figure 3.7: The recovery performance on 'Peppers' with two known plain-images: 'Lenna' and 'Babarra' (both 256×256)

§3.3.2.4 Enhancing the Recovered Plain-Image with Image Processing Techniques

To further improve the visual quality of the recovered plain-images, some noise reducing techniques can be used to further reduce the recovery errors. For the recovered plain-image f^*_{Lenna} in Fig. 3.4a, the enhanced plain-image f^*_{Lenna} with a 3×3 median filter and the corresponding difference image $\left|f^*_{\text{Peppers}} - f_{\text{Peppers}}\right|$ are shown in Fig. 3.8a and 3.8b, respectively. It can be seen that the visual quality of f^*_{Lenna} is enhanced significantly. Note that more complicated image processing techniques are still available to further polish the recovered plain-image, one of which will be introduced below in §3.3.5.



a) The plain-image f^*_{Peppers} enhanced with a 3×3 median filter b) The difference image $\left|f^*_{\text{Peppers}} - f_{\text{Peppers}}\right|$

Figure 3.8: Enhancing the recovered plain-image with a 3×3 median filter

§3.3.3 Known-Plaintext Attack 2: Breaking the Chaotic Map

In the above-discussed attack based on mask images, assuming that the size of f_m is $M \times N$, it is obvious that only MN leading pixels in a larger cipher-image can be recovered with f_m (and perhaps Q). To decrypt more pixels, the secret control parameter μ and a chaotic state $x(k)$ occurring before $x(MN/16-1)$ have to be known, so that one can calculate more chaotic states after $x(MN/16-1)$. That is, the chaotic map should be found. Actually, it is possible for an attacker to achieve this goal with a high probability and a sufficiently small complexity, even when only one plain-image is known. Similarly, the more the number of known plain-images are, the closer the probability will be to 1, the smaller the value of k will be, and the lower the attack complexity will be.

§3.3.3.1 Guessing a Chaotic State $x(k)$ from f_m

In the k-th pixel-block, for any unswapped pixel $f(16k+j)$,

$$f_m(16k+j) = f(16k+j) \oplus f'(16k+j) = Seed(16k+j),$$

which must be one value in the set

$$S_4 = \left\{ Seed1(k), \overline{Seed1(k)}, Seed2(k), \overline{Seed2(k)} \right\}. \tag{3.13}$$

Therefore, if there are enough unswapped pixels, the right values of $Seed1(k)$ and $Seed2(k)$ can be guessed by enumerating all 2-value and 1-value* combinations of $f_m(16k+0) \sim f_m(16k+15)$. To eliminate most wrong values of $Seed1(k), Seed2(k)$, the following requirements are useful:

- both $B(k,j)$ and $(Seed1(k), Seed2(k))$ are generated with $\{b(24k+j\}_{j=0}^{15}$;
- $Seed(16k+j)$ is uniquely determined by $B(k,j)$ and $Seed1(k), Seed2(k)$ following Eq. (3.5).

For each guessed values passing the above requirements, the corresponding chaotic state $x(k) = 0.b(24k+0)\cdots b(24k+23)$ is derived as follows:

- reconstruct $\{b(24k+i)\}_{i=0}^{15}$ from $Seed1(k)$, $Seed2(k)$;
- reconstruct $\{b(24k+16+i)\}_{i=0}^{7}$ with the following rule: if both $f_m(16k+2i) \in S_4$ and $f_m(16k+2i+1) \in S_4$ hold, $b(24k+16+i) = 0$, else $b(24k+16+i) = 1$.

*The 1-value combinations are included since $Seed1(k) = Seed2(k)$ may occur with a small probability.

Note that some extra errors will be introduced in the least 8 bits $\{b(24k+16+i)\}_{i=0}^{7}$, which makes the derived chaotic state $x(k)$ incorrect. Apparently, the errors are induced by the swapped pixels whose corresponding elements of f_m belong to S_4. In the following, the probability of such errors, $p_{se} = Prob\left[f_m(l) \in S_4\right]$, is studied. For any swapped pixel $f(l)$ in the k-th pixel-block ($l = 16k+0 \sim 16k+15$), according to Eqs. (3.9) and (3.10), one has

$$p_{se} = Prob\left[f^{(\oplus)}(l) \in S_4^{(\oplus)}\right] \tag{3.14}$$

where $f^{(\oplus)}(l) = f\left(2\lfloor l/2 \rfloor\right) \oplus f\left(2\lfloor l/2 \rfloor + 1\right)$ and

$$\begin{aligned} S_4^{(\oplus)} \quad = \quad & \Big\{ Seed1(k) \oplus Seed(l), \overline{Seed1(k)} \oplus Seed(l), \\ & Seed2(k) \oplus Seed(l), \overline{Seed2(k)} \oplus Seed(l) \Big\}. \end{aligned}$$

Considering the Gaussian-like distribution of $f^{(\oplus)}$ (see Fig. 3.5) and the fact that $0 \in S_4^{(\oplus)}$, p_{se} is generally not negligible for natural images. Without loss of generality, assume that each bit in $\{b(i)\}$ yields a balanced distribution over $\{0, 1\}$ and any two bits are independent of each other. One can deduce

$$P_1 = Prob[x(k) \text{ is correct}] = \sum_{i=0}^{8} p_b(8, i) \cdot p_c^i, \tag{3.15}$$

where $p_b(8, i) = \binom{8}{i} \cdot 2^{-8}$, which denotes the probability that there are i pairs of swapped pixels, and $p_c = 1 - p_{se}$. The relation between P_1 and p_c is given in Fig. 3.9.

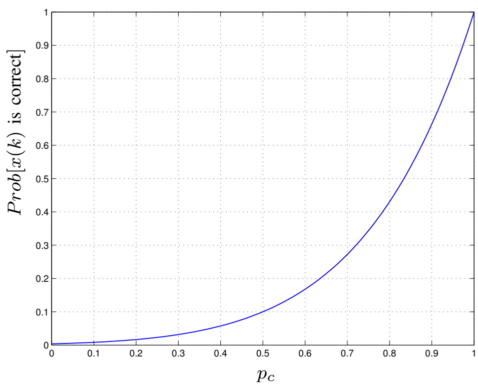


Figure 3.9: $P_1 = Prob[x(k) \text{ is correct}]$ vs. p_c

§3.3.3.2 Deriving μ from Two Consecutive Chaotic States

With two consecutive chaotic states, $x(k)$ and $x(k+1)$, the estimated value of the secret control parameter μ will be $\widetilde{\mu}_k = \dfrac{x(k+1)}{x(k)\cdot(1-x(k))}$. Due to the negative influence of quantization errors, generally $\widetilde{\mu}_k \neq \mu$. As known, chaotic maps are sensitive to fluctuation in the initial condition, so an approximate value of μ will generate completely different chaotic states after several iterations, which implies that $\widetilde{\mu}_k$ can not be directly used instead of μ as the secret key. Fortunately, following the error analysis of $\widetilde{\mu}_k$ given in the Theorem 3.1, one have when $x(k+1) \geq 2^{-n}$ ($n = 1 \sim 24$), $|\widetilde{\mu}_k - \mu| < 2^{n+3} \cdot 2^{-24}$. Specially, when $x(k+1) \geq 2^{-1} = 0.5$, one can exhaustively search $2^{1+3} = 16$ values in the neighborhood of $\widetilde{\mu}_k$ to find the right value of μ. To verify which guessed value of μ is the right one, one should iterate the Logistic map from $x(k+1)$ until $x(MN/16-1)$, and then check whether or not the corresponding elements in f_m match the calculated chaotic states. Once a mismatch occurs, the current guessed value is discarded, and the next guess will be tried. To minimize the verification complexity, one can check only a number of chaotic states sufficiently far from $x(k+1)$ to eliminate most (or even all) wrong values of $\widetilde{\mu}_k$, and verify the left few ones by checking all chaotic states from $x(k+2)$ to $x(MN/16-1)$.

Theorem 3.1: *The estimation error of* $\widetilde{\mu} = \dfrac{x(k+1)}{x(k)\cdot(1-x(k))}$, $\Delta\mu = \widetilde{\mu} - \mu$, *satisfies* $|\Delta\mu| < \frac{1}{2^{L-3}\cdot x(k+1)}$.

Proof: Obviously, the estimation error of $\widetilde{\mu}$ is caused by the quantization error $\Delta x(k+1)$ generated in the forward chaotic iteration $x(k+1) = \mu\cdot x(k)\cdot(1-x(k))$. In L-bit fixed-point finite precision, the quantization error does not exceed 2^{-L} for the floor or ceiling quantization function, and does not exceed $2^{-(L+1)}$ for the round quantization function. Since there are two L-bit digital multiplications in each forward chaotic iteration, one has

$$\begin{aligned}\overline{x(k+1)} &= (\mu\cdot x(k) + \Delta_1 x(k+1))\cdot(1-x(k)) + \Delta_2 x(k+1)\\ &= \mu\cdot x(k)\cdot(1-x(k)) + \Delta_1 x(k+1)\cdot(1-x(k)) + \Delta_2 x(k+1)\\ &= x(k+1) + \Delta x(k+1),\end{aligned}$$

where $\overline{x(k+1)}$ is the real value of $x(k+1)$ and $\Delta x(k+1) = \Delta_1 x(k+1)\cdot(1-x(k)) + \Delta_2 x(k+1)$. Then, one can get $|\Delta x(k+1)| \leq |\Delta_1 x(k+1)| + |\Delta_2 x(k+1)| < 2^{-L} + 2^{-L} = 2^{-(L-1)}$. From this result, the quantization error $|\Delta\mu|$ can be estimated as follows:

$$|\Delta\mu| = \left|\frac{\Delta x(k+1)}{x(k)\cdot(1-x(k))}\right| = \left|\frac{\Delta x(k+1)}{x(k+1)}\cdot\frac{x(k+1)}{x(k)\cdot(1-x(k))}\right|$$

$$= \frac{|\Delta x(k+1)|}{x(k+1)} \cdot \mu < \frac{2^{-(L-1)}}{x(k+1)} \cdot 4 = \frac{1}{2^{L-3} \cdot x(k+1)}.$$

■

Combining the above analyses, the final complexity of finding two correct consecutive chaotic states, $x(k)$, $x(k+1)$, and the right value of μ, is

$$O\left(\frac{2 \cdot \left(\binom{16}{2} + \binom{16}{1}\right)}{(0.5 \cdot P_1)^2} \cdot 2^{1+3}\right) = O\left(\frac{17408}{P_1^2}\right), \tag{3.16}$$

which is generally much smaller than the complexity of exhaustively searching all possible keys. As a reference value, when $p_c = 0.7$, the complexity is about $O\left(2^{17.8}\right) \ll O\left(2^{48}\right)$.

§3.3.3.3 A Quick Algorithm to Guess the Two Random Seeds

Following the above-discussed search process, the found correct chaotic states $x(k)$ and $x(k+1)$ will be close to $x(0)$. Considering the occurrence of two consecutive chaotic states larger than 0.5 as a Bernoulli experiment, the mathematical expectation of k will be $\dfrac{1}{(0.5 \cdot P_1)^2} = \dfrac{4}{P_1^2}$ [86]. This means that only tens of known plain-pixels* are enough for an attacker to break the chaotic map, which is a very desired feature for attackers. However, as an obvious disadvantage, the search complexity to guess the two random seeds is somewhat large. In fact, for each pixel-block, one can only test a few number of possible 2-value (and 1-value) combinations, not all. If this pixel-block looks not good for guessing the two random seeds, simply discard it and go to the next pixel-block. Following such an idea, a quicker algorithm can be designed to find the two random seeds. In this quick-search algorithm, the found correct chaotic states $x(k)$ and $x(k+1)$ may be far from $x(0)$, so the size of the mask image has to be much larger than $\dfrac{4}{P_1^2}$.

The quick-search algorithm is based on the following observation: the more the unswapped pixels there are in the k-th pixel-block, the more the number of values in $\{f_m(16k+j)\}_{j=0}^{15}$ will be in S_4. Accordingly, define a new sequence $\left\{\widetilde{f}_m(16k+j)\right\}_{j=0}^{15}$ as follows:

$$\widetilde{f}_m(16k+j) = \min\left(f_m(16k+j), \overline{f_m(16k+j)}\right). \tag{3.17}$$

Then, the following is also true: the more the unswapped pixels there are in the k-th pixel-block, the more the number of the values in $\left\{\widetilde{f}_m(16k+j)\right\}_{j=0}^{15}$ will be in S_2, where $S_2 = \left\{\min\left(Seed1(k), \overline{Seed1(k)}\right), \min\left(Seed2(k), \overline{Seed2(k)}\right)\right\}$.

*For example, even a 10×10 "tiny" image is enough.

Therefore, assuming that there are n_k pairs of unswapped pixels in the k-th pixel-block, the following fact is true: if n_k is sufficiently large, the two most-occurring elements in $\left\{\widetilde{f}_m(16k+j)\right\}_{j=0}^{15}$ are the two values in S_2, with a high probability. Then, when can one say that n_k is sufficiently large? In totally 8 pairs of elements, the average number of pairs in S_2 is $N(S_2) = n_k + (8 - n_k) \cdot p_{se}$, and the number of other pairs is $N(\overline{S_2}) = 8 - N(S_2) = (8 - n_k) \cdot (1 - p_{se})$. From a conservative point of view, let $N(\overline{S_2}) < \dfrac{N(S_2)}{2}$, which ensures that the occurring probability of each element of S_2 is larger than the probability of all other values. Solving this inequality, one can get $n_k \geq 6$, yielding $N(\overline{S_2}) \leq 2 < 3 \leq \dfrac{N(S_2)}{2}$.

Based on the above analyses, the quick-search algorithm is described as follows:

- *Step 1*: generate a new sequence, $\left\{\widetilde{f}_m(16k+j)\right\}_{j=0}^{15}$;
- *Step 2*: rank all values of $\left\{\widetilde{f}_m(16k+j)\right\}_{j=0}^{15}$ to find the top two mostly-occurring values, $value1$ and $value2$, assume their numbers are $num1$ and $num2$ respectively;
- *Step 3*: if $num1+num2 \geq 12$ and $num1, num2 \geq 3$, continue next step, else $k = k+1$, go to *Step 1*;
- *Step 4*: exhaustively search $Seed1(k)$ and $Seed2(k)$ in $\widetilde{S}_4 = \left\{value1, \overline{value1}, value2, \overline{value2}\right\}$.

If more than one value corresponds to the same position in the rank of $\left\{\widetilde{f}_m(16k+j)\right\}_{j=0}^{15}$, all of them should be enumerated as $value1$ and $value2$ in Step 2 to Step 3. In a real attack, some extra constraints such as secret bit reuse can be added to further optimize the above algorithm for different mask images. The attack complexity of this quick-search algorithm is hard to theoretically analyzed, since the distribution of those values that are not in S_4 is generally unknown. Fortunately, experiments show that the complexity is much smaller than the one given above. In Fig. 3.10, the performance of the quick-search algorithm is shown for the recovered plain-image f^*_{Peppers}, where different pixel-blocks are used to extract the chaotic states. Note that more than forty pixel-blocks are eligible to be used to extract the correct chaotic states, and the three shown here are randomly chosen for demonstration.

In the following, it is theoretically studied as how much MN should be to guarantee the efficiency of the quick-search algorithm, which is determined by the occurrence probability that two consecutive pixel-blocks satisfy the requirements

a) The recovered plain-image f^*_{Peppers} from the 7th pixel-block

b) The recovered plain-image f^*_{Peppers} from the 689th pixel-block

c) The recovered plain-image f^*_{Peppers} from the 1673rd pixel-block

d) Recovering a larger plain-image $f^*_{\text{Peppers2},768\times 768}$ from the 1673rd pixel-block

Figure 3.10: Demonstration of the quick-search algorithm, where 'Lenna' is the only known plain-image

given in Step 2 and Step 3. Assume that each bit in $\{b(i)\}$ yields a balanced distribution over $\{0,1\}$ and any two bits are independent of each other. The probability that one pixel-block satisfies the requirements, which is denoted by P_o, yields Eq. (3.18). Then, for the occurrence probability that two consecutive pixel-blocks satisfy the requirements, which is denoted by P_{o2}, one can calculate that $P_{o2} = P_o^2 \geq Prob\left[S_4 = \widetilde{S}_4\right]^2 = \left(\frac{4699}{2^{15}}\right)^2 \approx 0.02$. This means that there will be two consecutive pixel-blocks satisfy the requirements in $\frac{1}{P_{o2}} \approx 50$ pixel-blocks (about 800 pixels), from the probabilistic point of view. Therefore, the required size of the known plain-image should be larger than 800, which is even smaller than the size of a 30×30 image. Hence, the quick-search algorithm is very efficient to use for attacks.

$$
\begin{aligned}
& P_o \geq Prob\left[S_4 = \widetilde{S}_4\right] \\
= \quad & Prob\left[Seed1(k) \text{ and } Seed2(k) \text{ occur at least 3 times in } \left\{\widetilde{f}_m(16k+j)\right\}_{j=0}^{15}\right] \\
& \cdot Prob\left[\min\left(Seed1(k), \overline{Seed1(k)}\right) \neq \min\left(Seed2(k), \overline{Seed2(k)}\right)\right] \qquad (3.18) \\
= \quad & \textstyle\sum_{n_k=6}^{8} \left(\binom{8}{n_k} \cdot 2^{-8} \cdot \left(1 - \sum_{m=0}^{2} \binom{2n_k}{m} \cdot 2^{-2n_k}\right) \cdot \left(1 - 128^{-1}\right)\right)
\end{aligned}
$$

§3.3.3.4 Breaking the Chaotic Map with both f_m and Q

All the above-mentioned algorithms are based on only-one known plain-image. When more than one plain/cipher-image is known, the constructed swapping $(0,1)$-matrix Q will be very useful to increase the efficiency of the attack. As already known, the mask image f_m can be amended using the swapping information stored in Q. Since all amended elements in f_m are also values in S_4, it is obvious that the efficiency of the search algorithm for finding correct random seeds will be increased. In addition, the swapping matrix Q can be used to uniquely determine some bits in $\{b(24k+16+i)\}_{i=0}^{7}$ without checking $f_m(16k+2i) \in S_4$ and $f_m(16k+2i+1) \in S_4$. Thus, the total complexity in finding a correct chaotic state will be less, and the attack will succeed faster.

When two or more plain-images and/or cipher-images are known, most swapped pixels can be successfully distinguished. In this case, it is much easier to find a pixel-block of f_m whose elements are all in S_4, which means that $Seed1(k)$, $Seed2(k)$ can be quickly guessed by enumerating all values in S_4, and all the 8 bits $\{b(24k+16+i)\}_{i=0}^{7}$ can be absolutely determined.

§3.3.4 The Combined Known-Plaintext Attack

The above two known-plaintext attacks have their disadvantages: the first attack cannot decrypt the cipher-images larger than MN (the size of f_m), and the second one cannot decrypt all pixels before the position where the first correct chaotic state $x(k)$ is found. One can combine them, however, to make a better known-plaintext attack without these disadvantages: use the first attack to decrypt the pixels before $x(k)$ and then use the second attack to decrypt the others. Figure 3.11 shows the performance of this combined attack with only one known plain-image, where the recovered chaotic state in the second attack is selected as $x(1673)$ (see also Fig. 3.10c, d), which can clearly show the boundary of the two parts decrypted by the two attacks.

a) The recovered plain-image f_{Peppers} b) The recovered larger plain-image $f_{\mathrm{Peppers2},768\times768}$

Figure 3.11: The recovery performance of the combined known-plaintext attack

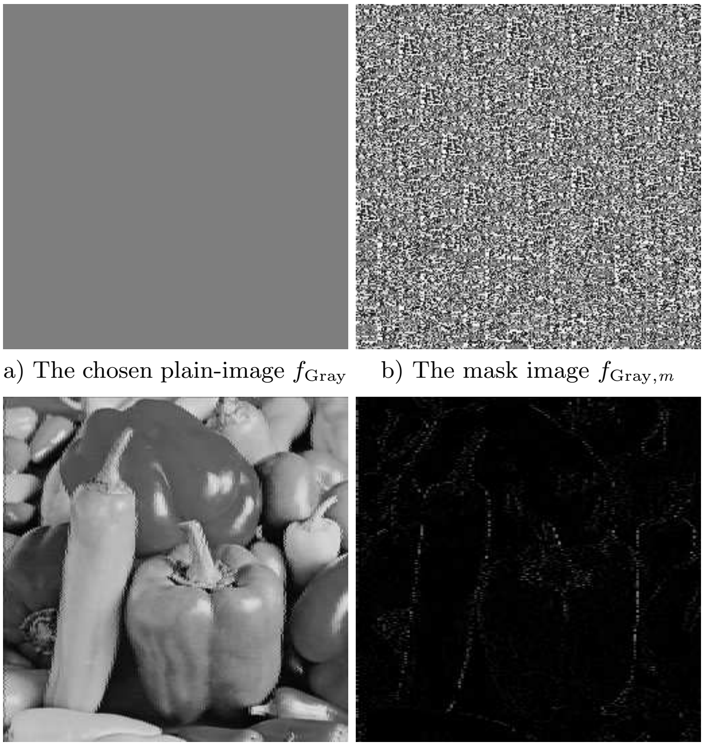
§3.3.5 The Chosen-Plaintext Attack

Apparently, all the above three known-plaintext attacks can be extended to chosen-plaintext attacks.

For the first kind of known-plaintext attack, the chosen-plaintext version can achieve much better recovery performance with a nearly-perfect mask image f_m, by choosing only one plain-image whose pixels are all fixed to be the same gray value. Given such a plain-image, from Corollary 3.1, any recovered plain-pixel will be the plain-pixel itself or its adjacent pixel. Thus, although the recovery error bounded by $a_1 = f_1(16k+2i) \oplus f_1(16k+2i+1)$ may still be large, it is expected that the visual quality of the recovered plain-image will be much better. It is also expected that all salt-pepper impulsive noises will disappear and a dithering effect of edges will occur, which is demonstrated in Fig. 3.12c with the plain-image f^*_{Peppers} recovered from the chosen plain-image shown in Fig. 3.12a. As a natural result, the visual quality of the recovered plain-image f^*_{Peppers} becomes much better as compared with the one shown in Fig. 3.4a.

Similarly to the known-plaintext attack, with some image processing techniques, the recovered plain-image in the chosen-plaintext attack can also be enhanced to further provide a better visual quality. Now, the question is: can one maximize the visual quality with an optimization algorithm? The answer is yes. In fact, with a subtly-designed algorithm, almost all dithering edges can be perfectly polished and a matrix Q containing partial swapping information can be constructed with only one chosen plain-image. In the following, this efficient algorithm and its real performance are studied in some details.

The proposed algorithm divides the image into $2n$-pixel blocks for enhancement, where $2n$ can exactly divide M. The basic idea is to exhaustively search



a) The chosen plain-image f_{Gray} b) The mask image $f_{\text{Gray},m}$

c) The recovered plain-image f^*_{Peppers} d) The difference image $\left|f^*_{\text{Peppers}} - f_{\text{Peppers}}\right|$

Figure 3.12: The recovery performance of the chosen-plaintext attack

the optimal swapping states of all pixels to achieve the minimal differential errors. For the m-th $2n$-pixel block $f_B(m) = \{f(m \cdot 2n + i)\}_{i=0}^{2n-1}$, the algorithm works as follows:

1. set $\{b_s(i) = 0\}_{i=0}^{n-1}$ and $\Delta_{min} = 256(n-1)$;
2. for $(b_0, \cdots, b_{n-1}) = (\overbrace{0, \cdots, 0}^{n}) \sim (\overbrace{1, \cdots, 1}^{n})$, do
   (a) assign $A = \{a_0, \cdots, a_{2n-1}\} = f_B(m)$;
   (b) for $i = 0 \sim n-1$, do $Swap_{b_i}(a_{2i}, a_{2i+1})$;
   (c) calculate $\Delta A = |a_2 - a_1| + |a_4 - a_3| + \cdots + |a_{2i} - a_{2i-1}| + \cdots + |a_{2n-2} - a_{2n-3}|$;
   (d) if $\Delta A < \Delta_{min}$, then set $\Delta_{min} = \Delta A$ and $\{b_s(i) = b_i\}_{i=0}^{n-1}$.
3. for $i = 0 \sim n-1$, do $Swap_{b_s(i)}(f(m \cdot 2n + 2i), f(m \cdot 2n + 2i + 1))$;

4. set the corresponding elements of the swapping matrix Q to be 1 for $b_s(i) = 1$.

The complexity of the above algorithm is $O(2^n \cdot MN)$. When $M = N = 256$ and $n = 8$, it is less than 2^{24}, which is practical even on PCs.

For the recovered plain-image f^*_{Peppers} shown in Fig. 3.12c, the above algorithm has been tested with parameter $n = 8$, and the result is given in Figs. 3.13a and 3.13b. Although the enhanced plain-image have 14378 (about 21.94% of all) pixels different from the original plain-image, its visual quality is so perfect that no any visual degradation can be distinguished. In fact, in a sense, the enhanced plain-image can be considered as a better version of the original one, since each $2n$-pixel block of the former reaches the minimum of the accumulated differential error. From such a point of view, this optimization algorithm can also be used to enhance the visual quality of the plain-image recovered by a known-plaintext attack. For the recovered plain-image shown in Fig. 3.4a, the enhancing result is given in Figs. 3.13c and 3.13d. It can be seen that dithering edges existing in the plain-image shown in Fig. 3.4a have been polished.

In the above algorithm, most swapped operations can be distinguished by using the minimum-detecting rule on the accumulated differential error of $f_B(m)$, which means that most elements in Q are correct for showing the real values of the swapping directive bits $\{b(24k + 16 + i\}_{i=0}^{7}$. Once 32 consecutive correct elements (two 16-pixel blocks) in Q have been found, it is possible to derive μ and a chaotic state $x(k)$, like in the situation of the second known-plaintext attack.

§3.4 Cryptanalysis of MES

§3.4.1 Three Properties of MES

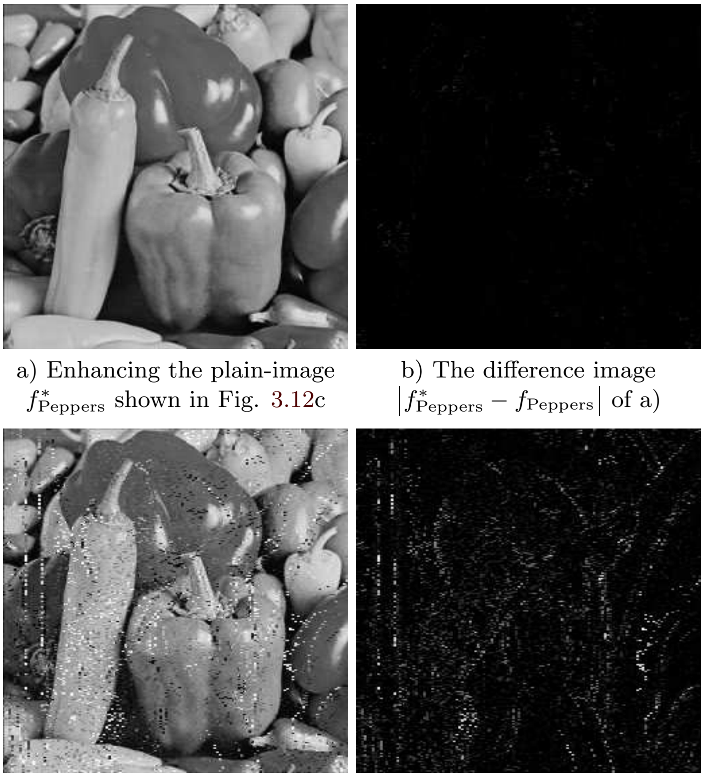
Define the XOR-differential ("differential" in short) of two signals f_0 and f_1 as $f_{0\oplus 1} = f_0 \oplus f_1$. Then, it is easy to prove the following three properties of MES, which will be the basis of the proposed attack.

Property 3.1: *The random masking in Step c) cannot change the differential value, i.e.,* $\forall\, k, j$, $f^{**(8)}_{0\oplus 1}(k, j) \equiv f^{*(8)}_{0\oplus 1}(k, j)$.

Proof: From Eq. (3.6), $f^{**(8)}_{0\oplus 1}(k, j) = f^{**(8)}_0(k, j) \oplus f^{**(8)}_1(k, j) = (f^{*(8)}_0(k, j) \oplus Seed(k, j)) \oplus (f^{*(8)}_1(k, j) \oplus Seed(k, j)) = f^{*(8)}_0(k, j) \oplus f^{*(8)}_1(k, j) = f^{*(8)}_{0\oplus 1}(k, j)$. ■

Property 3.2: *If the plaintext and the chaotic bit sequence are fixed, all differential bytes in* $f^{(8)}_{0\oplus 1}(k)$ *are fixed, i.e.,* $f^{(8)}_{0\oplus 1}(k)$ *are independent of the value of Open.*

Proof: For the first byte of each 8-byte block, if *temp* = *Open*, $f^{(8)}_{0\oplus 1}(k, 0) = 0$; otherwise, *temp* is one plain-byte occurring before $f^{(7)}(k)$, which means $f^{(8)}_{0\oplus 1}(k, 0)$



a) Enhancing the plain-image f^*_{Peppers} shown in Fig. 3.12c

b) The difference image $\left|f^*_{\text{Peppers}} - f_{\text{Peppers}}\right|$ of a)

c) Enhancing the plain-image f^*_{Peppers} shown in Fig. 3.4a

d) The difference image $\left|f^*_{\text{Peppers}} - f_{\text{Peppers}}\right|$ of c)

Figure 3.13: The performance of the optimization algorithm when $n = 8$

is one of the differential bytes occurring before $f^{(7)}_{0\oplus 1}(k)$. Apparently, the differential value $f^{(8)}_{0\oplus 1}(k, 0)$ is independent of the value of *Open*, but uniquely determined by the plaintext and the secret chaotic sequence. Since the other 7 bytes in $f^{(8)}_{0\oplus 1}(k)$ are also independent of *Open*, this property is thus proved. ■

Properties 3.1 and 3.2 mean that MES is reduced to be a three-stage cipher with *Open* $= 0$ (thus becomes a modification of TDCEA §2.3.2), from the differential point of view.

Property 3.3: *The byte permutation in Step b) cannot change each differential value, but its position in the 8-byte block.*

Proof: This property is obviously true since the byte permutation only change the position of each byte. ■

A natural result of the above property is: if $f^{(8)}_{0\oplus 1}(k, 0) = \cdots = f^{(8)}_{0\oplus 1}(k, 7)$, then it is true that $f^{(8)}_{0\oplus 1}(k) = f^{**(8)}_{0\oplus 1}(k)$. This means that MES is further reduced to be a two-stage cipher (and be a data-expansion modification of BRIE [32]), for differential blocks with 8 identical bytes.

§3.4.2 The Differential Attack

Utilizing Property 3.3 and the cryptanalysis on BRIE given in [81], one can easily break the secret bit recirculations in Step *d)*. Then, the secret byte permutations in Step *b)* and the secret data expansion in Step *a)* can be further broken by using Properties 3.1 and 3.2. Finally, the secret masking operations in Step *c)* are recovered immediately. After all secret operations in the four steps are revealed, most secret chaotic bits can be broken to derive the secret key with a sufficiently small complexity, which leads to the complete breaking of MES. Note that the proposed differential attack can be carried out by choosing either plaintexts or ciphertexts.

§3.4.2.1 Breaking the secret *ROLR* operations in Step d)

Choose two plaintexts to obtain the following differential signal $f_{0\oplus 1}$: $\forall\ i = 0 \sim N-1$, $f_{0\oplus 1}(i) \equiv a$. From the generation rule of $f^{(8)}(k,0)$, there exists a threshold integer, $k_0 \geq 1$, such that $f_{0\oplus 1}^{(8)}(k,0) \equiv 0$ when $k \leq k_0$ and $f_{0\oplus 1}^{(8)}(k,0) \equiv a$ when $k > k_0$. Assuming that $a \neq 0$ and each chaotic bit distribute uniformly over $\{0,1\}$, one can deduce that $Prob[k_0 = n] = Prob[l(0) = \cdots = l(n-1) = 0, l(n) \neq 0] = 7/8^{n+1}$. This means that $f_{0\oplus 1}^{(8)}(k,0) \equiv a$ is almost true when k is sufficiently large. In this case, $f_{0\oplus 1}^{(8)}(k,0) = \cdots = f_{0\oplus 1}^{(8)}(k,7)$ is true, so from Property 3.3 one can see that only Step *d)* is left for MES, i.e., $\forall\ j = 0 \sim 7$, $f_{0\oplus 1}^{\prime(8)}(k,j) = ROLR_{p(k,j)}^{q(k,j)}\left(f_{0\oplus 1}^{(8)}(k,j)\right) = ROLR_{p(k,j)}^{q(k,j)}(a)$. Now, MES is reduced to be BRIE, and the secret *ROLR* operations can be broken by setting $a = 1$ as discussed in [81]:

$$ROLR_{p(k,j)}^{q(k,j)} = ROLR_0^{8-\hat{q}(k,j)} = ROLR_1^{\hat{q}(k,j)}, \tag{3.19}$$

where $\hat{q}(k,j) = \log_2\left(f_{0\oplus 1}^{\prime(8)}(k,j)\right)$, which is the new position of the only 1-bit of $a = 1$ after the *ROLR* operation.

§3.4.2.2 Breaking the secret byte permutation in Step b)

Since the secret *ROLR* operations in Step *d)* has been recovered, from the differential point of view, MES becomes a permutation-only cipher with data expansion. As we analyzed in §2.2.3, all permutation-only ciphers are not secure enough against chosen-plaintext attacks. If two plaintexts are chosen to ensure that any two elements in each 8-byte differential block are different, one can uniquely determine the secret permutations by comparing $f_{0\oplus 1}^{(8)}(k,1) \sim f_{0\oplus 1}^{(8)}(k,7)$

and $f_{0\oplus 1}^{*(8)}(k,0) \sim f_{0\oplus 1}^{*(8)}(k,7)$. It is easy to do so in chosen-ciphertext attacks, by choosing 8 different cipher-bytes for each $f_{0\oplus 1}^{*(8)}(k)$. In chosen-plaintext attacks, since $f_{0\oplus 1}^{*(8)}(k,0)$ cannot be freely chosen, the condition is a little more complicated. Let us choose two plaintexts to get the following differential signal $f_{0\oplus 1}$: $\forall\, i = 0 \sim N$, $f_{0\oplus 1}(i) = (i{+}1) \bmod 256$. In this case, assuming that each chaotic bit distributes uniformly, one can calculate $P_c = Prob[f_{0\oplus 1}(k,0) \in \{f_{0\oplus 1}(k,j)\}_{j=1}^{7}]$, as:

- $P_c = 0$ when $0 \leq k \leq \lfloor 255/7 \rfloor - 1 = 35$;
- $P_c \leq 1/8^{35} = 1/2^{105}$ when $k \geq 36$.

It is obvious that P_c is negligible in all cases, and it is almost true that $\forall\, i,j \in \{0,\cdots,7\}$ and $i \neq j$, $f_{0\oplus 1}^{*(8)}(k,i) \neq f_{0\oplus 1}^{*(8)}(k,j)$. As a result, the secret permutation of the k-th block can be uniquely determined as a bijective index-mapping $F(k,i) = i'$, where $i,i' \in \{0,\cdots,7\}$. If some bytes in $f_{0\oplus 1}(k)$ happen to be identical, one can choose one more pair of plaintexts to try to recover the secret permutations.

§3.4.2.3 Breaking the secret data expansion in Step a)

Once the secret permutations of two consecutive blocks, $f_{0\oplus 1}^{(8)}(k)$ and $f_{0\oplus 1}^{(8)}(k+1)$, are broken, one can immediately get the value of $l(k)$ by finding the position of $f_{0\oplus 1}^{(8)}(k+1,0)$ in the 8 bytes of $f_{0\oplus 1}^{(8)}(k)$.

§3.4.2.4 Breaking the secret masking parameters in Step c)

After Steps *a)*, *b)* and *d)* are broken, the two intermediate blocks, $f_0^{*(8)}(k)$ and $f_0^{**(8)}(k)$ can be derived from f_0 and f_0', respectively. Then, the masking parameters can be calculated as follows: $\forall\, k,j$, $Seed(k,j) = f_0^{*(8)}(k,j) \oplus f_0^{**(8)}(k,j)$.

§3.4.2.5 Breaking the secret chaotic bits and the secret key

Though the recovered secret operations in the above procedure can be used as the equivalent of the secret key to decrypt the ciphertexts, one can still further derive the secret chaotic bits, and then try to derive the values of α, β, μ and $x(0)$. Since the knowledge of *Open* does not influence the decryption, it is excluded from the secret key.

In Step *a)*, the three involved chaotic bits, $b(33k+0) \sim b(33k+2)$ can be directly derived from the value of $l(k)$.

In Step *c)*, 25 chaotic bits are involved: $b(33k+0) \sim b(33k+7)$ and $b(33k+8) \sim b(33k+15)$ determine $Seed1(k)$ and $Seed2(k)$, respectively, and $b(33k+16) \sim b(33k+24)$ determine $B(k,0) \sim B(k,7)$. To derive the unknown bits, one has to search for the values of $Seed1(k)$ and $Seed2(k)$ in the set $\{Seed(k,0),\cdots,Seed(k,7)\}\subseteq\{Seed1(k),\overline{Seed1(k)},Seed2(k),\overline{Seed2(k)}\}$. Apparently, the maximal number of possible combinations of $Seed1(k)$ and $Seed2(k)$ is 8. The three known bits $b(33k+0)\sim b(33k+2)$ can be used to eliminate some invalid combinations. Also, note that $B(k,j)$ and $B(k,j+1)$ ($j=0\sim 6$) have a common bit, $b(33k+17+j)$, which can be used as a second constraint to eliminate invalid combinations of $Seed1(k)$ and $Seed2(k)$. In most cases, the values of $Seed1(k)$ and $Seed2(k)$ can be uniquely determined, and then all the 25 chaotic bits can be derived (see the experimental result in the next subsection).

In Step *d)*, 9 chaotic bits, $b(33k+24)\sim b(33k+32)$, are used to determine the values of $p(k,j)$ and $q(k,j)$, together with α and β. Observing the bit-recirculation procedure and Eq. (3.19), one can see that $\hat{q}(k,j)\in\boldsymbol{Q}=\{\alpha,\alpha+\beta,8-\alpha,8-(\alpha+\beta)\}$ holds. So, by exhaustively searching for all $1+\cdots+6=21$ possible combinations of α and β, one can determine the 9 chaotic bits with the following equations: $\forall\, j=0\sim 7$,

$$b(33k+25+j)=\begin{cases}0, & \hat{q}(k,j)\in\{\alpha,8-\alpha\},\\ 1, & \hat{q}(k,j)\in\{\alpha+\beta,8-(\alpha+\beta)\},\end{cases}\tag{3.20}$$

$$b(33k+24+j)=\begin{cases}0, & \hat{q}(k,j)\in\{\alpha,\alpha+\beta\},\\ 1, & \hat{q}(k,j)\in\{8-\alpha,8-(\alpha+\beta)\}.\end{cases}\tag{3.21}$$

Note Eq. (3.20) is invalid when $\alpha=8-(\alpha+\beta)$, i.e., $2\alpha+\beta=8$, and Eq. (3.21) is invalid when $\alpha=4$, $\alpha+\beta=4$ or $2\alpha+\beta=8$. According to how the two equations can be used to determine the 9 chaotic bits from $\{\hat{q}(k,j)\}_{j=0}^{7}$, all possible values of (α,β) can be divided into the following three classes.

- *C1)* $\alpha\neq 4$, $\alpha+\beta\neq 4$ *and* $2\alpha+\beta\neq 8$: both Eqs. (3.20) and (3.21) are valid, so all the 9 chaotic bits, $b(33k+24)\sim b(33k+32)$, can be uniquely determined. There are 12 *C1*-values, all of which satisfy $\#(\boldsymbol{Q})=4$.

- *C2)* $4\in\{\alpha,\alpha+\beta\}$ *(which ensures* $2\alpha+\beta\neq 8$*)*: Eq. (3.20) is valid and the 8 chaotic bits, $b(33k+25)\sim b(33k+32)$, can be uniquely determined. When $\alpha=4$ and $b(33k+25)=1$, or $\alpha\neq 4$ and $\widetilde{b}(33k+25)=0$, one can also determine $b(33k+24)$ by Eq. (3.21). There are 6 *C2*-values, all of which satisfy $\#(\boldsymbol{Q})=3$.

- *C3)* $2\alpha+\beta=8$: Eqs. (3.20) and (3.21) are not valid, so all the 9 chaotic bits have to be exhaustively guessed. There are 3 *C3*-values, which satisfy $\#(\boldsymbol{Q})=2$.

* For C2/C3-classes, note that $b(33k+24)$ can be recovered in Step *c)* in a high probability.

Since the above three classes correspond to different values of $\#(\boldsymbol{Q})$, one need not search for all 21 values of (α, β), but those corresponding to $\#(\boldsymbol{Q})$, which can reduce the search complexity to some extent. The value of $\#(\boldsymbol{Q})$ can be estimated from the cardinality of $\boldsymbol{Q}' = \{\hat{q}(k,j)\}_{k=0,j=0}^{N/7-1,7}$, or one of its subset. It is obvious that $\#(\boldsymbol{Q}') = \#(\boldsymbol{Q})$ almost true when N is sufficiently large.

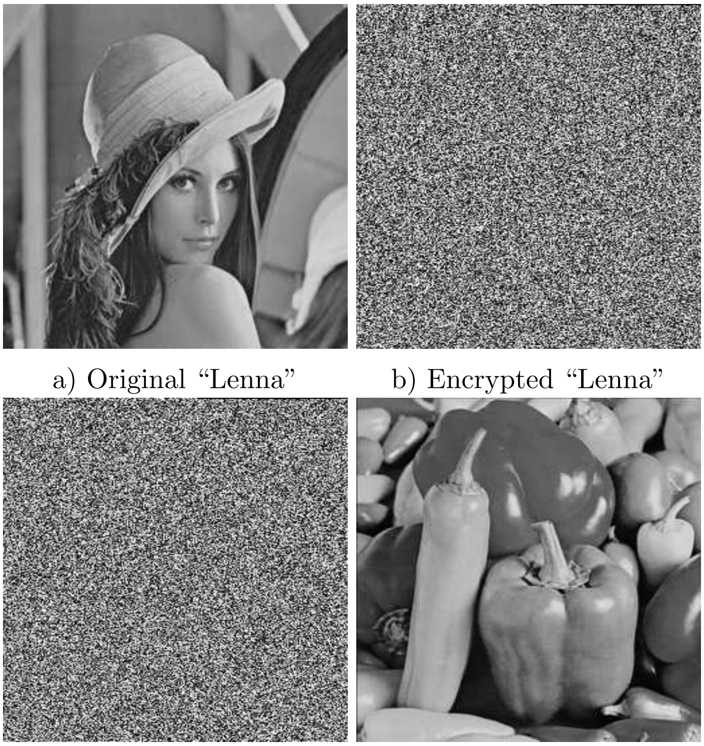
To verify which guessed value of (α, β) is the real one, the following procedure is useful by estimating the values of two consecutive chaotic states, $x(k)$ and $x(k+1)$, and the value of μ. Assuming all the 33 chaotic bits, $b(33k+0) \sim b(33k+32)$ have been successfully recovered (or guessed) with the above procedure, one can immediately get the value of $x(k) = 0.b(33k+0) \sim b(33k+32)$. After getting $x(k+1)$ in a similar way, one can calculate an estimation of μ like the procedures in §3.3.3.2.

By iterating the Logistic map from $x(k+1)$ until $x(N/7-1)$ and then checking the coincidence between these chaotic states and the corresponding bits that can be uniquely derived, one can detect wrong values of (α, β) and μ and distinguish the real ones. To minimize the complexity, one can check only a number of chaotic states, sufficiently far from $x(k+1)$, to eliminate most wrong values, and verify the few left ones by checking all chaotic states from $x(k+2)$ to $x(N/7-1)$.

§3.4.3 Experiments and the Attack Complexity

As discussed above, to carry out the differential chosen-plaintext attack, only three plaintexts are enough to construct two plaintext differentials, as follows: 1) $\forall\, i = 0 \sim N-1$, $f_{0\oplus 1}(i) \equiv 1$; 2) $\forall\, i = 0 \sim N-1$, $f_{0\oplus 1}(i) = (i+1) \bmod 256$. When one plaintext is chosen as an image "Lenna", the performance of the proposed attack has been tested, and the results are shown in Fig. 3.14. The two differentials are used to break the secret operations and then try to break some chaotic bits. It is found that for 8084 blocks in total 9363 ones, all the 33 involved chaotic bits can be uniquely determined. Two chaotic states, $x(1)$ and $x(2)$, are used to estimate μ, and then to find the secret key for recovering the ciphertext of another image "Peppers" (see Figs. 3.14c and d).

Finally, we briefly discuss the attack complexity. It can be easily verified that the complexity of breaking all secret operations is proportional to N. The complexity of breaking the secret key depends on the value of (α, β). When (α, β) belongs to C1 and C2 classes, the attack complexity is also proportional to N; when (α, β) belongs to C3 class, the attack complexity is $2^8 \cdot 2^8 = 2^{16}$ times of the complexity of that in C1/C2-cases, which is still practically small.



a) Original “Lenna” b) Encrypted “Lenna”

c) Encrypted “Peppers” d) Recovered “Peppers”

Figure 3.14: The differential chosen-plaintext attack to MES

§3.4.4 The Brute-Force Attacks

Another obvious problem of MES is that the key space is not cryptographically large. The secret key for decrypting MES includes $(\mu, x(0))$, which is represented by $2 \cdot 33 = 66$ secret bits, and (α, β), which has 21 possible values. Thus, one can see that the key space of MES is only $21 \cdot 2^{66}$, which is not sufficiently large from the cryptographical point of view [1]. What's worse, since the Logistic map is not chaotic for all values of μ far less than 4, the key space is even smaller than $21 \cdot 2^{66}$. To make MES practically secure in today's digital world, the key space should be not less than $O\left(2^{128}\right)$. One simple method to enlarge the key space is to realize the chaotic Logistic map with a higher finite precision, i.e., to increase the number of secret bits for representing μ and $x(0)$.

§3.5 Cryptanalysis of DSEA

§3.5.1 The Brute-Force Attack

The secret key of DSEA is $(L, initial\_key, \mu, x(0))$, which has $M \cdot 2^{3\cdot 8} = M \cdot 2^{24}$ possible values. Taking the complexity of verifying each key into consideration, the total complexity of searching for all possible keys is $O\left(2^{24} \cdot M^2\right)$. When the plaintext is selected as a typical image of size 256×256, the complexity will be $O(2^{56})$, which is much smaller than $O(2^M \cdot M) = O(2^{65552})$, the complexity claimed in [42]. Note that the real complexity is even smaller since not all values of μ can ensure the chaoticity of the Logistic map [82]. That is, the security of DSEA against brute-force attacks was over-estimated much in [42]. In today's digitized and networked world, the complexity of order $O(2^{128})$ is required for a cryptographically-strong cipher [1], which means DSEA is not practically secure.

§3.5.2 The Ciphertext-Only Attacks

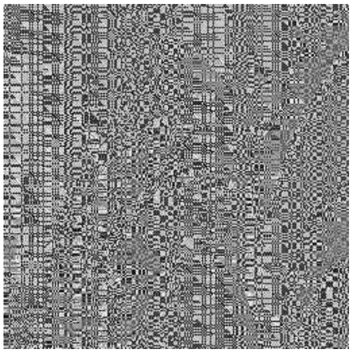
Since the transmission channel is generally insecure, the security against ciphertext-only attacks are required for any ciphers. However, it is found that DSEA is not sufficiently secure against ciphertext-only attacks, since much information about the plaintext and the secret key can be found from even one ciphertext.

Given an observed ciphertext g', generate two mask texts, g_0^* and g_1^*, as follows: $g_0^*(0) = 0$, $g_1^*(0) = 0, \forall\ n = 1 \sim M-1$, $g_0^*(n) = g'(n) \oplus \overline{g'(n-1)}$, $g_1^*(n) = g'(n) \oplus g'(n-1)$. From the encryption procedure of DSEA, it can be easily verified that the following result is true when $n \bmod L \neq 0$:

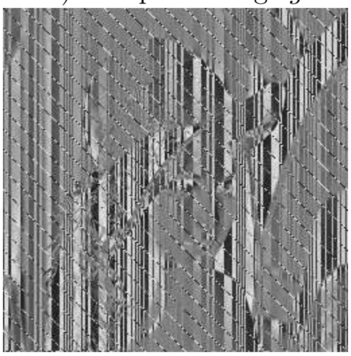
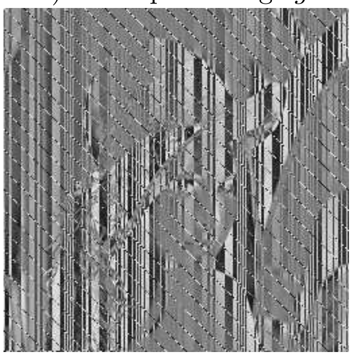
$$g(n) = \begin{cases} g_0^*(n), & b(n) = 0, \\ g_1^*(n), & b(n) = 1, \end{cases} \tag{3.22}$$

which means that $g(n)$ is equal to either $g_0^*(n)$ or $g_1^*(n)$. Assuming that each chaotic bit distributes uniformly over $\{0, 1\}$, one can deduce that the percentage of right plain-pixels in g_0^* and g_1^* is not less than $\frac{L-1}{L} \cdot \frac{1}{2} = \frac{1}{2} - \frac{1}{2L}$. When L is large, about half pixels in g_0^* and g_1^* are plain-pixels in g, and it is expected that some visual information of the plain-image can be distinguished from g_0^* and g_1^*.

To verify the above idea, one 256×256 image, "Lenna", has been encrypted to get g_0^* and g_1^*, with the following secret parameters: $L = 15$, $initial\_key = 170$, $\mu = 251/2^6 \approx 3.9219$, $x(0) = 69/2^8 \approx 0.2695$. The experimental results are shown in Fig. 3.15. In g_0^* there are 27726 pixels that are identical with those in g, and in g_1^* there are 33461 such pixels. Observing Figs. 3.15 c and d, one can see that the plain-image roughly emerges from both g_0^* and g_1^*.

a) The plain-image g b) The cipher-image g'

c) The mask image g_0^* d) The mask image g_1^*

Figure 3.15: A ciphertext-only attack to DSEA.

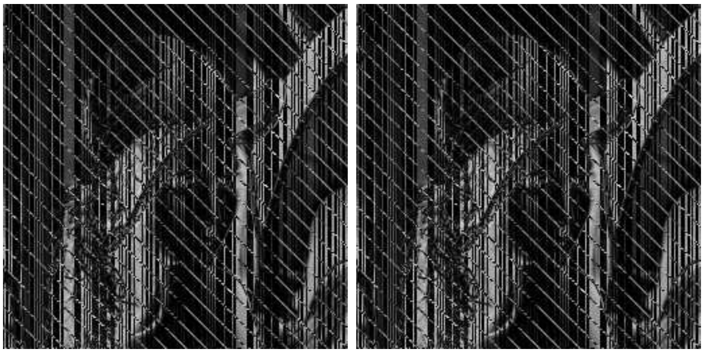
In addition, from either g_0^* or g_1^*, it is possible to directly get the value of L, if there exists strong correlation between adjacent bytes of the plaintext (speeches and natural images are good examples). This is due to the probability difference existing between the following two kinds of plain-bytes:

- when $n \bmod L \neq 0$, $g_0^*(n) = g(n)$ and $g_1^*(n) = g(n)$ with a probability of $\frac{1}{2}$;
- when $n \bmod L = 0$, $g_0^*(n) = g(n)$ and $g_1^*(n) = g(n)$ with a probability* of $\frac{1}{256}$: $g_0^*(n) = g(n)$ if and only if $g'(n-1) = \overline{initial\_key}$; $g_1^*(n) = g(n)$ if and only if $g'(n-1) = initial\_key$.

When there exists strong correlation between adjacent bytes, the above probability difference implies that there exists strong discontinuity around each position satisfying $n \bmod L = 0$ (with a high probability). The fixed occurrence period of such discontinuous bytes will generate periodically-occurring straight lines in the mask text when it is an image or displayed in 2-D mode, as shown in Figs. 3.15c

*Without loss of generality, it is assumed that each cipher-byte distributes uniformly in $\{0, \cdots, 255\}$.

and d. Then, it is easy to determine the occurrence period, i.e., the value of L, by checking the horizontal distance between any two adjacent lines. To make the straight line clearer, one can calculate the differential images of g_0^* and g_1^*, as shown in Fig. 3.16, where the differential image of an image $g = \{g(n)\}_{n=0}^{M-1}$ is defined as follows: $g_d(0) = g(0)$ and $\forall\, n = 1 \sim M-1$, $g_d(n) = |g(n) - g(n-1)|$. Note that the two differential images of g_0^* and g_1^* are identical according to the following theorem, from which one can get that $|g_0^*(n) - g_0^*(n-1)| = |g'(n) \oplus \overline{g'(n-1)} - g'(n-1) \oplus \overline{g'(n-2)}| = |g'(n) \oplus g'(n-1) - g'(n-1) \oplus g'(n-2)| = |g_1^*(n) - g_1^*(n-1)|$.



a) $g_{d,0}^*$ b) $g_{d,1}^*$

Figure 3.16: The differential images of g_0^* and g_1^*.

Theorem 3.2: *For any three s-bit integers, a, b, c, it is true that* $|(a \oplus b) - (b \oplus c)| = |(a \oplus \bar{b}) - (b \oplus \bar{c})|$.

Proof: Introduce four new variables, $A = a \oplus b$, $B = b \oplus c$, $A' = a \oplus \bar{b}$, $B' = b \oplus \bar{c}$. It can be easily verified that $A' = \overline{A}$ and $B' = \overline{B}$, since $a \oplus \bar{b} = a \oplus b \oplus b \oplus \bar{b} = a \oplus b \oplus (2^s - 1) = \overline{a \oplus b}$, where s is the binary precision of a, b. That is, $(a \oplus b) - (b \oplus c) = A - B$ and $(a \oplus \bar{b}) - (b \oplus \bar{c}) = \overline{A} - \overline{B}$. Let $A = (A_0 \cdots A_{s-1})_2 = \sum_{i=0}^{s-1} A_i \cdot 2^i$, $B = (B_0 \cdots B_{s-1})_2 = \sum_{i=0}^{s-1} B_i \cdot 2^i$. Since $\forall\, A_i, B_i \in \{0, 1\}$, $A_i - B_i = \bar{B}_i - \bar{A}_i$, it is obvious that $A - B = \sum_{i=0}^{s-1}(A_i - B_i) \cdot 2^i = \sum_{i=0}^{s-1}(\bar{B}_i - \bar{A}_i) \cdot 2^i = \overline{B} - \overline{A}$. As a result, $|(a \oplus b) - (b \oplus c)| = |A - B| = |\overline{B} - \overline{A}| = |\overline{A} - \overline{B}| = |(a \oplus \bar{b}) - (b \oplus \bar{c})|$, which completes the proof. ■

§3.5.3 The Known/Chosen-Plaintext Attacks

Although it was claimed that DSEA can resist this kind of attacks [42, Sec. IV.B], we found this claim is not true: with a limited number of continuous plain-bytes of only one known/chosen plaintext, one can completely break the secret key to decrypt other unknown plain-bytes of the known/chosen plaintext and any new ciphertexts encrypted with the same key. Apparently, even when the secret key is changed for each plaintext (as mentioned in [42, Sec. IV.B]), DSEA is insecure against known/chosen-plaintext attacks. In the following, let us discuss how to break the four sub-keys, respectively.

1) Breaking the sub-key L: as mentioned above, once one gets a ciphertext, he can easily deduce the value of L by observing the periodically-occurring straight lines in the two constructed mask texts, g_0^* and g_1^*. Furthermore, since the plaintext is also known, it is possible to generate an enhanced differential image, g_d^*, as follows: $g_d^*(0) = 0$, and $\forall\, n = 1 \sim M - 1$,

$$g_d^*(n) = \begin{cases} 0, & g(n) \in \{g_0^*(n), g_1^*(n)\}, \\ 255, & g(n) \notin \{g_0^*(n), g_1^*(n)\}. \end{cases} \tag{3.23}$$

See Fig. 3.17 for the enhanced differential image corresponding the cipher-image shown in Fig. 3.15b. Compared with Fig. 3.16, one can see that the straight lines become clearer.

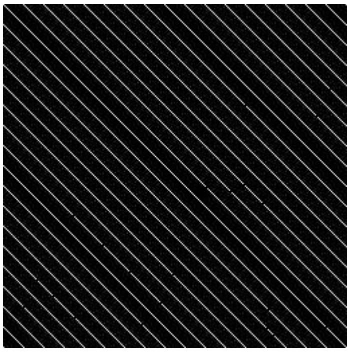


Figure 3.17: The enhanced differential image g_d^*.

2) Breaking the initial_key: for all values of n that satisfy $n \bmod L = 0$, it is obvious that

$$initial\_key = \begin{cases} g(n) \oplus g'(n), & b(n) = 1, \\ \overline{g(n) \oplus g'(n)}, & b(n) = 0. \end{cases} \tag{3.24}$$

Note that it is possible to uniquely determine the value of *initial_key*, when there may exist pixels satisfying $n \bmod L = 0$ and $g_d^*(n) = 0$, i.e., $g(n) \in \{g_0^*(n), g_1^*(n)\} = \left\{g'(n) \oplus \overline{g'(n-1)}, g'(n) \oplus g'(n-1)\right\}$. Considering $g'(n) = g(n) \oplus initial\_key$, one can immediately deduce that

$$initial\_key = \begin{cases} g'(n-1), & g(n) = g_1^*(n), \\ \overline{g'(n-1)}, & g(n) = g_0^*(n). \end{cases} \tag{3.25}$$

3) Breaking the chaotic PRBS and the other two sub-keys: once L and $initial\_key$ have been determined, the chaotic PRBS, $\{b(n)\}_{n=0}^{M-1}$, $x(0) = \sum_{i=0}^{7} b_i \cdot 2^{-(i+1)}$ can be immediately derived as follows:

- when $n \bmod L \neq 0$: if $g(n) = g_0^*(n)$ then $b(n) = 0$, else $b(n) = 1$;
- when $n \bmod L = 0$: if $initial\_key = g(n) \oplus g'(n)$ then $b(n) = 1$, else $b(n) = 0$.

With 16 consecutive chaotic bits, $b(8k + 0) \sim b(8k + 15)$, one can further derive two consecutive chaotic states: $x(k) = 0.b(8k + 0) \cdots b(8k + 7)$ and $x(k + 1) = 0.b(8k + 8) \cdots b(8k + 15)$, and then derive an estimation of the sub-key μ like the procedures in §3.3.3.2.

With the above steps, the whole secret key $(L, initial\_key, \mu, x(0))$ can be recovered, and then be used for decryption. For the plain-image "Lenna", a breaking result is shown in Fig. 3.18. It can be verified that the complexity of the known/chosen-plaintext attacks is only $O(M)$, which means a perfect breaking of DSEA.

Figure 3.18: The recovered plain-image of "Lenna" in a known-plaintext attack.

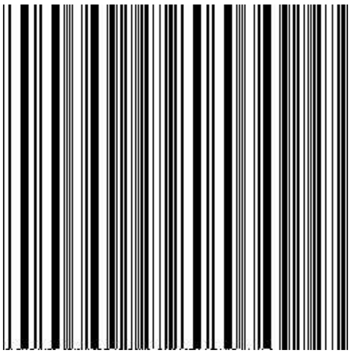
§3.5.4 Improving DSEA

In this section, we study some possible remedies to DSEA to resist the proposed attacks. It is concluded that DSEA cannot be simply enhanced to resist known/chosen-plaintext attacks.

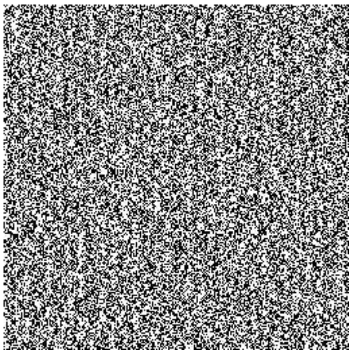
To ensure the complexity of the brute-force attack cryptographically large, the simplest idea is to increase the presentation precision of $x(0)$ and μ. Binary presentations of $x(0)$ and μ with 64-bit (long integers) are suggested to provide a complexity not less than $O(2^{128})$ against the brute-force attack.

Apparently, the insecurity of DSEA against ciphertext-only and known/chosen-plaintext attacks is mainly due to the invertibility of XOR operations. This is actually the weakness of all XOR-based stream ciphers. To make DSEA securer, one has to change the encryption structure and/or the basic masking operations, in other words, one has to design a completely new cipher, instead of enhancing DSEA to design a modified cipher.

In addition, there exists a special flaw in DSEA. According to [**?** , Sec. 2.5], when a chaotic system is implemented in s-bit finite computing precision, each chaotic orbit will lead to a cycle whose length is smaller than 2^s (and generally much smaller than 2^s). Figure 3.19a shows the pseudo-image of the chaotic PRBS recovered in a known-plaintext attack. It is found that the cycle of the chaotic PRBS is only $2^6 = 64$ and the period of the corresponding chaotic orbit is only $2^3 = 8$. Such a small period of the chaotic PRBS will make all attacks easier. To amend this defect, using a higher implementation precision or floating-point arithmetic is suggested. Figure 3.19b gives the pseudo-image of the chaotic PRBS when the chaotic states are calculated under double-precision floating-point arithmetic. It is obvious that the short-period effect of the chaotic PRBS is effectively avoided.



a) 8-bit fixed-point arithmetic



b) double-precision floating-point arithmetic

Figure 3.19: The pseudo-image of the chaotic PRBS, under two different finite-precision arithmetics.

§3.6 Yet Another Cryptanalysis of TDCEA

The known/chosen-plaintext attacks given in §2.5.4, §2.5.5 have two disadvantages: 1) the number of required known plain-images is somewhat large; 2) with n known plain-images of size $M \times N$, this attack can only decrypt cipher-images of size not greater than $M \times N$. In this section, we will introduce another known-plaintext attack and chosen-plaintext attack, by which we can get the secret keys with only one known plain-image (but with a larger complexity).

§3.6.1 Known-Plaintext Attack: Getting the Secret Key from One Known Plain-Image

The known-plaintext attack introduced in this subsection is actually an optimized brute-force attack. By utilizing the correlation information existing between two consecutive chaotic states and the control parameter μ, the multiplicative search of the two sub-keys $x(0)$ and μ can be reduced to be the additive search of two chaotic states $x(k)$ and $x(k+1)$. This can dramatically reduce the attack complexity. Also, since each guessed chaotic state can be verified by a few number of 8-pixel blocks, not by the whole known plain-image, the attack complexity can be further reduced.

The basic idea of this attack is based on the following facts: 1) each permutation matrix $\boldsymbol{W}_k$ is uniquely determined by the current chaotic state $x(k)$ and the two sub-keys α, β; 2) two consecutive chaotic states $x(k)$ and $x(k+1)$ satisfy $x(k+1) \approx \mu \cdot x(k) \cdot (1 - x(k))$. Once an attacker gets the right values of any two consecutive chaotic states, he can immediately get an estimation of μ, and then completely break TDCEA if α and β are also known.

To get the right value of a chaotic state $x(k)$ corresponding to the k-th bit matrix $\boldsymbol{M}_k$, one can use the permutation information existing in $\boldsymbol{M}_k$ and $\boldsymbol{M}'_k$. When there are t 0-bits and $(64 - t)$ 1-bits in $\boldsymbol{M}_k$, one can calculate that the number of all possible values of $\boldsymbol{M}'_k$ is $C(t) = \binom{64}{t} = \frac{64!}{t!(64-t)!}$. In comparison, the number of all possibilities of each permutation matrix is equal to the number of all possible values of the 3-tuple data $(x(k), \alpha, \beta)$, which is less than $N_s = 2^{17} \cdot 25$. When $5 \leq t \leq 59$, one has $C(t) \gg N_s$ (see Fig. 3.20). This means that the probability that a wrong value of $(x(k), \alpha, \beta)$ coincides with $\boldsymbol{W}'_k$ is close to zero, i.e., one can exhaustively search all possible values of $(x(k), \alpha, \beta)$ to find a few number of candidates of the right value. Apparently, such an exhaustive searching procedure is optimized when $t = 32$.

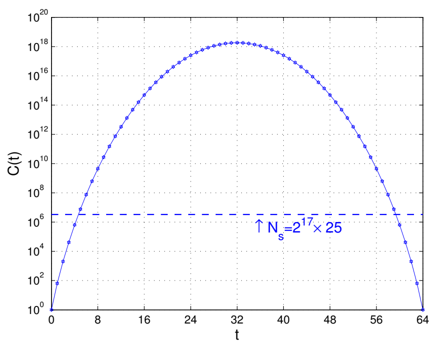


Figure 3.20: $C(t) = \binom{64}{t} = \frac{64!}{t!(64-t)!}$ with respect to t

Carrying out the above procedure on two consecutive bit matrices, one can find some candidates of two consecutive chaotic states, $x(k) = 0.b(17k+0)\cdots b(17k+16)$ and $x(k+1) = 0.b(17k+17)\cdots b(17k+33)$. Then, an estimated value of the sub-key μ can be derived as §3.3.3.2.

The proposed known-plaintext attack can be concretized step by step as follows.

- *Step 1*: Find the first two consecutive plain-blocks, $f^{(8)}(k)$ and $f^{(8)}(k+1)$, whose corresponding bit matrices $\boldsymbol{M}_k$ and $\boldsymbol{M}_{k+1}$ both have about 32 0-bits.

  Note: assuming that each bit in $\boldsymbol{M}_k$ distributes uniformly and independently, one can deduce that

  $$P_s = Prob\left[|t-32| \leq s\right] = \frac{\sum_{i=32-s}^{32+s}\binom{64}{i}}{2^{64}}, \tag{3.26}$$

  where t is the number of nonzero elements of $\boldsymbol{M}_k$ and $0 \leq s \leq 32$. When $s = 4$, $P_s \approx 0.7396$, which is sufficiently large for an attacker to find valid plain-blocks within all the $MN/8$ blocks.

- *Step 2*: Exhaustively search all possible values of $(x(k), \alpha, \beta)$, and record those coinciding with $\boldsymbol{M}_k$ and $\boldsymbol{M}'_k$. Assume that m_1 candidates are recorded in total: $\{x_i(k), \alpha_i^*, \beta_i^*\}_{i=0}^{m_1-1}$.

- *Step 3*: Search all possible values of $x(k+1)$ and all values of (α, β) in $\{\alpha_i^*, \beta_i^*\}_{i=0}^{m_1-1}$, and record those coinciding with $\boldsymbol{M}_{k+1}$ and $\boldsymbol{M}'_{k+1}$. Assume that m_2 candidates are recorded in total: $\{x_j(k+1), \alpha_j^{**}, \beta_j^{**}\}_{j=0}^{m_2-1}$.

- *Step 4*: For $i = 0 \sim m_1 - 1$ and $j = 0 \sim m_2 - 1$, do the following operations.

  - *Step 4a*: If $\alpha_i^* = \alpha_j^{**}$ and $\beta_i^* = \beta_j^{**}$, then calculate $\widetilde{\mu} = \dfrac{x_j(k+1)}{x_i(k)\cdot(1-x_i(k))}$ and continue to execute *Step 4b*; otherwise, go to the next loop.
  - *Step 4b*: Assuming that $x_j(k+1) \geq 2^{-n}$, exhaustively search all possible 2^{n+3} values of μ within the neighborhood of $\widetilde{\mu}$. For each searched value, iterate the Logistic map from $x_i(k+1)$ to $x_i(MN/8-1)$. If every chaotic state $x_i(l)$ and (α_i^*, β_i^*) agree with $\boldsymbol{M}_l$ and $\boldsymbol{M}'_l$ $(l = k+2 \sim MN/8-1)$, then stops (the attack completes).

The time complexity of this attack can be calculated as follows.

- The average complexity of *Step 2* is $2^{17}\cdot 25\cdot(14\cdot 8+\frac{1}{2}\cdot 8\cdot 8) < 2^{29}$.

- The complexity of *Step 3* is obviously less than that of *Step 2*.

- The average number of exhaustive searching loops in *Step 4* is $(m_1 \cdot m_2 \cdot C_x)$, where
$$C_x = \sum_{n=1}^{17} 2^{n+3} \cdot Prob\left[2^{-n} \leq x_j(k+1) < 2^{-(n-1)}\right],$$
which is the mathematical expectation of the space size of the searching neighborhood of $\widetilde{\mu}$. Considering the computational complexity for each searching loop, the average complexity of *Step 4* is of order $\frac{m_1 \cdot m_2 \cdot C_x}{2} \cdot 49MN$. Without loss of generality, assume that $x_j(k+1)$ distributes uniformly over the interval [0,1], i.e., $Prob\left[2^{-n} \leq x_j(k+1) < 2^{-(n-1)}\right] = 2^{-n}$. Thus, $C_x = \sum_{n=1}^{17} 2^{n+3} \cdot 2^{-n} = 2^3 \cdot 17 = 136$. Then, the average complexity becomes $O(\frac{833 m_1 m_2 MN}{2})$. Since, in almost cases, $MN \leq 4096 \cdot 4096 = 2^{24}$ and m_1, m_2 are generally very small, the complexity is generally not greater than $O(2^{36})$.

Combining the above results, one concludes that the total complexity is $O(2^{36})$, which is practically small even for a PC and much smaller than $O(2^{60})$, the complexity of the simple brute-force attack shown in §2.5.3.

Figure 3.21 shows an experimental result of the recovered plain-image "Peppers", where the 5-th and 6-th pixel-blocks are chosen to exhaustively search the secret key. As a result, all chaotic states from $x(5)$ are successfully derived and only $(5 \cdot 8 = 40)$ leading plain-pixels at the left-bottom corner are not recovered correctly.

Figure 3.21: The recovered plain-image "Peppers" by the second known-plaintext attack.

§3.6.2 Chosen-Plaintext Attack: Getting the Secret Key

In the first chosen-plaintext attack presented in §2.5.5, one can get 16 values, $s_k(0) \sim s_k(7)$ and $r_k(0) \sim r_k(7)$, for each pixel block $f^{(8)}(k)$. Based on the 16 values, the second known-plaintext attack discussed in §2.5.5 can be dramatically enhanced in most cases by introducing a much more effective way of deriving the 17 secret bits, $b(17k+0) \sim b(17k+16)$, of the chaotic state $x(k)$.

To simplify the following discussions, create a new vector, $rs_k(i)$ $(i = 0 \sim 15)$, which satisfies that $\forall\, i = 0 \sim 7$, $rs_k(i) = r_k(i)$ and $\forall\, i = 8 \sim 15$, $rs_k(i) = s_k(i-8)$.

Recalling the encryption procedure of TDCEA, it is obvious that the 16 values $\{rs_k(i)\}_{i=0}^{15}$ have a deterministic relation with the 17 secret bits $b(17k+0) \sim b(17k+16)$. Similar to §3.4.2.5, such a relation can be used to facilitate an exhaustive search of the 17 secret bits, i.e., the search of the k-th chaotic state $x(k) = 0.b(17k+0)\cdots b(17k+16)$.

Considering the fact that $\text{Rotate}X_i^{0,r} = \text{Rotate}X_i^{1,8-r}$, $\text{Rotate}Y_j^{0,s} = \text{Rotate}Y_j^{1,8-s}$, one can see that $\forall\, i = 0 \sim 15$, $k = 0 \sim MN/8-1$, $rs_k(i)$ must be a value in the set $\mathbb{S} = \{\alpha, \alpha+\beta, 8-\alpha, 8-(\alpha+\beta)\}$.

For each guessed value $(\widetilde{\alpha}, \widetilde{\beta})$, one can determine 16 bits, denoted by $\widetilde{b}(17k+1) \sim \widetilde{b}(17k+16)$, as estimations of $b(17k+1) \sim b(17k+16)$, as follows: $\forall\, i = 1 \sim 16$,

$$\widetilde{b}(17k+i) = \begin{cases} 0, & rs_k(i-1) \in \{\widetilde{\alpha}, 8-\widetilde{\alpha}\}, \\ 1, & rs_k(i-1) \in \{\widetilde{\alpha}+\widetilde{\beta}, 8-(\widetilde{\alpha}+\widetilde{\beta})\}. \end{cases} \tag{3.27}$$

Note that the above equation is invalid when $\widetilde{\alpha} = \widetilde{\alpha}+\widetilde{\beta}$ or $\widetilde{\alpha} = 8-(\widetilde{\alpha}+\widetilde{\beta})$, i.e., $\widetilde{\beta} = 0$ or $2\widetilde{\alpha}+\widetilde{\beta} = 8$. Similarly, one has another equation for estimating the values of $b(17k+0) \sim b(17k+15)$: $\forall\, i = 0 \sim 15$,

$$\widetilde{b}(17k+i) = \begin{cases} 0, & rs_k(i) \in \{\widetilde{\alpha}, \widetilde{\alpha}+\widetilde{\beta}\}, \\ 1, & rs_k(i) \in \{8-\widetilde{\alpha}, 8-(\widetilde{\alpha}+\widetilde{\beta})\}. \end{cases} \tag{3.28}$$

The above equation is invalid when $\widetilde{\alpha} = 4$, $\widetilde{\alpha}+\widetilde{\beta} = 4$ or $2\widetilde{\alpha}+\widetilde{\beta} = 8$.

According to how much information that one can get from $\{rs_k(i)\}_{i=0}^{15}$, all values of $(\widetilde{\alpha}, \widetilde{\beta})$ can be divided into the following classes in the chosen-plaintext attack.

- *C1)* $\widetilde{\alpha} \neq 4$, $\widetilde{\alpha}+\widetilde{\beta} \neq 4$, $\widetilde{\beta} \neq 0$ *and* $2\widetilde{\alpha}+\widetilde{\beta} \neq 8$: $\widetilde{b}(17k+1) \sim \widetilde{b}(17k+16)$ and $\widetilde{b}(17k+0) \sim \widetilde{b}(17k+15)$ can be uniquely determined by Eq. (3.27) and Eq. (3.28), respectively, so all the 17 bits, $\widetilde{b}(17k+0) \sim \widetilde{b}(17k+16)$, can be uniquely recovered.
  - There are 12 *C1*-values of $(\widetilde{\alpha}, \widetilde{\beta})$, as follows: $(1,1)$, $(1,2)$, $(1,4)$, $(1,5)$, $(2,1)$, $(2,3)$, $(2,5)$, $(3,3)$, $(3,4)$, $(5,1)$, $(5,2)$, $(6,1)$.
- *C2)* $4 \in \{\widetilde{\alpha}, \widetilde{\alpha}+\widetilde{\beta}\}$ *and* $\widetilde{\beta} \neq 0$ *(which ensures* $2\widetilde{\alpha}+\widetilde{\beta} \neq 8$*)*: $\widetilde{b}(17k+1) \sim \widetilde{b}(17k+16)$ can be uniquely determined by Eq. (3.27), but $\widetilde{b}(17k+0)$ has to

be guessed*.

- There are 6 *C2*-values of $(\widetilde{\alpha},\widetilde{\beta})$, as follows: $(1,3)$, $(2,2)$, $(3,1)$, $(4,1)$, $(4,2)$, $(4,3)$.

- *C3)* $\widetilde{\alpha}\neq 4$ *and* $\widetilde{\beta}=0$ *(which ensures* $2\widetilde{\alpha}+\widetilde{\beta}\neq 8$*)*: $\widetilde{b}(17k+0)\sim\widetilde{b}(17k+15)$ can be uniquely determined by Eq. (3.28), but $\widetilde{b}(17k+16)$ has to be guessed.

  - There are 6 *C3*-values of $(\widetilde{\alpha},\widetilde{\beta})$, as follows: $(1,0)$, $(2,0)$, $(3,0)$, $(5,0)$, $(6,0)$, $(7,0)$.

- *C4)* $2\widetilde{\alpha}+\widetilde{\beta}=8$: all the 17 bits has to be exhaustively guessed, as in the second known-plaintext attack discussed in Sec. 4.2.

  - There are 4 *C4*-values of $(\widetilde{\alpha},\widetilde{\beta})$, as follows: $(1,6)$, $(2,4)$, $(3,2)$, $(4,0)$.

The above four different cases correspond to different values of $\#(\mathbb{S})$ as follows:

- $\#(\mathbb{S})=4$: $(\widetilde{\alpha},\widetilde{\beta})$ is one of the 12 *C1*-values;
- $\#(\mathbb{S})=3$: $(\widetilde{\alpha},\widetilde{\beta})$ is one of the 6 *C2*-values;
- $\#(\mathbb{S})=2$: $(\widetilde{\alpha},\widetilde{\beta})$ is one of the 6 *C3*-values and the following *C4*-values: $\{(1,6),(2,4),(3,2)\}$;
- $\#(\mathbb{S})=1$: $(\widetilde{\alpha},\widetilde{\beta})=(4,0)$ (a *C4*-value).

Since one can guess the value of $\#(\mathbb{S})$ by observing the cardinality of the set $\{rs_k(0),\cdots,rs_k(15)\}\subseteq\mathbb{S}$, it is possible to search (α,β) in part of all possible values to reduce the attack complexity. Apparently, the success probability of such a guess is $P_e=Prob[\mathbb{S}=\{rs_k(0),\cdots,rs_k(15)\}]$. Since the theoretical deduction of P_e is rather difficult, experiments are performed to test all 2^{17} possible values of $b(17k+0)\sim b(17k+16)$. It results in that $P_e=122684/2^{17}\approx 0.936$, which is sufficiently large. Note that it is easy to further increase the success probability of the guess, by observing $n>1$ blocks at the same time. In doing so, the success probability will be *greater* than $P_e^{(n)}=1-(1-P_e)^n$ under the assumption that the chaotic bits for different blocks distribute uniformly and independently. As n increases, $P_e^{(n)}$ will approach 1 exponentially. In real attacks, even $n=2$ is enough in almost all cases, since $P_e^{(2)}\approx 0.996$. If all guessed values determined

*Note that $\widetilde{b}(17k+0)$ can be **uniquely** determined in the following two sub-cases: a) when $\widetilde{\alpha}=4$ and $\widetilde{b}(17k+1)=1$, one can uniquely determine $\widetilde{b}(17k+0)$ by Eq. (3.28) since $\widetilde{\alpha}+\widetilde{\beta}\neq 4$; b) when $\widetilde{\alpha}\neq 4$ and $\widetilde{b}(17k+1)=0$, one can also uniquely determine $\widetilde{b}(17k+0)$ by Eq. (3.28). The two sub-cases occur with a probability of 0.5 when $\{b(i)\}$ distributes uniformly over $\{0,1\}$.

by $\#(\{rs_k(0), \cdots, rs_k(15)\})$ fail to pass the verification, it means that the rare event $\{rs_k(0), \cdots, rs_k(15)\} \subset \mathbb{S}$ occurs*. In this case, one has to continue to exhaustively search all other values of (α, β).

When the real value of (α, β) belongs to *C1*, *C2*, *C3* classes, the complexity of the chosen-plaintext attack will be much smaller than the complexity of its known-plaintext counterpart, due to the following reasons:

- the exhaustive searching procedure for the 17 bits of each chaotic state is simplified to be a deterministic calculation procedure dominated by Eqs. (3.27) and/or (3.28);
- the number of guessed values of (α, β) is reduced from 28 to 12 for *C1*, 6 for *C2* and *C3*;
- some values of (α, β) can be verified by checking whether or not $\{rs_k(0), \cdots, rs_k(15)\} \subseteq \{\widetilde{\alpha}, \widetilde{\alpha} + \widetilde{\beta}, 8 - \widetilde{\alpha}, 8 - (\widetilde{\alpha} + \widetilde{\beta})\}$;
- one can intentionally choose the second chaotic state to ensure $x(k+1) \geq 0.5$, i.e., $b(17(k+1)+0) = 1$, so as to reduce C_x, the average searching complexity of μ, from 136 to $2^{1+3} = 16$.
- the exhaustive search of μ can be validated by just comparing the calculated chaotic state with the bits derived by Eqs. (3.27) and/or (3.28).

When the real value of (α, β) belongs to *C4* class, the average complexity of the chosen-plaintext attack is also smaller than the one of its known-plaintext counterpart, since the value of (α, β) can be immediately determined* with a sufficiently high probability, $P_e^{(n)} \approx 1$, that is, only when the rare event $\{rs_k(0), \cdots, rs_k(15)\} \subset \mathbb{S}$ occurs, one needs to exhaustively search the value of (α, β).

§3.7 Conclusion

In this chapter, the security properties of three chaotic multimedia encryption algorithms, RCES [37, 41], MES [44] and DSEA[42], have been analyzed in detail. In addition, scheme TDCEA[38, 43] has been investigated from different point of view with §2.5. The following security problems are found: 1) RCES can be broken with one and/or two known-plaintexts and one chosen-plaintext respectively; 2)

*Note that the occurrence probability is not zero, though it is very close to zero when n is sufficiently large.

*If $\#(\mathbb{S}) = 1$, then $(\alpha, \beta) \equiv (4, 0)$; otherwise, one can determine the value of (α, β) quickly by checking the following three candidates: $(1, 6), (2, 4), (3, 2)$.

given two chosen-plaintexts, one can break MES; 3) DSEA can be partially and totally broken with one ciphertext and one known/chosen-plaintext respectively; 4) The security of the three schemes against brute-force was all over-estimated much by the authors; 5) TDCEA can be broken with one known-plaintext and two chosen-plaintexts respectively. Both theoretical and experimental analyses have been given to support the feasibility of the above proposed attacks.

The insecurity of these algorithms are caused by a careless design. Although the four schemes cannot be used in practice as secure ciphers to protect multimedia data, they serve as typical examples for caution.

Chapter 4

Cryptanalyses of Two Neural-Network-Based Encryption Schemes

§4.1 Introduction

From 1998, research group of Yen proposed a number of multimedia encryption schemes [84, Sec. 4.4.3]. Among them, a class of chaotic-neural-network based encryption schemes were presented in [31, 34, 36]. In a recent paper [26], this class of schemes were simply extended to arbitrary block size without influencing the security and then applied for JPEG2000 image encryption. In this chapter, we name this class of schemes as Yen's scheme. This chapter evaluates the security of the neural-network-based scheme and points out two security problems: 1) it can be easily broken by the known/chosen-plaintext attacks with only one known/chosen plaintext; 2) its security against the brute-force attack was much over-estimated.

In ISNN'04 ([24]), Zhou etal. proposed another clipped-neural-network (CNN) based chaotic cipher, which is named as Zhou's scheme in this chapter. This chaotic cipher employs a chaotic pseudo-random signal and the output of an 8-cell clipped neural network to mask the plaintext, along with modulus additions and XOR operations. Also, the evolution of the neural network is controlled by the chaotic signal. With such a complicated combination, it was hoped that the chaotic cipher can resist chosen-plaintext attacks. Unfortunately, our analysis shows that it is still not secure against chosen-plaintext attacks. By choosing only two plaintexts, an attacker can derive an equivalent key to break the cipher.

The rest of this chapter is organized as follows. In the next section, a brief introduction of two neural-network-based encryption algorithms is given. Cryptanalysis of Yen's scheme is given in §4.3, where some experimental results are given to support the proposed attacks, and some remedies of the scheme are also discussed. The chosen-plaintext attack of Zhou's scheme is given in §4.4, with some theoretical and experimental results. The last section is the conclusion.

§4.2 Two Neural-Network-Based Encryption Schemes

§4.2.1 The Yen's Scheme

Assuming that $\{f(n)\}_{n=0}^{M-1}$ is a 1-D signal for encryption, the encryption procedure of Yen's scheme can be briefly depicted as follows:

- *The secret key* is the control parameter μ and the initial point $x(0)$ of the chaotic Logistic map Eq. (2.8), which are all L-bit binary decimals.

- *The initialization procedure*: under L-bit finite computing precision, run the Logistic map from $x(0)$ to get a chaotic sequence $\{x(i)\}_{i=0}^{\lceil 8M/K\rceil-1}$, and extract K bits below the decimal dot of each chaotic state* to generate a chaotic bit sequences $\{b(i)\}_{i=0}^{8M-1}$, where $x(i) = 0.b(Ki+0)\cdots b(Ki+K-1)\cdots$.

- *The encryption procedure*: For the n-th plain-element $f(n) = \sum_{i=0}^{7} d_i(n)\cdot 2^i$, the corresponding cipher-element $f'(n) = \sum_{i=0}^{7} d'_i(n)\cdot 2^i$ is determined by the following process:

  – for $i = 0 \sim 7$ and $j = 0 \sim 7$, 64 weights w_{ji} are calculated as follows: if $i = j$, $w_{ji} = 0$; else $w_{ji} = 1 - 2b(8n+i) = \begin{cases} 1, & b(8n+i) = 0, \\ -1, & b(8n+i) = 1; \end{cases}$

  – for $i = 0 \sim 7$, 8 biases θ_i are calculated as follows:

$$\theta_i = \frac{2b(8n+i)-1}{2} = \begin{cases} -1/2, & b(8n+i) = 0, \\ 1/2, & b(8n+i) = 1; \end{cases}$$

  – the i-th cipher-bit $d'_i(n)$ is calculated as follows:

$$d'_i(n) = \operatorname{sign}\left(\sum_{j=0}^{7} w_{ji}\cdot d_i(n) + \theta_i\right), \tag{4.1}$$

  where sign(·) denotes the sign function, i.e., $\operatorname{sign}(x) = \begin{cases} 1, & x \geq 0, \\ 0, & x < 0. \end{cases}$

- *The decryption procedure* is the same as the above one.

*In real implementations of Yen's scheme, the K bits can be extracted from the direct multiplication result $\mu x(i-1)(1-x(i-1))$, before $x(i)$ is obtained by quantizing the value. As a result, it is possible that $K > L$. For example, in [36], $K = 32 > L = 17$.

The above encryption procedure looks very complicated, however, actually it can be simplified to be a much more precise form. Observing the proofs of [31, 34, Proposition 1] and [36, Lemma 1], one can see the following fact:

$$d_i'(n) = \begin{cases} 0, & \text{if } d_i(n) = 0 \text{ and } b(8n+i) = 0, \\ 1, & \text{if } d_i(n) = 1 \text{ and } b(8n+i) = 0, \\ 1, & \text{if } d_i(n) = 0 \text{ and } b(8n+i) = 1, \\ 0, & \text{if } d_i(n) = 1 \text{ and } b(8n+i) = 1, \end{cases} \tag{4.2}$$

which means that

$$d_i'(n) = d_i(n) \oplus b(8n+i). \tag{4.3}$$

Obviously, Yen's scheme is a stream cipher encrypting the plain-signal bit by bit, where the key stream for masking is the chaotic bit sequence $\{b(i)\}$.

§4.2.2 The Zhou's Scheme

At first, we give an introduction to the CNN employed in the chaotic ciphers. The neural network contains 8 neural cells, denoted by $S_0, \cdots, S_7 \in \{1, -1\}$, and each cell is connected with other cells with eight synaptic weights $w_{ij} \in \{1, 0, -1\}$, among which only three are non-zeros. The syntax weights between two connected cells are identical: $\forall\, i, j = 0 \sim 7$, $w_{ij} = w_{ji}$. The neural network evolves with the following mechanism: $\forall\, i = 0 \sim 7$,

$$f(S_i) = \text{sign}\left(\widetilde{S}_i\right) = \begin{cases} 1, & \widetilde{S}_i > 0, \\ -1, & \widetilde{S}_i < 0, \end{cases} \tag{4.4}$$

where $\widetilde{S}_i = \sum_{j=0}^{7} w_{ij} S_j$. Note that $\widetilde{S}_i \neq 0$ holds at all time.

Now let us see how the chaotic cipher works with the above CNN. Without loss of generality, assume that $f = \{f(i)\}_{i=0}^{N-1}$ is the plaintext signal, where $f(i)$ denotes the i-th plain-byte and N is the plaintext size in byte. Accordingly, denote the ciphertext by $f' = \{f'(i)\}_{i=0}^{N-1}$, where $f'(i)$ is a double-precision floating-point number corresponding to the plain-byte $f(i)$. Then, the encryption procedure can be briefly depicted as follows*.

- *The secret key* includes the initial states of the 8 neural cells in the CNN, $S_0(0), \cdots, S_7(0)$, the initial condition $x(0)$ and the control parameter r of the following chaotic tent map:

$$T(x) = \begin{cases} rx, & 0 < x \leq 0.5, \\ r(1-x), & 0.5 < x < 1, \end{cases} \tag{4.5}$$

*Note that some original notations used in [24] have been changed in order to achieve a better description.

where r should be very close to 2 to ensure the chaoticity of the tent map.

- *The initial procedure*: 1) in double-precision floating-point arithmetic, run the tent map from $x(0)$ for 128 times before the encryption starts; 2) run the CNN for $128/8 = 16$ times (under the control of the tent map, as discussed below in the last step of the encryption procedure); 3) set $x(0)$ and $S_0(0), \cdots, S_7(0)$ to be the new states of the tent map and the CNN.

- *The encryption procedure*: for the i-th plain-byte $f(i)$, do the following steps to get the ciphertext $f'(i)$:

  - evolve the CNN for one step to get its new state: $S_0(i), \cdots, S_7(i)$;
  - in double-precision floating-point arithmetic, run the chaotic tent map for 8 times to get 8 chaotic states: $x(8i+0), \cdots, x(8i+7)$;
  - generate 8 bits by extracting the 4-th bits of the 8 chaotic states: $b(8i+0), \cdots, b(8i+7)$, and then $\forall\, j = 0 \sim 7$, set $E_j = 2 \cdot b(8i+j) - 1$;
  - encrypt $f(i)$ as follows*:

$$f'(i) = \left(\left(\frac{f(i) \oplus B(i)}{256} + x(8i+7) \right) \bmod 1 \right), \tag{4.6}$$

    where $B(i) = (b(8i+0), \cdots, b(8i+7))_2 = \sum_{j=0}^{7} b(8i+j) \cdot 2^{7-j}$;
  - $\forall\, i = 0 \sim 7$, if $S_i \neq E_i$, update all the three non-zero weights of the i-th neural cell and the three mirror weights as follows: $w_{ij} = -w_{ij}$, $w_{ji} = -w_{ji}$.

- *The decryption procedure* is similar to the above one with the following decryption formula:

$$f(i) = (256 \cdot ((f'(i) - x(8i+7)) \bmod 1)) \oplus B(i). \tag{4.7}$$

§4.3 Cryptanalysis of Yen's Scheme

§4.3.1 The Brute-Force Attacks

In [31, 34, 36], it was claimed that the computing complexity of a brute-force attack to Yen's scheme is $O\left(2^{8M}\right)$, since there are $8M$ bits in $\{b(i)\}_{i=0}^{8M-1}$ (which is unknown to the attacker). However, this statement is not true due to the following fact: the $8M$ bits are uniquely determined by the secret key, i.e., the

*In [24], $x(8i+7)$ was mistaken as $x(8)$.

control parameter μ and the initial condition $x(0)$, which have only $2L$ secret bits. This means that there are only 2^{2L} different chaotic bit sequences.

Now, let us see what is the real complexity of a brute-force attack. For each guessed value of $x(0)$ and μ, about $8M/K$ chaotic iterations and $8M$ XOR operations are needed for verification. Assuming that each L-bit digital multiplication needs L times of additions, then each chaotic iteration needs $2L+1$ times of additions. Therefore, the complexity of a brute-force attack to Yen's scheme will be $O\left(2^{2L}\cdot\left(\frac{8M(2L+1)}{K}+8M\right)\right)=O\left(2^{2L}M\right)$, which is much smaller than 2^{8M} when M is not too small. What's more, considering the fact that the Logistic map can exhibit strong chaotic behavior only when μ is close to 4 [82], the complexity should be even smaller than $O\left(2^{2L}M\right)$.

The above analysis shows that the security of Yen's scheme was much overestimated by the authors, even under the simplest attack. Because of the rapid progress of digital computer and distributed computing techniques, the complexity not lower than $O\left(2^{128}\right)$ is required for a cryptographically strong cipher [1]. To achieve such a security level, $L\geq 64$ is required. As a comparison, $L=8$ in [34] and $L=17$ in [36], which are both too small*.

§4.3.2 The Known/Chosen-Plaintext Attacks

In known-plaintext or chosen-plaintext attacking scenarios, Yen's scheme can be broken with only one known/chosen plaintext $\{f(n)\}_{n=0}^{M-1}$ and its corresponding ciphertext $\{f'(n)\}_{n=0}^{M-1}$, with a complexity that is smaller than the complexity of a brute-force attack.

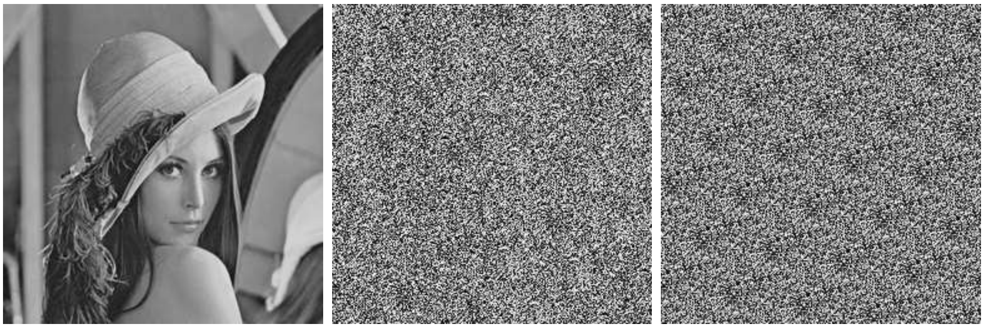
From Eq. (5.3), one can get $b(8n+i)=g_i(n)\oplus g_i'(n)$. That is, an attacker can successfully reconstruct the chaotic bit sequence $\{b(i)\}_{i=0}^{8M-1}$ by simply XORing $\{f(n)\}_{n=0}^{M-1}$ and $\{f'(n)\}_{n=0}^{M-1}$ bit by bit. Assuming $\{f_m(n)=f(n)\oplus f'(n)\}_{n=0}^{M-1}$, one has $f_m(n)=0.b(8n+0)\cdots b(8n+7)$. Without deriving the secret key $(\mu,x(0))$, given any ciphertext g' encrypted with the same secret key, the attacker can use f_m to decrypt the M leading bytes of the corresponding plaintext g: $n=0\sim M-1$, $g(n)=g'(n)\oplus f_m(n)$. Here, we call f_m the *mask signal* (or the *mask image* when Yen's scheme is used to encrypt digital images), since the plaintext can be decrypted by using f_m to "mask" (i.e., XOR) the ciphertext†.

To demonstrate the above attack, with the parameters $L=17, K=32$ [36] and the secret key $\mu=3.946869$, $x(0)=0.256966$, some experiments are given for the encryption of digital images. In Fig. 4.1, a 256×256 known/chosen plain-image "Lenna", its corresponding cipher-image, and the mask image $f_m=f\oplus f'$

*In [31], the value of L is not explicitly mentioned. Since [31] is an initial version of [34], it is reasonable to assume $L=8$.

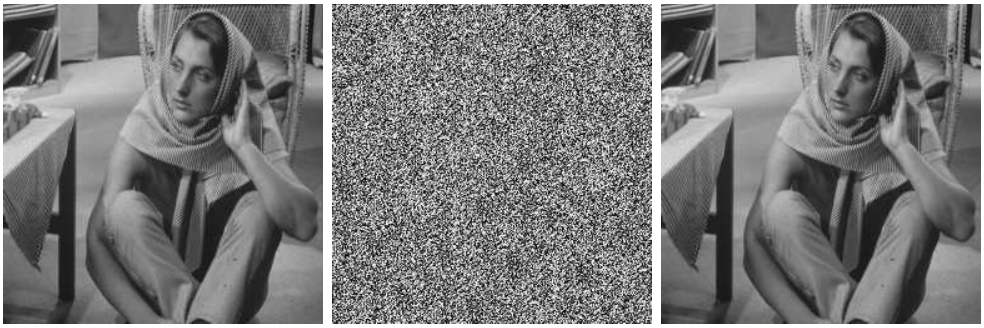
†In fact, it is a common defect of most stream ciphers [1].

are shown. If another plain-image "Babarra" (of size 256×256) is encrypted with the same key, it can be broken with the mask image f_m derived from "Lenna" as shown in Fig. 4.2. For a larger plain-image "Peppers" (of size 384×384), the 256×256 leading pixels can be successfully broken with f_m as shown in Fig. 4.3.



a) The plain-image f b) The cipher-image f' c) The mask image f_m

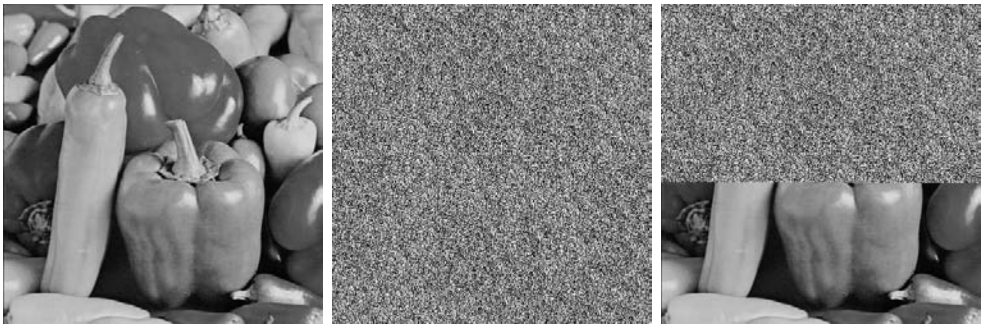
Figure 4.1: One known/chosen plain-image "Lenna" (256×256), its corresponding cipher-image, and the mask image $f_m = f \oplus f'$



a) The plain-image "Babarra" b) The encrypted "Babarra" c) The recovered "Babarra" with f_m

Figure 4.2: Decrypt a plain-image "Babarra" (256×256) with f_m shown in Fig. 4.1c

From the above experiments, one can see that the breaking performance of known/chosen-plaintext attacks based on f_m is limited. Fortunately, from the reconstructed bit sequence $\{b(i)\}_{i=0}^{8M-1}$, it is easy for an attacker to derive the values of μ and $x(0)$, and then to completely break Yen's scheme. Even when only part of a plaintext $f(n_1) \sim f(n_2)$ is known to the attacker, he can still derive the values of μ and a chaotic state $x(i)$, which can be used to calculate all following chaotic states, i.e., all following chaotic bits $\{b(i)\}_{i=8n_2}^{\infty}$. In this case, all plain-



a) The plain-image "Peppers" b) The encrypted "Peppers" c) The recovered "Peppers" with f_m

Figure 4.3: Decrypt a plain-image "Peppers" (384×384) with f_m shown in Fig. 4.1c

pixels after the n_1-th position can be broken. In the following, let us discuss how to derive chaotic states and the value of μ.

Firstly, let us see how a chaotic state $x(i)$ is derived. Recall the generation procedure of $\{b(i)\}_{i=0}^{8M-1}$. It is easy to reconstruct a K-bit approximate of the chaotic sequence by dividing $\{b(i)\}_{i=0}^{8M-1}$ into K-bit segments: $\{\widetilde{x}(i)\}_{i=0}^{\lceil 8M/K \rceil -1}$, where $\widetilde{x}(i) = 0.b(Ki+0)\cdots b(Ki+K-1)$ and

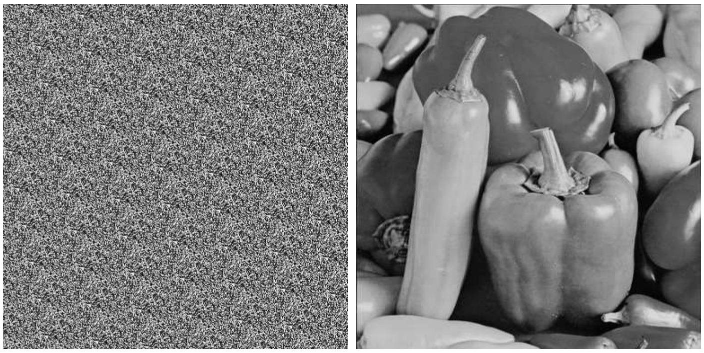
$$|\Delta x(i)| = |\widetilde{x}(i) - x(i)| \leq 0.\overbrace{0\cdots0}^{K}\overbrace{1\cdots1}^{L-K} = \sum_{j=K+1}^{L} 2^{-j} < 2^{-K}. \tag{4.8}$$

Apparently, when $L \leq K$, $\widetilde{x}(i) = x(i)$; when $L > K$, the exact value of each chaotic state $x(i)$ can be derived by exhaustively guessing the $L-K$ unknown bits, and the guess complexity is $O\left(2^{L-K}\right)$.

Once two consecutive chaotic states $x(i)$ and $x(i+1)$ are derived, the estimated value of μ can be calculated like §3.3.3.2 with a sufficiently small search complexity. With the mask image f_m derived from the known plain-image "Lenna" (of size 256×256) shown in Fig. 4.1a, the values of $x(0)$ and μ are calculated following the above procedure to completely decrypt the larger plain-image "Peppers" (of size 384×384). The decryption result is given in Fig. 4.4.

Finally, it deserves being mentioned that even without deriving the secret key there is another way based on a mask signal f_m to decrypt any plaintext of arbitrary size. It is due to the following fact: for a digital chaotic system implemented in L-bit finite computing precision, each chaotic orbit will lead to a cycle whose length is smaller than 2^L (and generally much smaller than 2^L, see [83, Sec. 2.5]). For the implementation of Yen's scheme in [36], $L = 17$, $K = 32$. Thus, the cycle length of each chaotic orbit will be much smaller than 2^{17} in most

Figure 4.4: The decrypted "Peppers" (384×384) with the secret key derived from f_m shown in Fig. 4.1c



a) The extended mask image f_m^* b) The recovered "Peppers" with f_m^*

Figure 4.5: Decrypt "Peppers" (384×384) with f_m^* extended from f_m shown in Fig. 4.1c

cases. Such a length is not sufficiently large in comparison with the size of many plaintexts, especially for digital images and videos. For example, a 256×256 image corresponds to a chaotic orbit $\{x(i)\}$ whose length is $8 \cdot 256 \cdot 256/32 = 2^{14}$. For almost every value of μ and $x(0)$, the cycle length of $\{x(i)\}$ is even much smaller than 2^{14}, which means that there exists an visible repeated pattern in $\{x(i)\}$. Carefully observing the mask image f_m shown in Fig. 4.1c, one can easily find such a repeated pattern. Then, it is easy to get the cycle of f_m, and to extend it to arbitrary sizes by appending more cycles at the end of the original mask signal. This means that any ciphertext can be decrypted with a mask signal f_m^* extended from the mask image f_m. Using such a method, the larger plain-image "Peppers" is completely decrypted as shown in Fig. 4.5.

§4.3.3 Improving Yen's Scheme

The simplest way to improve the original Yen's scheme is to make L sufficiently large so as to ensure the complexity of the brute-force attack cryptographically large. In addition, to make the complexity of guessing the $L-K$ unknown bits of each chaotic state cryptographically large, $L-K$ should also be sufficiently large. To be practical, $(L,K)=(64,8)$ is suggested. In this case, the complexity to get the value of $x(0)$ is $O\left(2^{L-K}\right)=O\left(2^{56}\right)$, and the complexity to get the value of μ (i.e., to get two consecutive chaotic states) is $O\left(2^{2(L-K)}\right)=O\left(2^{112}\right)$. Such a complexity is sufficiently large to make both the brute-force attack and the attack of deriving the secret key from f_m impossible in practice.

However, because Yen's scheme is a stream cipher, making $L-K$ sufficiently large cannot enhance the security against the known/chosen-plaintext attacks based on the mask signal f_m. To resist such attacks, a substitution encryption part should be used to make Yen's scheme a product cipher. Note that the security of the modified Yen's scheme is ensured by the new substitution part, not the Yen's scheme itself. So, essentially speaking, the Yen's scheme cannot be enhanced to resist known/chosen-plaintext attacks.

§4.4 Cryptanalysis of Zhou's Scheme

§4.4.1 The Chosen-Plaintext Attack

Although it was claimed that the chaotic cipher under study can resist this kind of attacks [24, Sec. 4], our cryptanalysis shows that such a claim is not true. By choosing two plaintexts, f_1 and f_2, satisfying $\forall\, i=0\sim N-1$, $f_1(i)=\overline{f_2(i)}$, one can derive two masking sequences as equivalent keys for decryption.

Before introducing the chosen-plaintext attack, three lemmas are given, which are useful in the following discussions.

Lemma 4.1: *$\forall\, a,b,c\in\mathbb{R}, c\neq 0$ and $n\in\mathbb{Z}^+$, if $a=(b \mod c)$, one has $a\cdot n=((b\cdot n) \bmod (c\cdot n))$.*

Proof: From $a=(b \mod c)$, one knows that $\exists\, k\in\mathbb{Z}$, $b=c\cdot k+a$ and $0\leq a<c$. Thus, $\forall\, n\in\mathbb{Z}^+$, $b\cdot n=c\cdot n\cdot k+a\cdot n$ and $0\leq a\cdot n<c\cdot n$, which immediately leads to $a\cdot n=((b\cdot n) \bmod (c\cdot n))$ and completes the proof of this lemma. ■

Lemma 4.2: *$\forall\, a,b,c,n\in\mathbb{R}$ and $0\leq a,b<n$, if $c=((a-b) \mod n)$, one has $a-b\in\{c,c-n\}$.*

Proof: This lemma can be proved under two conditions. i) When $a\geq b$, it is obvious that $((a-b) \bmod n)=a-b=c$. ii) When $a<b$, $((a-b) \bmod n)=$

$((n + a - b) \bmod n)$. Since $-n < a - b < 0$, one has $0 < n + a - b < n$, which means that $((a - b) \bmod n) = n + a - b = c$. That is, $a - b = c - n$. Combining the two conditions, this lemma is thus proved. ■

Lemma 4.3: *Assume that a, b are both 8-bit integers. If $a = b \oplus 128$, then $a \equiv (b + 128) \pmod{256}$.*

Proof: This lemma can be proved under two conditions. i) When $0 \leq a < 128$: $b = a \oplus 128 = a + 128$, so $a \equiv (b + 128) \pmod{256}$. ii) When $128 \leq a \leq 255$: $b = a \oplus 128 = a - 128$, so $a \equiv (b - 128) \equiv (b + 128) \pmod{256}$. ■

From Lemma 4.1, one can rewrite the encryption formula Eq. (4.6) as follows:

$$256 \cdot f'(i) = (((f(i) \oplus B(i)) + 256 \cdot x(8i + 7)) \bmod 256) \quad . \tag{4.9}$$

Given two plain-bytes $f_1(i) \neq f_2(i)$ and the corresponding cipher-blocks $f_1'(i), f_2'(i)$, one has $256 \cdot (f_1'(i) - f_2'(i)) \equiv ((f_1(i) \oplus B(i)) - (f_2(i) \oplus B(i))) \pmod{256}$. Without loss of generality, assume that $f_1'(i) > f_2'(i)$ and that $\Delta_{f_{1,2}} = 256 \cdot (f_1'(i) - f_2'(i))$. It is true that $0 < \Delta_{f_{1,2}} < 256$. Thus, one has

$$\Delta_{f_{1,2}} = (((f_1(i) \oplus B(i)) - (f_2(i) \oplus B(i))) \bmod 256) \quad . \tag{4.10}$$

Because $f_1(i) \oplus B(i)$ and $f_2(i) \oplus B(i)$ are 8-bit integers and $\Delta_{f_{1,2}} \neq 0$, from Lemma 4.2, one of the following facts is true:

1. $(f_1(i) \oplus B(i)) - (f_2(i) \oplus B(i)) = \Delta_{f_{1,2}} \in \{1, \cdots, 255\}$; (4.11a)
2. $(f_2(i) \oplus B(i)) - (f_1(i) \oplus B(i)) = \left(256 - \Delta_{f_{1,2}}\right) \in \{1, \cdots, 255\}$ (4.11b)

For the above two equations, when $f_1(i) = \overline{f_2(i)}$ is satisfied, two possible values of $B(i)$ can be uniquely derived according to the following theorem.

Theorem 4.1: *Assume that a, b, c, x are all 8-bit integers, and $c > 0$. If $a = \bar{b}$, then the equation $(a \oplus x) - (b \oplus x) = c$ has an unique solution $x = a \oplus (1, c_7, \cdots, c_1)_2$, where $c = (c_7, \cdots, c_0)_2 = \sum_{i=0}^{7} c_i \cdot 2^i$.*

Proof: Since $a = \bar{b}$, one has $b \oplus x = \overline{a \oplus x}$. Thus, by substituting $y = a \oplus x$ and $\bar{y} = \overline{a \oplus x} = b \oplus x$ into $(a \oplus x) - (b \oplus x) = c$, one can get $y - \bar{y} = c$, which is equivalent to $y = \bar{y} + c$. Let $y = \sum_{i=0}^{7} y_i \cdot 2^i$, and consider the following three conditions, respectively.

1) When $i = 0$, from $y_0 \equiv (\bar{y}_0 + c_0) \pmod{2}$, one can immediately get $c_0 = 1$. Note the following two facts: i) when $y_0 = 0$, $\bar{y}_0 + c_0 = 2$, a carry bit is generated for the next bit, so $y_1 \equiv (\bar{y}_1 + c_1 + 1) \pmod{2}$ and $c_1 = 0$; ii) when $y_0 = 1$, $\overline{y_0} + c_0 = 1$, no carry bit is generated, so $y_1 \equiv (\bar{y}_1 + c_1) \pmod{2}$ and $c_1 = 1$.

Apparently, it is always true that $y_0 = c_1$. Also, a carry bit is generated if $c_1 = 0$ is observed.

2) When $i = 1$, if there exists a carry bit, set $c_1' = c_1 + 1 \in \{1, 2\}$; otherwise, set $c_1' = c_1 \in \{0, 1\}$. From $y_1 \equiv (\bar{y}_1 + c_1') \pmod 2$, one can immediately get $c_1' = 1$. Then, using the same method shown in the first condition, one has $y_1 = c_2$ and knows whether or not a carry bit is generated for $i = 2$. Repeat the above procedure for $i = 2 \sim 6$, one can uniquely determine that $y_i = c_{i+1}$.

3) When $i = 7$, it is always true that the carry bit does not occur, so $c_7' = 1$, and $y_7 \equiv 1$.

Combining the above three conditions, one can get $y = (1, c_7, \cdots, c_1)_2$, which results in $x = a \oplus (1, c_7, \cdots, c_1)_2$. ■

Assume that the two values of $B(i)$ derived from Eqs. (4.11a) and (4.11b) are $B_1(i)$ and $B_2(i)$, respectively. The following corollary shows that the two values have a deterministic relation: $B_2(i) = B_1(i) \oplus 128$.

Corollary 4.1: *Assume that a, b, c, x are all 8-bit integers, $a = \bar{b}$ and $c > 0$. Given two equations, $(a \oplus x) - (b \oplus x) = c$ and $(b \oplus x') - (a \oplus x') = c'$, if $c' = 256 - c$, then $x' = x \oplus 128$.*

Proof: Since $c + \bar{c} = 255$, one has $c' = 256 - c = \bar{c} + 1$. Let $c = \sum_{i=0}^{7} c_i \cdot 2^i$, and observe the first condition of the proof of Theorem 4.1. One can see that $c_0 = 1$, so $c_0' = \bar{c}_0 + 1 = 1$. Since there is no carry bit, one can deduce that $\forall\, i = 1 \sim 7$, $c_i' = \bar{c}_i$. Applying Theorem 4.1 for $(a \oplus x) - (b \oplus x) = c$, one can uniquely get $x = a \oplus (1, c_7, \cdots, c_1)_2$. Then, applying Theorem 4.1 for $(b \oplus x') - (a \oplus x') = c'$, one has $x' = b \oplus (1, c_7', \cdots, c_1')_2 = \bar{a} \oplus (1, \bar{c}_7, \cdots, \bar{c}_1)_2 = (a_7, \bar{a}_6 \oplus \bar{c}_7, \cdots, \bar{a}_0 \oplus \bar{c}_1)_2 = (a_7, a_6 \oplus c_7, \cdots, a_0 \oplus c_1)_2 = a \oplus (1, c_7, \cdots, c_1)_2 \oplus (1, 0, \cdots, 0)_2 = x \oplus 128$. Thus, this corollary is proved. ■

For any one of the two candidate values of $B(i)$, one can further get an equivalent chaotic state $\hat{x}(8i + 7)$ from $B(i)$, $f(i)$ and $f'(i)$ as follows:

$$\hat{x}(8i + 7) = 256 \cdot f'(i) - (f(i) \oplus B(i)) \equiv 256 \cdot x(8i + 7) \pmod{256} \ . \tag{4.12}$$

With $B(i)$ and $\hat{x}(8i + 7)$, the encryption formula Eq. (4.6) becomes

$$f'(i) = \frac{((f(i) \oplus B(i)) + \hat{x}(8i + 7)) \bmod 256}{256} \ , \tag{4.13}$$

and the decryption formula Eq. (4.7) becomes

$$f(i) = ((256 \cdot f'(i) - \hat{x}(8i + 7)) \bmod 256) \oplus B(i) \ . \tag{4.14}$$

Assume that $\hat{x}_1(8i+7)$ and $\hat{x}_2(8i+7)$ are calculated by Eq. (4.12), from $B_1(i)$ and $B_2(i)$, respectively. Then, we have the following proposition.

Proposition 4.1: *$(B_1(i), \hat{x}_1(8i+7))$ and $(B_2(i), \hat{x}_2(8i+7))$ are equivalent for the above encryption procedure Eq. (4.13), though only one corresponds to the correct value generated from the secret key. That is,*

$$((f(i) \oplus B_1(i)) + \hat{x}_1(8i+7)) \equiv ((f(i) \oplus B_2(i)) + \hat{x}_2(8i+7)) \pmod{256} .$$

Proof: From $B_1(i) = B_2(i) \oplus 128$, one has $f(i) \oplus B_1(i) = (f(i) \oplus B_2(i) \oplus 128)$. Then, following Lemma 4.3, it is true that $(f(i) \oplus B_1(i)) \equiv ((f(i) \oplus B_2(i)) + 128) \pmod{256}$. As a result, $\hat{x}_1(8i+7) = (256 \cdot f'(i) - (f(i) \oplus B_1(i))) \equiv (256 \cdot f'(i) - ((f(i) \oplus B_2(i)) - 128)) \pmod{256} \equiv (\hat{x}_2(8i+7) + 128) \pmod{256}$, which immediately leads to the following fact: $((f(i) \oplus B_1(i)) + \hat{x}_1(8i+7)) \equiv ((f(i) \oplus B_2(i)) + \hat{x}_2(8i+7)) \pmod{256}$. Thus, this proposition is proved. ■

Considering the symmetry of the encryption and decryption procedures, the above proposition immediately leads to a conclusion that $(B_1(i), \hat{x}_1(8i+7))$ and $(B_2(i), \hat{x}_2(8i+7))$ are also equivalent for the decryption procedure Eq. (4.14).

From the above analyses, with two chosen plaintexts f_1 and $f_2 = \bar{f}_1$, one can get the following two sequences: $\{B_1(i), \hat{x}_1(8i+7)\}_{i=0}^{N-1}$ and $\{B_2(i), \hat{x}_2(8i+7)\}_{i=0}^{N-1}$. Given a ciphertext $f' = \{f'(i)\}_{i=0}^{N-1}$, $\forall\, i = 0 \sim N-1$, one can use either $(B_1(i), \hat{x}_1(8i+7))$ or $(B_2(i), \hat{x}_2(8i+7))$ as an equivalent of the secret key to decrypt the i-th plain-byte $f(i)$, following Eq. (4.14). This means that the chaotic cipher under study is not sufficiently secure against the chosen-plaintext attack.

§4.4.2 Experiments

To demonstrate the feasibility of the proposed attack, some experiments have been performed for image encryption, with secret key $r = 1.99$, $x(0) = 0.41$ and $[S_0(0), \cdots, S_7(0)] = [1, -1, 1, -1, 1, -1, 1, -1]$. One plain-image "Lenna" of size 256×256 is chosen as f_1 and another plain-image is manually generated as follows: $f_2 = \bar{f}_1$. The two plain-images and their cipher-images are shown in Fig. 4.6. With the two chosen plain-images, two sequences, $\{B_1(i), \hat{x}_1(8i+7)\}_{i=0}^{256\cdot 256-1}$ and $\{B_2(i), \hat{x}_2(8i+7)\}_{i=0}^{256\cdot 256-1}$, are generated by using the above-mentioned algorithm. The first ten elements of the two sequences are given in Table 4.1. $\forall\, i = 0 \sim (256 \cdot 256 - 1)$, either $(B_1(i), \hat{x}_1(8i+7))$ or $(B_2(i), \hat{x}_2(8i+7))$ can be used to recover the plain-byte $f(i)$. As a result, the whole plain-image ("Peppers" in this test) can be recovered as shown in Fig. 4.6f.

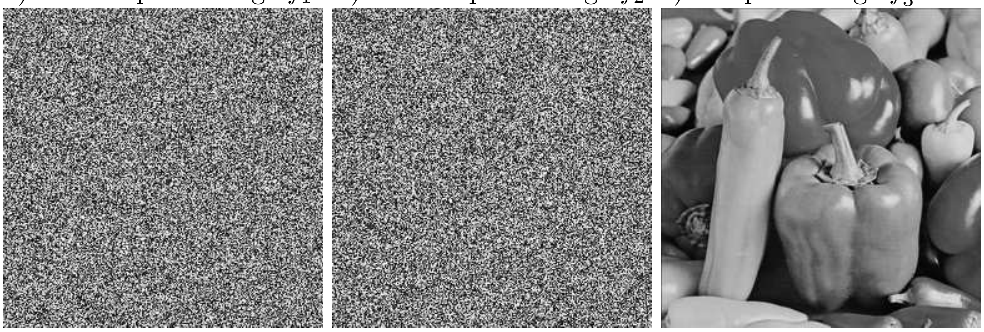
§4.5 Conclusion

In this chapter, the security of two neural-network-based encryption schemes have been investigated in detail. It is found that the two schemes can be successfully

Table 4.1: The first ten elements of $\{B_1(i), \hat{x}_1(8i+7)\}_{i=0}^{256\cdot 256-1}$ and $\{B_2(i), \hat{x}_2(8i+7)\}_{i=0}^{256\cdot 256-1}$.

i	0	1	2	3	4	5	6	7	8	9
$B_1(i)$	146	231	54	202	59	243	166	173	233	82
$B_2(i)$	18	103	182	74	187	115	38	45	105	210
$\hat{x}_1(8i+7)$	242.40	38.63	242.62	222.09	81.03	214.73	240.91	203.59	138.20	9.33
$\hat{x}_2(8i+7)$	114.40	166.63	114.62	94.09	209.03	86.73	112.91	75.59	10.20	137.33

a) Chosen plain-image f_1 c) Chosen plain-image f_2 e) A cipher-image f_3'



b) Cipher-image f_1' d) Cipher-image f_2' f) Recovered image f_3

Figure 4.6: The proposed chosen-plaintext attack.

broken with one known/chosen plaintext and two chosen plaintexts respectively. Both theoretical and experimental analyses have been given to support the above attacks. In addition, the Yen's scheme is not secure against brute-force attacks, some possible methods to enhance it are also discussed. It can be seen that the security of a encryption scheme is mainly dependent on the encryption structure instead of the underlying complex theory. In conclusion, the two schemes are not suggested in applications requiring a high level of security.

Chapter 5
Cryptanalysis of a Data Security Protection Scheme for VoIP

§5.1 Introduction

With the rapid development of the Internet and digital communication technologies, it becomes possible to make a telephone call with a PC connected to the Internet, or to connect the link between two telephones partially via the Internet. This technique is called voice over Internet protocol (VoIP in short) [87]. In fact, VoIP can be extended to realize many other services, such as fax over Internet protocol (FoIP), video teleconferences, and so on. Due to the obvious benefits and potential applications of the VoIP technology, it attracts more and more interests from both vendors and consumers.

Differing from the traditional telephony service, VoIP faces new risks in the networked world: since all data are transmitted over the Internet, any attack to digital computers can be used to break a VoIP system. Thus, it is very important to provide sufficient security for VoIP services with a reasonable cost. In [63, 64], a new VoIP protocol with a hierarchical data security protection (HDSP) scheme was proposed as a possible solution to the following two concerns: reducing continuous packet loss to avoid large voice corruption, and encrypting the transmitted voice to provide a high level of security. HDSP works under the control of a chaotic bit sequence, which is generated from the secret key by iterating a discrete-time chaotic map. In [63], it was claimed that HDSP can resist the known-plaintext attack.

The present chapter analyzes the security of HDSP in detail. Due to some insecure properties of the HDSP scheme, the following facts are found: 1) given n known plaintexts, only about $50/2^n$ percent of secret chaotic bits cannot be uniquely determined; 2) given only one specially chosen plaintext, all secret chaotic bits can be uniquely derived; 3) the secret key can be derived with a practically small complexity even when only one plaintext is known (or chosen). As a result, HDSP is very weak against known/chosen-plaintext attacks. In addition, it is found that the security of HDSP against the brute-force attack is not practically strong.

The organization of this chapter is as follows. §5.2 briefly introduce HDSP. §5.3 is the main body of this chapter, which focuses on the cryptanalysis of the HDSP scheme. Some experimental results are shown in §5.4 to support the theoretical results given in §5.3. §5.5 briefly discusses how to improve the security of HDSP. The last section concludes the chapter.

§5.2 The Hierarchical Data Security Protection (HDSP) Scheme

In the HDSP-based VoIP system, the encryption part is placed after speech encoder, and the decryption part is placed before the speech decoder. So, the plaintext of HDSP is the bit-stream encoded by the speech CODEC, not the raw voice signal. The ciphertext is obtained by performing the following two steps on the plaintext [63, 64]:

1. *the inter-frame interleaving (frame swapping)* – divide the plaintext into S_f-byte frames, and pseudo-randomly permute the orders of S_g continuous frames (i.e., every S_g frames compose an interleaving group);

2. *the intra-frame encryption (bit swapping and masking)* – for each input byte, pseudo-randomly permute some bits, and mask the 4 odd bits with XOR operations.

The frame/bit swapping and the bit masking operations are all controlled by a secret chaotic bit sequence $\{b(i)\}$, which is generated by iterating the chaotic Logistic map Eq. (2.8). Although the *inter-frame interleaving* was only intended for avoiding possible loss of continuous packets [63, 64], the use of secret chaotic bits has effectively made it one part of the whole encryption scheme.

For a plaintext $g = \{g(i)\}_{i=0}^{N-1}$, where $g(i)$ denotes the i-th byte of g, the HDSP encryption scheme can be described as follows (to better describe the original algorithm, some notations and definitions used in [63, 64] have been intentionally changed):

- *The secret key* is the control parameter μ and the initial condition $x(0)$ of the chaotic Logistic map, which are both represented in the 16-bit binary form.

- *The initialization procedure*: run the Logistic map from $x(0)$ to generate a chaotic sequence $x(i)_{i=0}^{\lceil L_b/16 \rceil -1}$, where L_b denotes the number of bits required during the following encryption procedure, and then extract the 16-bit representation of each chaotic state $x(i)$ to obtain a PRBS $\{b(i)\}_{i=0}^{L_b-1}$.

- *The encryption procedure* is composed of the following stages:

    1. *The inter-frame interleaving*:

– divide g into S_f-byte frames: $\{frame(i)\}_{i=0}^{N_f-1}$, where $N_f = \lfloor N/S_f \rfloor$;
– further divide g into S_g-frame groups: $\{group(i)\}_{i=0}^{N_g-1}$, where $N_g = \lfloor N_f/S_g \rfloor$;
– set $L = \lfloor \log_2 S_g \rfloor$ and $\Delta_L = S_g - 2^L$;
– all S_g frames in each group is permuted with S_g pseudo-random swapping operations: for $i = 0 \sim (N_g - 1)$ and for $j = 0 \sim (S_g - 1)$, swap $frame(i \cdot S_g + j)$ and $frame(i \cdot S_g + j')$, where

$$j' = \sum_{k=0}^{L-1} 2^k \cdot b(s+k) + \sum_{k=0}^{\Delta_L - 1} b(s+L+k), \tag{5.1}$$

and $s = (i \cdot S_g + j) \cdot L$.

* Note: in this stage, totally $(N_g \cdot S_g \cdot L + \Delta_L)$ chaotic bits are required: $b(0) \sim b(N_g \cdot S_g \cdot L + \Delta_L - 1)$.

2. *The intra-frame encryption* – assuming $g^* = \{g^*(i)\}_{i=0}^{N-1}$ is the output plaintext of the inter-frame interleaving stage, the ciphertext $g' = \{g'(i)\}_{i=0}^{N-1}$ is determined in the following two steps:
   – *pseudo-randomly swap the 4 most significant bits (MSB-s) and the 4 least significant bits (LSB-s) of each byte* $g^*(i) = \sum_{k=0}^{7} d_k^*(i) \cdot 2^k$ *to get an intermediate byte* $g^{**}(i) = \sum_{k=0}^{7} d_k^{**}(i) \cdot 2^k$: $\forall\, k = 0 \sim 3$,

$$\left(d_k^{**}(i), d_{k+4}^{**}(i)\right) = Swap_{b(4i+k)}\left(d_k^*(i), d_{k+4}^*(i)\right). \tag{5.2}$$

   – *mask the 4 odd bits of* $g^{**}(i)$ *to get the cipher-byte* $g'(i) = \sum_{k=0}^{7} d'_k(i) \cdot 2^k$: $\forall\, k = 1, 3, 5, 7$,

$$d'_k(i) = d_k^{**}(i) \oplus b(4i+k). \tag{5.3}$$

   * Note: in this stage, totally $(4N + 2)$ chaotic bits are required: $b(0) \sim b(4N-1), b(4N+1), b(4N+3)$.

* Note: in the whole encryption procedure, totally $\max(N_g \cdot S_g \cdot L + \Delta_L, 4N + 2)$ chaotic bits are required. Since the index of the last required chaotic bit is $\max(N_g \cdot S_g \cdot L + \Delta_L - 1, 4N + 3)$, one has $L_b = \max(N_g \cdot S_g \cdot L + \Delta_L, 4N + 4)$.

- *The decryption procedure* is the reversion of the encryption procedure, and can be briefly described as follows:

  1. *The inverse intra-frame decryption (bit swapping and masking operations)*:

- *mask the 4 odd bits of the cipher-byte* $g'(i) = \sum_{k=0}^{7} d'_k(i) \cdot 2^k$ *to restore the intermediate byte* $g^{**}(i)$*;*
- *pseudo-randomly swap the 4 MSB-s and the 4 LSB-s of each byte* $g^{**}(i) = \sum_{k=0}^{7} d_k^{**}(i) \cdot 2^k$ *to restore another intermediate byte* $g^{*}(i) = \sum_{k=0}^{7} d_k^{*}(i) \cdot 2^k$.

2. *The same inter-frame interleaving* is inversely exerted on $g^* = \{g^*(i)\}_{i=0}^{N-1}$ to restore the plaintext $g = \{g(i)\}_{i=0}^{N-1}$, where the term "inversely" means that $i = 0 \sim (N_g - 1)$ and $j = (S_g - 1) \sim 0$.

§5.3 Cryptanalysis of HDSP

§5.3.1 The Brute-Force Attack

The secret key in HDSP is $(\mu, x(0))$, which is represented by $2 \cdot 16 = 32$ secret bits. Thus, the size of the key space is 2^{32}. Considering that the complexity of verifying each key is equal to the one of the encryption procedure of HDSP – $O(N)$[63, Sec. 4.1], the total complexity of the brute-force attack is $O\left(N \cdot 2^{32}\right)$. However, because not all values of μ can ensure the chaoticity of the Logistic map[82], actually the size of the key space is smaller than 2^{32} and the attack complexity is smaller than $O\left(N \cdot 2^{32}\right)$.

Here, note the following facts: 1) in the known-plaintext attack scenario, the key can be validated by simply comparing the decrypted signal with the known plaintext; 2) in the ciphertext-only attack scenario, the key has to be validated by some inherent features of the plaintexts, which may increase the attack complexity to some extent, but generally the complexity will be not larger than $O\left(N^2 \cdot 2^{32}\right)$.

To guarantee a high level of security in today's digital world, a complexity of order $O\left(2^{100}\right)$ is required[1]. Apparently, $O\left(N \cdot 2^{32}\right)$ is too small to reach such a goal. An attacker can easily find the secret key within a few hours (or even minutes) using a PC with a 1G CPU, if N is not too large (when N is too large, checking a small segment of the plaintext can work). To ensure a high level of security, 64-bit binary representations of μ and $x(0)$ are suggested to provide a complexity of order $O\left(2^{128}\right)$ against brute-force attacks.

§5.3.2 The Known-Plaintext Attack

In [63, Sec. 4.2], it was claimed that HDSP can efficiently resist the known-plaintext attack*. In this subsection, this claim is re-evaluated, concluding that

* As introduced in §1.2, the known-plaintext attack becomes possible when an attacker can temporarily access the encryption machine. Under such a condition, it is reasonable to assume that the attacker can get the plaintext output from the speech CODEC, since details

HDSP is not sufficiently secure against known-plaintext attacks. Basically, with n known plaintexts, from the probabilistic point of view, about $\left(100 - \frac{50}{2^n}\right)$ % of all chaotic bits can be correctly restored and only $16/2^n$ bits have to be exhaustively guessed to break the secret key. Even with $n = 1$, the complexity of breaking the secret key is sufficiently small: only $O\left(2^8\right)$. Once the secret key is uncovered, the HDSP scheme can be completely broken. Furthermore, because HDSP works like a stream cipher, even without deriving the secret key, one can still use the partially-reconstructed chaotic bit sequence to partially (almost completely when n is relatively large) recover the unknown plaintexts.

Similar to the encryption procedure of HDSP, the known-plaintext attack also works in two stages: 1) break the inter-frame interleaving; 2) break the intra-frame encryption. The two stages provide a partial (or total) reconstruction of the secret chaotic bit sequence, which can be used to restore the secret key. Generally, if N is sufficiently large, only one known plaintext is enough for an attacker to restore the sub-key μ and an equivalent of another sub-key $x(0)$. In addition, if two or more plaintexts of size N are known, it is possible to correctly restore most chaotic bits, which can also be directly used as a replacement of the secret key to decrypt the ciphertexts encrypted with the same key (if their sizes are not larger than N).

Next, some important properties of the intra-frame encryption stage of HDSP are discussed, which are the essential reasons for the insecurity of HDSP against known/chosen-plaintext attacks. These insecure properties are structural defects of the encryption procedure used in HDSP, independent of any specifics of the speech data and the speech CODEC. The theoretical analysis of the proposed attacks will be based on the assumption that the plaintext (i.e., the output of the speech CODEC) is a uniformly-distributed random source. When the plaintext does not obey the uniform distribution, our experiments show that the performance of the proposed attacks may be somewhat better or worse (see Sec. 5).

§5.3.2.1 Some insecure properties of the HDSP encryption scheme

In the intra-frame encryption stage of HDSP, the i-th byte of the input signal, $g^*(i) = \sum_{k=0}^{7} d_k^*(i) \cdot 2^k$, and the i-th byte of the output signal (i.e., the ciphertext), $g'(i) = \sum_{k=0}^{7} d_k'(i) \cdot 2^k$, satisfy the following properties.

Property 5.1: *For $k = 0, 2$: a) $d_k^*(i) + d_{k+4}^*(i) \equiv d_k'(i) + d_{k+4}'(i)$; b) when $d_k^*(i) \neq d_{k+4}^*(i)$, one has $b(4i + k) = d_k^*(i) \oplus d_k'(i) = d_{k+4}^*(i) \oplus d_{k+4}'(i)$.*

Proof: From Eqs. (5.2) and (5.3), one can see that the 4 even bits are swapped without being masked. Thus, for $k = 0, 2$, $d_k^*(i) + d_{k+4}^*(i)$ remains unchanged after the intra-frame encryption, i.e., $d_k^*(i) + d_{k+4}^*(i) \equiv d_k'(i) + d_{k+4}'(i)$.

of the CODEC are not kept secret in the HDSP-based VoIP system.

When $d_k^*(i) \neq d_{k+4}^*(i)$, which means that $d_k^*(i) = \overline{d_{k+4}^*(i)}$ and $d_{k+4}^*(i) = \overline{d_k^*(i)}$, the bit swapping operation $Swap_w(a,b)$ becomes (in this case, $a \neq b$)

$$Swap_w(a,b) = \begin{cases} (a,b) = (a \oplus 0, b \oplus 0) = (a \oplus w, b \oplus w), & w = 0, \\ (b,a) = (\bar{a}, \bar{b}) = (a \oplus 1, b \oplus 1) = (a \oplus w, b \oplus w), & w = 1. \end{cases} \tag{5.4}$$

That is, $Swap_w(a,b) \equiv (a \oplus w, b \oplus w)$. Then, one has $d'_k(i) = d_k^*(i) \oplus b(4i+k)$ and $d'_{k+4}(i) = d_{k+4}^*(i) \oplus b(4i+k)$, which immediately leads to $b(4i+k) = d_k^*(i) \oplus d'_k(i) = d_{k+4}^*(i) \oplus d'_{k+4}(i)$. Thus, the proof is completed. ■

Property 5.2: *For $k = 1,3$: a) when $d_k^*(i) = d_{k+4}^*(i)$, one has $b(4i+k) = d_k^*(i) \oplus d'_k(i)$ and $b(4(i+1)+k) = d_{k+4}^*(i) \oplus d'_{k+4}(i)$; b) when $d_k^*(i) \neq d_{k+4}^*(i)$, one has $d'_k(i) \equiv d_k^*(i)$ and $b(4i+k) \oplus b(4(i+1)+k) = d_{k+4}^*(i) \oplus d'_{k+4}(i)$.*

Proof: The two conditions are proved separately.

a) When $d_k^*(i) = d_{k+4}^*(i)$, the bit swapping operation disappears and the cipher-bit is determined by the masking operation only: $d'_k(i) = d_k^*(i) \oplus b(4i+k)$ and $d'_{k+4}(i) = d_{k+4}^*(i) \oplus b(4i+k+4)$. That is, $b(4i+k) = d_k^*(i) \oplus d'_k(i)$ and $b(4(i+1)+k) = d_{k+4}^*(i) \oplus d'_{k+4}(i)$.

b) When $d_k^*(i) \neq d_{k+4}^*(i)$, from Eq. (5.4), $Swap_w(a,b) \equiv (a \oplus w, b \oplus w)$, so one can get $d_k^{**}(i) = d_k^*(i) \oplus b(4i+k)$ and $d_{k+4}^{**}(i) = d_{k+4}^*(i) \oplus b(4i+k)$. Substituting the two results into Eq. (5.3), one has the following results:

- $d'_k(i) = d_k^{**}(i) \oplus b(4i+k) = (d_k^*(i) \oplus b(4i+k)) \oplus b(4i+k) \equiv d_k^*(i)$;
- $d'_{k+4}(i) = d_{k+4}^{**}(i) \oplus b(4i+k+4) = \left(d_{k+4}^*(i) \oplus b(4i+k)\right) \oplus b(4i+k+4) = d_{k+4}^*(i) \oplus (b(4i+k) \oplus b(4i+k+4))$, which is equivalent to $b(4i+k) \oplus b(4(i+1)+k) = d_{k+4}^*(i) \oplus d'_{k+4}(i)$.

From the above two conditions, the property is proved. ■

Property 5.3: *For $k = 1,3$, $\forall\, i = 0 \sim N-2$, if $d_k^*(i) = d_{k+4}^*(i)$ and $d_k^*(i+1) = d_{k+4}^*(i+1)$, then $b(4(i+1)+k) = d_{k+4}^*(i) \oplus d'_{k+4}(i) = d_k^*(i+1) \oplus d'_k(i+1)$.*

Proof: This property is a natural corollary of the above Property 5.2a. ■

In the following, it will be shown that the above properties make the known-plaintext attack feasible in practice.

§5.3.2.2 Breaking the inter-frame interleaving

The frame swapping operations in the inter-frame interleaving stage actually correspond to a pseudo-random and secret frame-permutation in each group.

One can represent the permutation of $group(i)$ by a permutation vector $\boldsymbol{v}(i) = [v(i,0),\cdots,v(i,S_g-1)]$, where $\forall\ j_1 \neq j_2$, $v(i,j_1) \neq v(i,j_2)$. With the permutation vector, the inter-frame interleaving of $group(i)$ can be described as follows: $\forall\ j = 0 \sim (S_g-1)$, the j-th frame in $group(i)$ is permuted to be the $v(i,j)$-th frame, i.e., $frame(i\cdot S_g + j)$ is permuted to be $frame(i\cdot S_g + v(i,j))$. Basically, in this restoration stage, the goal is to restore the permutation vectors of all groups: $\boldsymbol{v}(0) \sim \boldsymbol{v}(N_g-1)$.

To restore the permutation vectors, at least an input signal (i.e., the plaintext g) and the corresponding output signal (i.e., g^*) should be known. However, in the known-plaintext attack, generally the intermediate signal g^* is not known. Fortunately, due to Property 5.1a proved above, some information of g^* can be obtained from the ciphertext g'. This generally is enough to restore the permutation vectors.

Next, define three sequences, $\widehat{g} = \{\widehat{g}(i)\}_{i=0}^{N-1}$, $\widehat{g}^* = \{\widehat{g}^*(i)\}_{i=0}^{N-1}$ and $\widehat{g}' = \{\widehat{g}'(i)\}_{i=0}^{N-1}$, where

$$\widehat{g}(i) = (d_0(i)+d_4(i)) + (d_2(i)+d_6(i))\cdot 3 \in \{0,\cdots,8\}, \tag{5.5}$$

$$\widehat{g}^*(i) = (d_0^*(i)+d_4^*(i)) + (d_2^*(i)+d_6^*(i))\cdot 3 \in \{0,\cdots,8\}, \tag{5.6}$$

$$\widehat{g}'(i) = (d_0'(i)+d_4'(i)) + (d_2'(i)+d_6'(i))\cdot 3 \in \{0,\cdots,8\}. \tag{5.7}$$

From Property 5.1a, one can see that $\widehat{g}^* = \widehat{g}'$. Considering that the inter-frame interleaving stage does not change the values of all frames but their positions, one can use $\widehat{g}$ and $\widehat{g}'$ to restore the permutation vectors. To do so, one has to divide both $\widehat{g}$ and $\widehat{g}'$ into N_f frames, $\left\{\widehat{frame}(i)\right\}_{i=0}^{N_f-1}$, $\left\{\widehat{frame}'(i)\right\}_{i=0}^{N_f-1}$, and N_g groups, $\left\{\widehat{group}(i)\right\}_{i=0}^{N_g-1}$, $\left\{\widehat{group}'(i)\right\}_{i=0}^{N_g-1}$, respectively, in the same way as the encryption procedure of HDSP. Now, $\forall\ i = 0 \sim (N_g-1)$, the permutation vector of $group(i)$ can be estimated as follows*

- *Step 1*: For $j = 0 \sim (S_g-1)$, calculate $R_f(i,j) = \sum_{k=0}^{S_f-1} \widehat{g}_f(i,j,k)\cdot 9^k$ and $R_f'(i,j) = \sum_{k=0}^{S_f-1} \widehat{g}_f'(i,j,k)\cdot 9^k$, where $\widehat{g}_f(i,j,k)$ and $\widehat{g}_f'(i,j,k)$ denote the k-th byte of the j-th frame in $\widehat{group}(i)$ and in $\widehat{group}'(i)$, respectively.

- *Step 2*: Compare the values of $R_f(i,0) \sim R_f(i,S_g-1)$ and $R_f'(i,0) \sim R_f'(i,S_g-1)$ to get two partitions of the index-set $\mathbb{S}_g = \{0,\cdots,S_g-1\}$: $\{\Lambda(k)\}_{k=0}^{K-1}$ and $\{\Lambda'(k)\}_{k=0}^{K-1}$, where K is the number of different values

*A more general description on this algorithm of breaking secret permutations with a number of known/chosen plaintexts can be found in §2.2.

in the set $\{R_f(i,0), \cdots, R_f(i, S_g-1)\} = \{R'_f(i,0), \cdots, R'_f(i, S_g-1)\}$ and $\forall\, a, b \in \Lambda(k), \forall\, a', b' \in \Lambda'(k)$, $R_f(i,a) = R_f(i,b) = R'_f(i,a') = R'_f(i,b')$.

  * Note 1: $\{\Lambda(k)\}_{k=0}^{K-1}$ and $\{\Lambda'(k)\}_{k=0}^{K-1}$ are partitions of $\mathbb{S}_g$, which means that $\bigcup_{k=0}^{K-1} \Lambda(k) = \bigcup_{k=0}^{K-1} \Lambda'(k) = \mathbb{S}_g$ and $\forall\, k_1 \neq k_2$, $\Lambda(k_1) \cap \Lambda(k_2) = \Lambda'(k_1) \cap \Lambda'(k_2) = \varnothing$.
  * Note 2: because the frame swapping operations do not change the value of $R_f(i,j)$, it is obvious that $\forall\, k = 0 \sim (K-1)$, the cardinality (size) of $\Lambda(k)$ is equal to that of $\Lambda'(k)$, i.e., $\#(\Lambda(k)) = \#(\Lambda'(k))$.

- *Step 3*: Derive an estimation of the permutation vector $\boldsymbol{v}(i)$ of $group(i)$ from $\{\Lambda(k)\}_{k=0}^{K-1}$ and $\{\Lambda'(k)\}_{k=0}^{K-1}$, under the following two conditions:
  - *Condition 1*: if $\forall\, k \in \{0, \cdots, K-1\}$, $\Lambda(k)$ contains only one element, i.e., $\#(\Lambda(k)) = \#(\Lambda'(k)) = 1$ (and $K = S_g$), then the permutation vector $\boldsymbol{v}(i)$ of $group(i)$ can be *uniquely* derived: $\forall\, k = 0 \sim (K-1 = S_g-1)$, and $v(i, \Lambda(k))$ is set to be the only element in $\Lambda'(k)$.
  - *Condition 2*: if $\exists k \in \{0, \cdots, K-1\}$, $\Lambda(k)$ contains more than one elements, i.e., $\#(\Lambda(k)) = \#(\Lambda'(k)) \geq 2$ (and $K < S_g$), then the permutation vector $\boldsymbol{v}(i)$ cannot be uniquely derived from $\{\Lambda(k)\}_{k=0}^{K-1}$ and $\{\Lambda'(k)\}_{k=0}^{K-1}$. But one can get an estimated permutation vector $\widetilde{\boldsymbol{v}}(i) = [\widetilde{v}(i,0), \cdots, \widetilde{v}(i, S_g-1)]$ as follows: for $k = 0 \sim (K-1)$, determine a one-to-one mapping $f_{\Lambda(k)} : \Lambda(k) \to \Lambda'(k)$, and then $\forall\, a \in \Lambda(k)$, set $\widetilde{v}(i,a) = f_{\Lambda(k)}(a)$.

Under *Condition 1* in Step 3, the permutation vector can be correctly derived without any error. However, under *Condition 2*, the estimated permutation vector $\widetilde{\boldsymbol{v}}$ may be wrong with a non-negligible probability. This is due to the following fact: $\forall\, k = 0 \sim (K-1)$, there are $(\#(\Lambda(k)))!$ possible mappings of $f_{\Lambda(k)}$, so there are $\prod_{k=0}^{K-1}(\#(\Lambda(k)))!$ possible estimations of $\boldsymbol{v}(i)$, among which only one is the correct permutation vector $\boldsymbol{v}$.

Now, let us study the occurrence probability of *Condition 2*. Assuming the number of all possible values of $R_f(i,j)$ is N_{R_f}, this probability can be easily calculated as follows:

$$Prob[\text{Condition 2 occurs}] = 1 - \left(1 - \frac{0}{N_{R_f}}\right) \cdot \left(1 - \frac{1}{N_{R_f}}\right) \cdots \left(1 - \frac{S_g - 1}{N_{R_f}}\right). \tag{5.8}$$

From the definition of R_f, one has $N_{R_f} = 9^{S_f}$. In most cases, 9^{S_f} is much larger than S_g, so the occurrence probability of *Condition 2* is so small that it can be simply ignored in practice. When *Condition 2* cannot be ignored (i.e., when 9^{S_f} is not much larger than S_g), the following constraints in the intra-frame encryption stage can be used to detect wrong estimations:

- *Constraint 1*: $k = 1, 3$, $\forall\, i = 0 \sim (N-1)$, if $d_k^*(i) \neq d_{k+4}^*(i)$, $d_k'(i) = d_k^*(i)$.
- *Constraint 2*: $k = 1, 3$, $\forall\, i = 0 \sim (N-2)$, if $d_k^*(i) = d_{k+4}^*(i)$ and $d_k^*(i+1) = d_{k+4}^*(i+1)$, then $d_{k+4}^*(i) \oplus d_{k+4}'(i) = d_k^*(i+1) \oplus d_k'(i+1) = b(4(i+1)+k)$.
- *Constraint 3*: As shown later in the next sub-subsection, the two chaotic bit sequences $\{b(4i+1)\}_{i=0}^{N}$ and $\{b(4i+3)\}_{i=0}^{N}$ are completely correlated, i.e., they satisfy Eq. (5.9) below. It will be precisely explained there as how to use this constraint to detect wrong permutation vectors.

The above three constraints can be deduced from Properties 5.2 and 5.3. Once any of these constraints is violated, it can be immediately asserted that the current permutation vector is wrong.

Finally, the following fact should be noticed: in *Condition 2*, the larger the number of the known plaintexts is, the less the probability of $\widetilde{\boldsymbol{v}}(i) \neq \boldsymbol{v}(i)$ will be. Given n known plaintexts $g_1 \sim g_n$, the number of all possible combinations of $R_{f,1}(i,j) \sim R_{f,n}(i,j)$ becomes $N_{R_f}^n = 9^{nS_f}$, which means that the probability that *Condition 2* occurs exponentially decrease as n increases. That is, the probability of getting a wrong permutation vector will be exponentially decrease, so it is negligible when n is sufficiently large*.

§5.3.2.3 Breaking the intra-frame encryption

Once the inter-frame interleaving is correctly broken, one can get the intermediate signal $g^* = \{g^*(i)\}_{i=0}^{N-1}$ from the plaintext g successfully. With g^* and the ciphertext g', the intra-frame encryption can also be partially (or even totally) broken. During this breaking stage, one can partially (or even totally) reconstruct the secret chaotic bit sequence, which can be used to further derive the secret key. The following properties, summarized above, are now available for an attacker to break the intra-frame encryption:

- *Property 5.1b*: $k = 0, 2$, when $d_k^*(i) \neq d_{k+4}^*(i)$, $b(4i+k) = d_k^*(i) \oplus d_k'(i) = d_{k+4}^*(i) \oplus d_{k+4}'(i)$;
- *Property 5.2a*: $k = 1, 3$, when $d_k^*(i) = d_{k+4}^*(i)$, $b(4i+k) = d_k^*(i) \oplus d_k'(i)$ and $b(4(i+1)+k) = d_{k+4}^*(i) \oplus d_{k+4}'(i)$;

*Generally speaking, "n is sufficiently large" if $n \gg (\log_9 S_g)/S_f$, which ensures that $9^{nS_f} \gg S_g$.

- *Property 5.2b*: $k = 1, 3$, when $d_k^*(i) \neq d_{k+4}^*(i)$, $b(4i+k) \oplus b(4(i+1)+k) = d_{k+4}^*(i) \oplus d_{k+4}'(i)$.

Based on the above three properties, we can estimate how many chaotic bits one is able to get in a known-plaintext attack. Without loss of generality, assume that each bit in $g^*(i)$ and $g'(i)$ distributes uniformly over $\{0, 1\}$ and any two bits are independent of each other. Under this assumption, one can deduce the following results.

- *Even bits ($k = 0, 2$, from Property 5.1b)*:
  - *with only one known plaintext*: the probability of $d_k^*(i) \neq d_{k+4}^*(i)$ is $\frac{1}{2}$, so averagely 50% of all even chaotic bits can be correctly restored;
  - *with $n > 1$ known plaintexts*: when $d_k^*(i) \neq d_{k+4}^*(i)$ holds for at least one plaintext, the bit $b(4i+k)$ can be correctly restored, and then it can be deduced that the probability of the above event is $1 - \left(\frac{1}{2}\right)^n$.
- *Odd bits ($k = 1, 3$, from Property 5.2a/b)*:
  - *with only one known plaintext*: $\forall\, i = 0 \sim (N-1)$, the value of $b(4i + k) \oplus b(4(i+1)+k)$ can be correctly determined, i.e., one can get a new sequence $\left\{b_k^{\oplus}(i) = b(4i+k) \oplus b(4(i+1)+k)\right\}_{i=0}^{N-1}$. Apparently, if only one bit $b(4i^*+k)$ is known, the whole bit sequence $\{b(4i+k)\}_{i=0}^{N}$ can be correctly restored with the deterministic sequence $\{b_k^{\oplus}(i)\}_{i=0}^{N-1}$ as follows:

$$b(4i^*+k)\begin{cases} \xrightarrow{\oplus b_k^{\oplus}(i^*-1)} b(4(i^*-1)+k) \xrightarrow{\oplus b_k^{\oplus}(i^*-2)} \cdots \xrightarrow{\oplus b_k^{\oplus}(0)} b(4\cdot 0+k), \\ \xrightarrow{\oplus b_k^{\oplus}(i^*)} b(4(i^*+1)+k) \xrightarrow{\oplus b_k^{\oplus}(i^*+1)} \cdots \xrightarrow{\oplus b_k^{\oplus}(N-1)} b(4N+k). \end{cases} \tag{5.9}$$

    From Property 5.2a above, two bits $b(4i+k)$ and $b(4(i+1)+k)$ can be correctly restored when $d_k^*(i) = d_{k+4}^*(i)$. Thus, it can be deduced that the probability that at least two bits in $\{b(4i+k)\}_{i=0}^{N}$ are correctly restored is

$$1 - \left(Prob[d_k^*(i) \neq d_{k+4}^*(i)]\right)^N = 1 - \frac{1}{2^N}.$$

    Since N is generally sufficiently large, it is probabilistically true in almost all cases that all odd bits can be correctly restored.
    - Note 1: even under an extreme condition where no intermediate byte $g^*(i)$ satisfying $d_k^*(i) = d_{k+4}^*(i)$, one can randomly guess the value of any bit $b(4i+k)$ and then get the whole bit sequence $\{b(4i+k)\}_{i=0}^{N}$. In this case, at most four guesses (two for $k = 1$ and

the other two for $k = 3$) are needed to correctly restore all odd bits. In this sense, all odd bits can always be correctly reconstructed.

* Note 2 (a precise explanation of *Constraint 3*): assuming that two bytes $g^*(i_1)$ and $g^*(i_2)$ satisfy $d_k^*(i_1) = d_{k+4}^*(i_1)$ and $d_k^*(i_2) = d_{k+4}^*(i_2)$ and all in-between bytes $g^*(i_1+1) \sim g^*(i_2-1)$ do not satisfy this condition, where $i_2 \geq i_1 + 2$, the sub-sequence $\{b(4i + k\}_{i=i_1+1}^{i_2-1}$ can be uniquely derived by using Eq. (5.9) twice:

$$\begin{aligned} b(4i_1+k) \Rightarrow \quad & b(4(i_1+1)+k) \rightarrow \cdots \rightarrow b(4(i_2-1)+k), \\ & b(4(i_1+1)+k) \leftarrow \cdots \leftarrow b(4(i_2-1)+k) \quad \Leftarrow b(4i_2+k). \end{aligned}$$

If the two derived sub-sequences are not identical, it can be asserted that at least one permutation of the $frame(s)$ between $g(i_1)$ and $g(i_2)$ is wrong, i.e., at least one permutation vector of the $group(s)$ between $g(i_1)$ and $g(i_2)$ is wrong, and, when there is only one group between $g(i_1)$ and $g(i_2)$, the permutation vector of this group must be wrong.

– *with $n > 1$ known plaintexts*: the probability that at least two bits in $\{b(4i + k)\}_{i=0}^{N}$ can be correctly restored is $\left(1 - \dfrac{1}{2^{n \cdot N}}\right)$.

As a summary, when only one plaintext is known, averagely 50% of even bits and all odd bits, i.e., 75% of all bits in $\{b(i)\}_{i=0}^{4N-1}$ and the last two bits $b(4N+1), b(4N+3)$, can be correctly restored; furthermore, when $n \geq 1$ plaintexts are known, the percentage of correctly restored bits in $\{b(i)\}_{i=0}^{4N-1}$ becomes

$$\left(50 + 25 + \frac{25}{2} + \cdots + \frac{25}{2^{n-1}}\right)\% = \left(100 - \frac{50}{2^n}\right)\%,$$

which exponentially approaches 100% as n increases. For the rest unrecovered chaotic bits, one can randomly guess their values, and averagely half of the bits can be correctly matched to the true values. That is, the correctly restored bits in $\{b(i)\}_{i=0}^{4N-1}$ will reach $\left(100 - \frac{50}{2^{n+1}}\right)\%$, which is 87.5% for $n = 1$.

In addition, if some permutation vectors have been uniquely determined in the stage of breaking the inter-frame interleaving, the corresponding guessed bits can be checked with Eq. (5.1), and the correctly restored bits can be even more. In addition, since the wrong bits distribute randomly within the whole bit sequence and since human ears have a high capability to bear large audio noises, their negative influence on the quality of the voice data may not be so much, as verified by our experiments to be discussed later.

Finally, let us study the success probability of the decryption with the partially reconstructed chaotic bit sequence $b(0) \sim b(4N-1), b(4N+1), b(4N+3)$, when only one plaintext is known (i.e., $n = 1$). Since the two even bits $b(4i), b(4i+2)$, corresponding to a plain-byte $g(i)$, are correctly restored with probability $\left(\frac{1}{2}\right)^2 = \frac{1}{4}$, it seems that the probability of correctly decrypting an unknown plain-byte $\widetilde{g}(i) = \sum_{k=0}^{7} \widetilde{d}_k(i) \cdot 2^k$ should also be $\frac{1}{4}$. However, such an "intuition" is not true, because this probability is actually the addition of the following four probabilities:

- the probability that $b(4i), b(4i+2)$ are both correct, which is $\frac{1}{4}$;
- the probability that $b(4i)$ is correct, but $b(4i+2)$ is incorrect, and $\widetilde{d}_2^*(i) = \widetilde{d}_6^*(i)$, which is $\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$;
- the probability that $b(4i)$ is incorrect, but $b(4i+2)$ is correct, and $\widetilde{d}_0^*(i) = \widetilde{d}_4^*(i)$, which is also $\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$;
- the probability that $b(4i), b(4i+2)$ are both incorrect, $\widetilde{d}_0^*(i) = \widetilde{d}_4^*(i)$ and $\widetilde{d}_2^*(i) = \widetilde{d}_6^*(i)$, which is $\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{16}$.

So, the final probability of correctly decrypting a plain-byte is $\frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{16} = \frac{9}{16}$. In a similar way, this probability with $n(\geq 1)$ known plaintexts can be calculated to be

$$\begin{aligned} P(n) &= \left(1 - \frac{1}{2^n}\right)^2 + \left(1 - \frac{1}{2^n}\right) \cdot \frac{1}{2^n} \cdot \frac{1}{2} + \frac{1}{2^n} \cdot \left(1 - \frac{1}{2^n}\right) \cdot \frac{1}{2} + \frac{1}{2^n} \cdot \frac{1}{2^n} \cdot \frac{1}{2} \cdot \frac{1}{2} \\ &= 1 - \frac{1}{2^n} + \frac{1}{2^{2n+2}} = 1 - \frac{2}{2^{n+1}} + \left(\frac{1}{2^{n+1}}\right)^2 = \left(1 - \frac{1}{2^{n+1}}\right)^2. \end{aligned}$$

Considering each undetermined chaotic bit is identical with the original bit with probability $\frac{1}{2}$, one has

- $Prob[b(4i) \text{ is incorrect}] = Prob[b(4i+2) \text{ is incorrect}] = \frac{1}{2} \cdot \frac{1}{2^n} = \frac{1}{2^{n+1}}$;
- $Prob[b(4i) \text{ is correct}] = Prob[b(4i+2) \text{ is correct}] = 1 - \frac{1}{2^{n+1}}$.

Then, the probability of correctly decrypting a plain-byte in real attacks will become $P'(n) = P(n+1) = \left(1 - \frac{1}{2^{n+2}}\right)^2$. This result agrees with our experiments well (see Table 5.2 below).

§5.3.2.4 Completely breaking the encryption scheme (breaking the secret key)

With the reconstructed permutation vectors $\boldsymbol{v}(0) \sim \boldsymbol{v}(N_g - 1)$ and the partially restored chaotic bit sequence $b(0) \sim b(4N-1), b(4N+1), b(4N+3)$, a ciphertext g' can be decrypted to get an estimated plaintext $\widetilde{g} = \{\widetilde{g}(i)\}_{i=0}^{N-1}$ as follows:

- use the partially reconstructed chaotic bits to cancel the intra-frame encryption;
- derive the inverse permutation vector of each interleaving group and use it to cancel the inter-frame interleaving.

In the previous sub-subsection, it was shown that the breaking performance is not satisfactory because about 25% of plain-bytes cannot be correctly decrypted when only one plaintext is known. Although the performance can be exponentially enhanced with more known plaintexts, the above procedure cannot decrypt any plain-byte beyond the position N. Therefore, to thoroughly break the HDSP encryption scheme, one has to break the secret key itself. In the following, we show how to break the secret key from some consequent partially-reconstructed chaotic bits.

a) Breaking the initial condition $x(0)$ or an equivalent $x(i)$

Recalling the generation of chaotic bits in the initialization procedure of HDSP, one can see that each chaotic state $x(i)$ can be represented in the binary form as $0.b(16i+0)\cdots b(16i+15)$. Thus, if the first 16 chaotic bits are all correctly restored, one can directly get the initial condition $x(0) = 0.b(0)\cdots b(15)$. However, following the discussion in the previous sub-subsection, generally not all even bits in $b(0) \sim b(15)$ can be uniquely determined. To restore these bits, one has to exhaustively guess their values. Assuming the number of undetermined bits is $m \in \{0, \cdots, 8\}$, the guessing complexity is $O(2^m)$. Averagely, $m = \frac{1}{4}\cdot 16 = 4$, and the searching complexity is $O\left(2^4\right)$, which is practically small for PC-s. Even under the worst condition, $m = 8$, the guessing complexity is only $O\left(2^8\right)$.

When $x(0)$ is enhanced to be represented with $B > 16$ bits and all the $B' \leq B$ bits are extracted to generate the chaotic bit sequence $\{b(i)\}$, the guessing complexity under the worst condition will be $O\left(2^{\frac{B'}{2}}\cdot 2^{B-B'}\right) = O\left(2^{B-\frac{B'}{2}}\right)$. If $\left(B-\frac{B'}{2}\right)$ is sufficiently large*, the guessing complexity will be too large for the exhaustive guess of $x(0)$ on PC-s. In this case, instead of guessing $x(0)$, one can try to find another chaotic state $x(i) = 0.b(i\cdot B')\cdots b(i\cdot B' + (B'-1))$ as an equivalent of the sub-key $x(0)$, where the number of undetermined bits is less than $m < \frac{B'}{4}$, to get a reduced guessing complexity not greater than $O\left(2^{B-B'+m}\right) < O\left(2^{B-\frac{3B'}{4}}\right)$. With $x(i)$ and the derived μ (see below), one can exactly reconstruct all chaotic bits beyond the position $(i\cdot B')$ and then restore all plain-bytes starting from the first group located after the plain-byte that is encrypted by the $(i\cdot B')$-th

*For example, when $B = 80$, $B' = 16$, one has $B - \frac{B'}{2} = 72$, which can be considered to be sufficiently large.

chaotic bit. Apparently, for the original HDSP with $B = B' = 16$, such an idea of reducing the complexity is also feasible (but somewhat meaningless). Of course, using this method, the guessing complexity has a lower bound $O\left(2^{B-B'}\right)$, which is the complexity of exhaustively guessing the $(B - B')$ bits that do not occur in the chaotic bit sequence.

b) Breaking the control parameter μ

Given 32 consequent chaotic bits $b(16i + 0) \sim b(16i + 31)$, two consecutive chaotic states can be determined: $x(i) = 0.b(16i) \cdots b(16i + 15)$ and $x(i + 1) = 0.b(16i + 16) \cdots b(16i + 31)$, and then an estimated value of the secret sub-key μ can be derived as §3.3.3.2. All correctly-restored chaotic bits after $b(16i+31)$ can be used to verify whether or not a searched value of μ is right.

As discussed above, generally there are undetermined bits in $b(16i + 0) \sim b(16i + 31)$. The average number of such bits is $\frac{1}{4} \cdot 32 = 8$, and the maximal number is 16. This means that one has to exhaustively search all values of the undetermined bits to calculate a number of different values of $\widetilde{\mu}$. One can see that the complexity will be about $O\left(2^8\right)$ in average and be $O\left(2^{16}\right)$ in the worst condition. To further reduce the searching complexity, one can try to find two consecutive chaotic states that contain less than $m < 8$ undetermined bits*. The occurrence probability of such an event is $p(m) = \sum_{i=0}^{m} \binom{16}{i} \left(\frac{1}{2}\right)^{16}$, and the average position of its first occurrence is $1/p(m) = \frac{2^{16}}{\sum_{i=0}^{m} \binom{16}{i}}$. In Table 5.1, the values of $p(m)$ and $1/p(m)$ for $m = 0 \sim 8$ are shown. If the plaintext does not satisfy a uniform distribution, the probability of $d_k^*(i) \neq d_{k+4}^*(i)$ may not be $\frac{1}{2}$, and then the probability $p(m)$ may be different from the theoretical value (compare Tables 1 and 3).

Table 5.1: The occurrence probability of 32 consequent bits with less than $m \leq 8$ undetermined bits, and the average position of the first occurrence of such bits in the chaotic bit sequence.

m	0	1	2	3	4	5	6	7	8
$p(m) \approx$	0.0000153	0.000259	0.00209	0.0106	0.0384	0.105	0.227	0.402	0.598
$\lceil 1/p(m) \rceil$	65536	3856	479	95	27	10	5	3	2

*Similar to the condition of $x(0)$, when $B > 16$ bits are used to represent chaotic states, it will become very useful to do so.

§5.3.3 The Chosen-Plaintext Attack

In the above-mentioned known-plaintext attack, the undetermined even bits in the chaotic sequence are due to the following fact: for many (25% of all) plain-bytes, $d_0(i) = d_4(i)$ or $d_2(i) = d_6(i)$, i.e., $d_0^*(i) = d_4^*(i)$ or $d_2^*(i) = d_6^*(i)$ holds for the corresponding intermediate bytes. In the chosen-plaintext attack, one can create a plaintext $g = \{g(i)\}_{i=0}^{N-1}$ as follows: $\forall\ i = 0 \sim (N-1)$, $d_0(i) \neq d_4(i)$ and $d_2(i) \neq d_6(i)$. With such a plaintext and its ciphertext g', all chaotic bits can be uniquely determined. Thus, the secret sub-key $x(0) = 0.b(0)\cdots b(15)$ will be accurate, and another sub-key μ can be exactly derived from any two consecutive chaotic states $x(i)$ and $x(i+1)$ as §3.3.3.2. As a result, one can see that HDSP is not secure at all against the chosen-plaintext attack.

§5.4 Experiments

In this section, some experiments are shown to support the theoretical analysis on the known-plaintext attack*. The parameters used in the experiments are $N = 65536$, $S_f = 32$, $S_g = 16$, and the secret key is $x(0) = \frac{16326}{2^{16}} \approx 0.249$, $\mu = \frac{259752}{2^{16}} \approx 3.96$. Note that the values of $x(0)$ and μ are both randomly generated via the standard `rand()` function, not specially chosen to maximize the proposed attacks. The eight involved plaintexts are shown in Fig. 5.1, from top to bottom, denoted by $g_0 \sim g_7$, respectively, where the first seven ones are candidates of known plaintexts and the last one is used to show the breaking performance. The corresponding ciphertexts of the eight plaintexts are respectively denoted by $g_0' \sim g_7'$, which are not shown here since all of them are like meaningless noisy signals.

To simplify the experiments, we remove the speech CODEC in the whole HDSP-based VoIP system and directly use the uncompressed raw data as the plaintext. Such a simplification will not make any influence on the breaking of the secret key from the recovered chaotic bit sequence. For the known-plaintext attack, when the partially-recovered chaotic bit sequence is directly used to decrypt a ciphertext, the existence of speech CODEC may enlarge the recovery errors in the decoded voice signal. If such an enlargement is so serious that the intelligibility of the recovered voice signal is damaged, one can turn to derive the secret key, which always works as a perfect tool to break HDSP.

*The chosen-plaintext attack is omitted here, since it is just a special case of the known-plaintext attack.

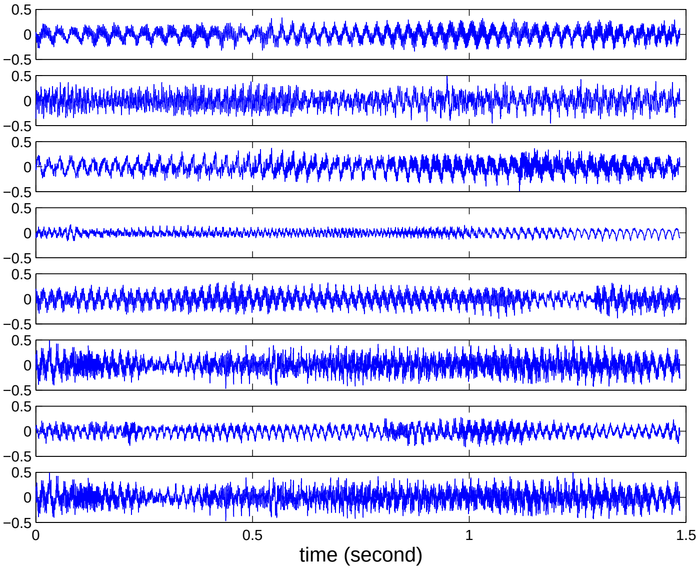


Figure 5.1: The eight plaintexts used in the experiments: $g_0 \sim g_7$ (from top to bottom).

§5.4.1 Partially Reconstructing the Chaotic Bit Sequence

Following the breaking procedure discussed in the last section, when plaintexts $g_0, \cdots, g_{n-1}$ and their ciphertexts $g'_0, \cdots, g'_{n-1}$ are known (for $n = 1 \sim 7$), we test the partial reconstruction of the chaotic bit sequence and the breaking performance when directly using it to decrypt the ciphertext g'_7. The percentage of undetermined chaotic bits for different values of n is shown in Table 5.2, from which one can see that the percentage of the undetermined bits and and the percentage of the undetermined bytes are both close to the theoretical expectations: $\frac{1}{2^{n+1}}$, and $1 - P'(n) = 1 - \left(1 - \frac{1}{2^{n+2}}\right)^2$, respectively.

Table 5.2: The percentage of the undetermined bits in the partially reconstructed chaotic bit sequence, Per_1, and the percentage of plain-bytes of g_7 that are not correctly decrypted, Per_2, when $n = 1 \sim 7$ plaintexts are known.

n	1	2	3	4	5	6	7
Per_1	26.4%	13.8%	7.23%	4.23%	2.28%	1.17%	0.640%
$\frac{1}{2^{n+1}} \approx$	25.0%	12.5%	6.25%	3.13%	1.56%	0.781%	0.391%
Per_2	24.5%	13.1%	6.90%	3.92%	2.05%	1.06%	0.591%
$1 - P'(n) = 1 - \left(1 - \frac{1}{2^{n+2}}\right)^2 \approx$	23.4%	12.1%	6.15%	3.10%	1.55%	0.780%	0.390%

By randomly assigning values to all the undetermined bits, the partially-reconstructed bit sequence is used to decrypt the ciphertext g_7' so as to get an estimation of the plaintext g_7. The decrypted results with respect to different values of n are given in Fig. 5.2. The decryption errors between the recovered plaintexts and the original plaintext g_7 are shown in Fig. 5.3, and the percentage of the decryption errors are listed in the last row of Table 5.2. Although the recovery error when $n = 1$ looks rather large, the recovered plaintext is still recognizable by human ears. The reason is that almost all frequency information remains in the recovered plaintext. For a comparison of the power energy spectrum of the original plaintext g_7 and those of the seven recovered plaintexts when $n = 1 \sim 7$, see Fig. 5.4. It is obvious that all important frequency peaks remain in the spectra of all the seven recovered plaintexts (but with larger amplitudes). As a result, even under the condition that the secret key is not derived, one known plaintext is still enough for recovering an intelligible version of the secret voice information. In addition, with a good noise reduction algorithm, the audio quality of the decrypted signal can be further enhanced.

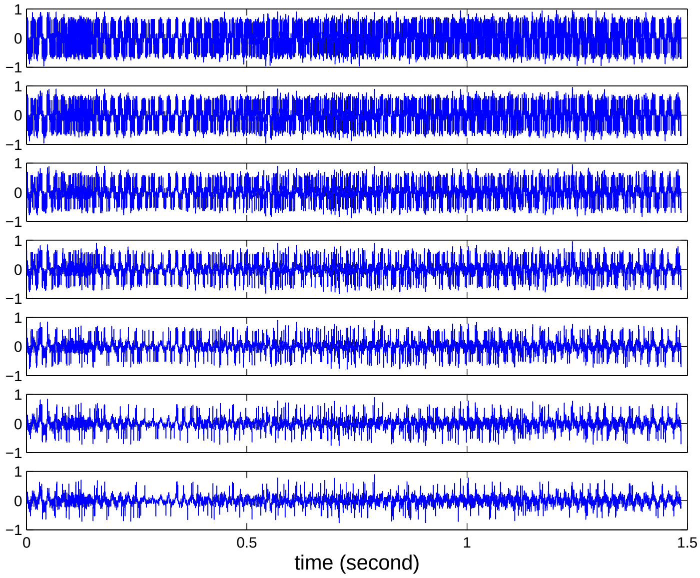


Figure 5.2: The decrypted plaintexts of g_7' with the partially-reconstructed chaotic bit sequence when $n = 1 \sim 7$ (from top to bottom) plaintexts are known.

§5.4.2 Breaking the Secret Key

As analyzed above, generally it is possible to derive the secret key $x(0)$ (or its equivalent $x(i)$) and μ from the partially-reconstructed chaotic bit sequence. To do so, one needs to find 32 consequent chaotic bits in which less than m bits are

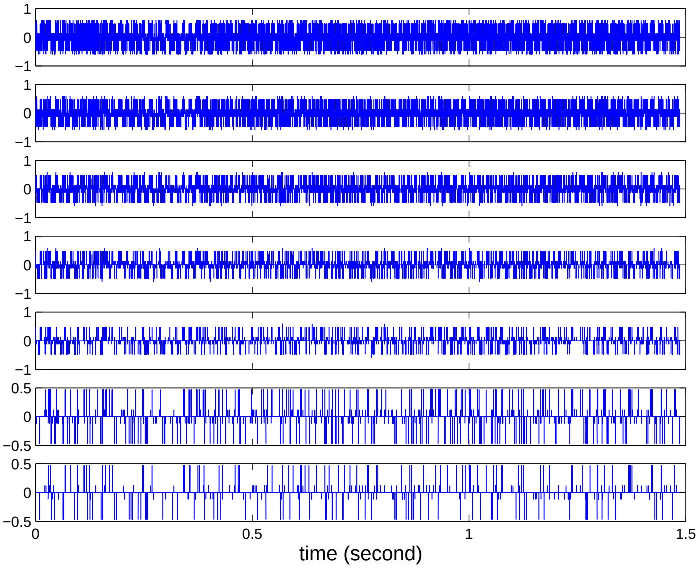


Figure 5.3: The decryption errors between the recovered plaintexts and the original plaintext g_7 when $n = 1 \sim 7$ (from top to bottom) plaintexts are known.

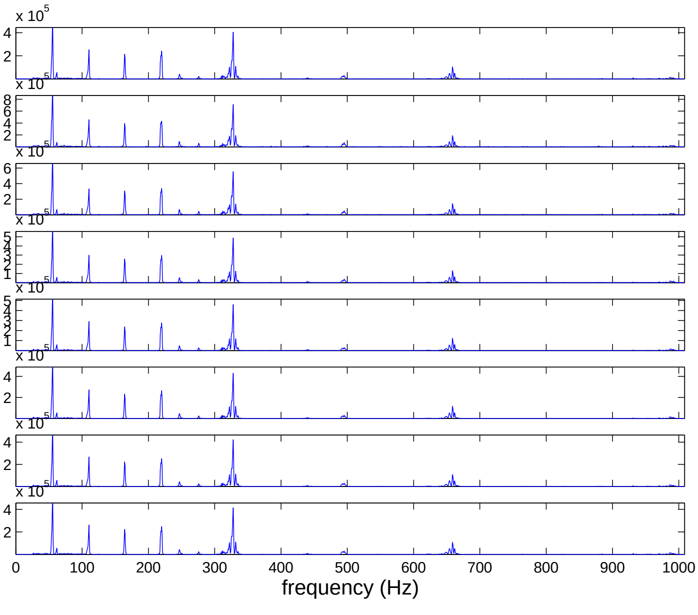


Figure 5.4: The power energy spectrum of the original plaintext g_7 (the 1st line) and the spectra of the recovered plaintexts when $n = 1 \sim 7$ (from the 2nd line to the last) plaintexts are known.

undetermined. When only the plaintext g_0 is known, for different values of m, the numbers of all 32-bit groups satisfying the above requirement are listed in Table

5.3. One can see that even for $m = 0$ there are enough positions to derive the secret key. By taking the first occurrence of the 32 consequent bits to successfully derive a chaotic state $x(i)$ and the value of μ, one can decrypt any ciphertext from *the first group after* $x(i)$, which begins at the position $\left\lceil \frac{4 \cdot i}{S_g \cdot S_f} \right\rceil \cdot (S_g \cdot S_f)$. When g_0 is known, it was found that the first 32 consequent bits satisfying $m = 0$ occurs at $x(163)$, which is used to derive μ and then to decrypt g'_7. The decrypted plaintext and the recovery error are shown in Fig. 5.5. It can be seen that all plain-bytes from $g\left(\left\lceil \frac{4 \cdot 163}{16 \cdot 32} \right\rceil \cdot (16 \cdot 32)\right) = g(1024)$ are exactly recovered.

Table 5.3: The number of 32 continuous chaotic bits that have $m \leq 8$ undetermined bits, and the occurrence frequency.

m	0	1	2	3	4	5	6	7	8
$N(m)$	72	185	464	888	1574	2537	4106	6113	8715
$Freq[N(m)] \approx$	0.0045	0.0113	0.0283	0.0542	0.0961	0.155	0.251	0.373	0.532

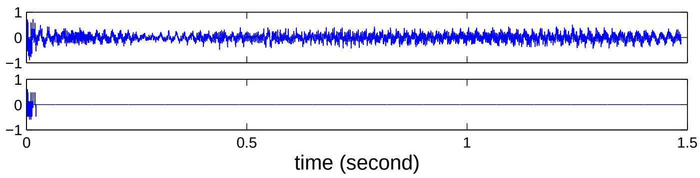


Figure 5.5: The decrypted plaintext of g'_7 and the recovery error with the derived key.

§5.5 Improving HDSP

As shown in the last section, the insecurity of HDSP against known/chosen-plaintext attacks is attributed to the properties proved in sub-subsection §5.3.2.1. In the first stage of breaking the inter-frame interleaving, Properties 1a, 2a, 2b and 3 are involved; in the second stage of breaking the intra-frame encryption, Properties 1b, 2a and 2b are involved. Also, the second breaking stage relies on the first one, since the intermediate signal g^* will not be available if the first stage does not work. This implies that the insecure properties play different roles in the known/chosen-plaintext attacks, which can be shown as follows:

$$g \xrightarrow[\text{Property 5.2a, 5.2b}]{\text{Property 5.1a, 5.3}} \left.\begin{matrix} g^* \\ \\ g' \end{matrix}\right\} \xrightarrow[\text{5.2a, and 5.2b}]{\text{Property 5.1b,}} \{b(i)\} \to \left\{\begin{matrix} x(0) \\ x(i), x(i+1) \to \mu \end{matrix}\right. \tag{5.10}$$

One can see that Property 5.1a is the basis of the whole attack, since Property 5.2a, 5.2b and 5.3 are just used to detect wrong permutation vectors. As a hint, if HDSP is modified to eliminate Property 5.1a, then the security against known/chosen-plaintext attack will be enhanced. However, if S_g is too small, it may be possible for an attacker to exhaustively search all $(S_g!)$ possible permutation vectors so as to pass the first breaking stage. Therefore, from the cryptographical point of view, all insecure properties of HDSP should be avoided to provide a high level of security. In addition, to resist other potential attacks, all known security defects should also be removed. In the following, we discuss how to amend the original HDSP scheme for better encryption.

- *Property 5.1a* is caused by the fact that no even bits are masked pseudo-randomly by the secret chaotic bits. Also, *Property 5.1b* is related to this defect. To fix this defect, we suggest *masking all bits, including odd bits and even bits.*

- *Properties 5.1b, 5.2a and 5.3* are caused by the incapability of swapping operations for encryption purpose, and *Properties 2a and 3* are also partially caused by the invertibility of the XOR operation. These properties can be destroyed by *changing the bit swapping and masking operations to other more complicated ones*, for example, inserting an extra masking operation before the two bits are swapped, or changing bit swapping operation to a different bit function.

- *Property 5.2b* is caused by two flaws: a) the equality of the masking operation and the swapping operation $Swap_w(a, b)$ when $a \neq b$ (see Eq. (5.4)); b) the reuse of all odd bits: for $k = 1, 3$, $\forall\, i = 1 \sim N-1$, each bit $b(4i+k)$ are used thrice – once for the swapping operation of $d_k^*(i)$, and twice for the masking operations of $d_{k+4}^*(i-1)$ and $d_k^*(i)$; similarly, when $i = 0$, the bit $b(k)$ is used twice for the swapping and masking operations of $d_k^*(0)$. The above two flaws disable the encryption for some bits, and make the two chaotic bit sequences $\{b(4i+1)\}_{i=0}^N$ and $\{b(4i+3)\}_{i=0}^N$ totally correlated. The first flaw can be fixed with the same method for eliminating Property 5.2a, and the second one can be removed by *avoiding any reuse of chaotic bits*, which means that at least $4 + 8 = 12$ chaotic bits are required for the encryption of each plain-byte (if swapping and masking operations are not replaced by other functions).

§5.6 Conclusion

In this chapter, the security of a recently-proposed encryption technique for VoIP, called HDSP [63], has been studied in detail. It is pointed out that HDSP cannot resist known/chosen-plaintext attacks, and that only one known/chosen plaintext is enough to break the secret key. It has also been found that the security of HDSP against the brute-force attack is very weak even for PC-s. Both theoretical and experimental analyses have been given to support the feasibility of the proposed attacks. As a conclusion, HDSP is not suggested in sensitive applications, particularly, if the secret key may be reused to encrypt more than one plaintext (which is the scenario where known/chosen-plaintext attacks work very well [1]). Finally, some remedies for enhancing HDSP have been suggested.

Chapter 6
Conclusion and Remarks on Future Research

§6.1 Lessons from The Above Cryptanalyses

From cryptanalyses of the above eight multimedia encryption schemes, some principles can be obtained for the design of good multimedia encryption schemes. Although the security of theses schemes against the known/chosen-plaintext attack is very weak of different degree, they are still useful as typical carelessly-designed examples to show what one should do and what one should not do.

Principle 1: Security against the known/chosen-plaintext attacks should be provided. As surveyed in [84], besides these eight schemes, many other multimedia encryption schemes are also insecure against the known/chosen-plaintext attack. However, without the capability against the known/chosen-plaintext attacks, it will be insecure to repeatedly use the same secret key to encrypt multiple multimedia files. When the cryptosystems are used to encrypt multimedia streams transmitted over networks, this problem can be relaxed due to the use of time-variant session keys [1]. Considering that most multimedia encryption systems are proposed to encrypt local multimedia files, the security against the known/chosen-plaintext attacks is generally required.

Principle 2: Do not use invertible encryption function. Rewrite the encryption function of a symmetric cipher as $C = E(P, K)$. The function $E(\cdot, \cdot)$ is said to be invertible, if K can be derived from C and P with its inverse function $E^{-1}(\cdot, \cdot)$, i.e., $K = E^{-1}(P, C)$. Most modern ciphers employs a mixture of operations defined in different groups to make the encryption function non-invertible.

In RCES/MES/DSEA/HDSP and the two neural network based schemes, one main encryption function is XOR, which is an invertible operation since $P \oplus K = C \Rightarrow K = P \oplus C$. It is the essential reason why information about secret key and/or plaintext can be got by different methods. Similarly, the invertibility of other basic operations, permutation, bit recirculation, is the reason for the success of breaking HCIE,TDCEA,MES. The whole security of one scheme cannot be obtained by combining much more insecure basic operations since it can be broken with the *Divide and Conquer* strategy.

To enhance the security of the eight schemes, the basic operations should be replaced with some key-dependent invertible functions and/or more complex ones. References [9, 16, 18, 21] suggest some typical image ciphers that use such an idea to ensure the security against the known/chosen-plaintext attacks.

Principle 3: The correlation information within/between the plaintexts should be sufficiently reduced. As shown in §3.3, the high correlation information between adjacent pixels is an important reason of the good performances for the known/chosen-plaintext attacks at RCES. In DSEA/HDSP, the main reason for breaking of secret key and encryption procedures is the strong correlation between the plaintexts (or some intermediate variables *observable* for attackers). Strong correlation between different secret parameters should be avoided also.

In fact, there exists a large amount of correlation information within multimedia data, even between pixels whose distances are large, such as pixels in a smooth area. To provide sufficient security against attacks, the correlation information within the multimedia data should be sufficiently concealed. A typical method to conceal the correlation information is to carry out complex long-distance permutation operations [9, 16, 18, 21]. Note that the long-distance permutations are not necessary conditions, but sufficient ones, since any secure text cipher can also provide enough security for multimedia data.

Principle 4: Any non-uniformity existing in the cipher-images should be avoided. From a cryptographer's point of view, any non-uniformity is not welcome due to the risk of causing statistics-based attacks, such as the well-known differential attacks [1]. So, it should be carefully checked whether or not there exists any non-uniformity in the ciphertexts.

The essential reason for the insecurity of RCES against the known/chosen-plaintext attacks can also be ascribed to the non-uniformity of the distribution of $f(l) \oplus f'(l)$ over $\{0, \cdots, 255\}$:

- for any unswapped pixel, $Prob[f(l) \oplus f'(l) = Seed(l)] = 1$, i.e., the distribution is one with the most non-uniformity;

- for any swapped pixel, the distribution of $f(l) \oplus f'(l)$ has the same non-uniformity level as the one of $f(l) \oplus f(l+1)$ (see the distribution of $f_{\text{Peppers}}^{(\oplus)}$ shown in Fig. 3.5).

This also suggests that all pixels should be permuted. Actually, in the second known-plaintext attack in §3.3.3, the feasibility of the quick-search algorithm in finding the two random seeds is benefited from the non-uniformity of the distribution of $\left\{\widetilde{f}_m(16k+j)\right\}_{j=0}^{15}$ over the discrete set $\{0, \cdots, 127\}$. If each $\widetilde{f}_m(16k+j)$ distributes uniformly over $\{0, \cdots, 127\}$, the exhaustive search algorithm will be practically impossible when the block size is changed to a sufficiently large value. Additionally, the breaking of secret key, L, of DSEA is also attributed to $g_0^*(n)/g_1^*(n)$ happen to be equal to $g(n)$ with different probability.

Principle 5: Never repeatedly use any (secret) intermediate variables generated in the encryption procedure. This problem exists in almost all the schemes proposed by Yen's group, where the same bit is reused to control different operation of neighboring pixels. Ordinarily, the reuse will result in reducing the complexity of brute-force attack and make known/chosen-plaintext more feasible. However, it become extremely serious for HDSP since it make HDSP be disabled for some plain-bits and two secret bit sequences become totally correlated.

§6.2 Summary of this Thesis

Now let us summary this thesis. Our work described in this thesis mainly focuses on the cryptanalysis of some encryption schemes protecting multimedia data, especially the capacity to withstand the known/chosen-plaintext attack. In the following we would like to sum up our achievements separately.

1. As for the security of permutation-only encryption schemes, this thesis answered two questions:
    - How many known/chosen plain-images are needed for an attacker to achieve a rather good breaking performance?
    - How much is the attack complexity when n plain-images are known or chosen?
2. The security of chaos-based image encryption scheme called RCES was analyzed and found that it can be broke with only one or two known/chosen-plaintext. Both theoretical and experimental analyses are given to show the performance of the suggested known/chosen-plaintext attacks. The cryptanalysis show us how the strong correlation of multimedia data may bring potential insecurity to multimedia encryption schemes.
3. Scheme MES is successfully broke with a differential chosen-plaintext attack, which tell us that the whole security of one encryption scheme can not be obtained by combining much more insecure basic encryption operations.
4. The security of DSEA is analyzed. It is found that the algorithm is even not secure enough against ciphertext-only attack with only one ciphertext. All secret information can be recovered by one known/chosen plaintext and the corresponding ciphertext.
5. The security of TDCEA is analyzed comprehensively. It is demonstrated that some essential security defects exist in TDCEA. The scheme can be

broke with two different known-plaintext attack methods and their chosen-plaintext attack counterparts.

6. The properties of two neural network based encryption schemes are analyzed in detail. It is found that one can be easily broken by known/chosen-plaintext attacks and the other can be broken by a chosen-plaintext attack. From the cryptanalyses we can see that the security of a encryption scheme depends on its good structure instead of so-called complex theory.

7. Some insecure properties of the HDSP scheme are analyzed, and then used as the basis for the proposed known/chosen-plaintext attacks. The cryptanalysis of the scheme tell us that the HDSP scheme serves a 'good' counterexample to show the importance of prohibiting any reuse of secret bits.

8. Some principles for designing secure multimedia encryption schemes are drawn from common essential security defects of the eight schemes, which are analyzed in the previous four chapters.

§6.3 Future Research

During the research for this thesis a number of interesting problems were encountered, which are still not solved. We summarize briefly what we plan to do in the near future as follows.

- Continue to study the security of some other proposed multimedia encryption schemes which seem to be vulnerable to some attacks. To do these, some related theories, such as chaos, number theory etc., need to be understood;

- Extend the attack methods that we have used to investigate the security and robustness of some chaos-based secure communication schemes under corresponding attack scenarios;

- Based on the security knowledge we have accumulated, study the security of some Hash functions and authentication schemes against proposed attack methods;

- Study deep into the general insecurity properties of multimedia encryption schemes, and try to propose some corresponding countermeasures, furthermore, design some new schemes that have good security properties and can meet the real application requirements at the same time.

Acknowledgement

Here, I would like to express my deep gratitude to all people who made this thesis possible.

Firstly, I wish to thank my supervisor, Prof. Guangchang Dong, for his invaluable supervision and guidance during my 3-year master study. I would also like to thank Prof. Gang Chen, for his enthusiastic care and encouragement.

Deeply gratitude is given to Dr. Shujun Li at City University of Hong Kong. Acquaintance with him is originated from my careful reading of his paper [81] in Dec., 2003. Since then, we begin to coauthor eight papers [88–95] mainly by communication via email. Up to today, the number of our come-and-go emails have reached 1074! The author benefits much from his strong analysis capability and precise research attitude. This thesis would not have been possible without ideas, suggestions, criticisms, and friendship from him.

Thank Mr. Xiaoyu Zhao at Ningbo University for his valuable help on the author's initial research. Thank Dr. Dan Zhang at College of Software Technology, Zhejiang University for his generous support and zealous encouragement.

Thank Prof. Guanrong Chen at City University of Hong Kong for his invaluable guidance for most papers the author completed.

Thank Prof. Lei Hu at The State Key Laboratory of Information Security, Chinese Academy of Sciences, for his selfless help on publication of paper[90].

Thank Prof. Jiun-In Guo, Mr. Qing Zhou, Dr. Miao Ma, Mr. Xiaoyu Ruan, Prof. Wen-Whei Chang for their help on literature retrieval.

Thank Dr. Xinxiao Li at Software Engineering Center, Toshiba, Japan for that he made presentation for the author in PCM 2004.

Thank image processing group members, Mr. Ping Wei, Mr. Tao Kong, Ms. Xuehui Wei, Mr. Beijing Chen and others for their support and help.

I would like to thank my roommates, Mr. Yinli Liu, Mr. Tingfang Zhou and Mr. Wenhao Song, and my friends, Dr. Yaping Jing, Mr. Chao Xu, Ms. Min Huang, Ms. Ping Zhou, Ms. Yusong Cao, Mr. Mingjing Liu et al. for their help in different ways.

I also wish to thank my family for their support throughout the years.

Finally I thank the support of the National Natural Science Foundation, China, under grant no. 60202002.

May 19, 2005

Bibliography

[1] Bruce Schneier. *Applied Cryptography – Protocols, Algorithms, and Souce Code in C.* John Wiley & Sons, Inc., New York, second edition, 1996.

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

[3] Howard Chi Ho Cheng. Partial encryption for image and video communication. Master thesis, Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada, Fall 1998.

[4] Nikolaos G. Bourbakis and Chris Alexopoulos. Picture data encryption using SCAN patterns. *Pattern Recognition*, 25(6):567–581, 1992.

[5] Henry Key-Chang Chang and Jiang-Long Liu. A linear quadtree compression scheme for image encryption. *Signal Processing: Image Communication*, 10(4):279–290, 1997.

[6] N. Bourbakis and C. Alexopoulos. Picture data encryption using scan patterns. *Pattern Recognition*, 25(6):567–581, 1992.

[7] Chris Alexopoulos, Nikolaos G. Bourbakis, and N. Ioannou. Image encryption method using a class of fractals. *J. Electronic Imaging*, 4(3):251–259, 1995.

[8] Kuo-Liang Chung and Lung-Chun Chang. Large encrypting binary images with higher security. *Pattern Recognition Letters*, 19(5-6):461–468, 1998.

[9] Suchindran S. Maniccam and Nikolaos G. Bourbakis. Image and video encryption using SCAN patterns. *Pattern Recognition*, 37(4):725–737, 2004.

[10] Dongxu Qi, Jiancheng Zou, and Xiaoyou Han. A new class of scrambling transformation and its application in the image information covering. *Science in China - Series E (English Edition)*, 43(3):304–312, 2000.

[11] Kenji Yano and Kiyoshi Tanaka. Image encryption scheme based on a truncated Baker transformation. *IEICE Trans. Fundamentals*, E85-A(9):2025–2035, 2002.

[12] Xiao-Yu Zhao and Gang Chen. Ergodic matrix in image encryption. In *Proc. Second International Conference on Image and Graphics*, *Proc. SPIE*, vol. 4875, pages 394–401, 2002.

[13] Takeyuki Uehara, Reihaneh Safavi-Naini, and Philip Ogunbona. Securing wavelet compression with random permutations. In *Proc. IEEE Pacific-Rim Conference on Multimedia (IEEE-PCM'2000)*, pages 332–335, 2000.

[14] D. Van De Ville, W. Philips, R. Van de Walle, and I. Lemahieu. Image scrambling without bandwidth expansion. *IEEE Trans. Circuits and Systems for Video Technology*, 14(6):892–897, 2004.

[15] Rodolfo Zunino. Fractal circuit layout for spatial decorrelation of images. *Electronics Letters*, 34(20):1929–1930, 1998.

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

[17] Yaobin Mao, Guanrong Chen, and Charles K. Chui. A symmetric image encryption scheme based on 3D chaotic cat maps. *Chaos, Solitons & Fractals*, 21(3):749–761, 2004.

[18] Yaobin Mao, Guanrong Chen, and Shiguo Lian. A novel fast image encryption scheme based on 3D chaotic Baker maps. *Int. J. Bifurcation and Chaos*, 14(10):3613–3624, 2004.

[19] Chin-Chen Chang, Min-Shian Hwang, and Tung-Shou Chen. A new encryption algorithm for image cryptosystems. *J. Systems and Software*, 58(2):83–91, 2001.

[20] Philip P. Dang and Paul M. Chau. Image encryption for secure internet multimedia applications. *IEEE Trans. Consumer Eletronics*, 46(3):395–403, 2000.

[21] Josef Scharinger. Fast encryption of image data using chaotic Kolmogorov flows. *J. Electronic Imaging*, 7(2):318–325, 1998.

[22] Andreas Pommer and Andreas Uhl. Selective encryption of wavelet-packet encoded image data: Efficiency and security. *Multimedia Systems*, 9(3):279–287, 2003.

[23] Howard Cheng and Xiaobo Li. Partial encryption of compressed images and videos. *IEEE Trans. Signal Processing*, 48(8):2439–2451, 2000.

[24] Tsing Zhou, Xiaofeng Liao, and Yong Chen. A novel symmetric cryptography based on chaotic signal generator and a clipped neural network. In *Advances in Neural Networks - ISNN 2004: International Symposium on Neural Networks Proceedings, Part II*, *Lecture Notes in Computer Science* **3174**, pages 639–644, 2004.

[25] Shiguo Lian, Jinsheng Sun, and Zhiquan Wang. Perceptual cryptography on spiht compressed images or videos. In *Proc. IEEE Int. Conf. Multimedia & Expo (ICME'2004)*, 2004.

[26] Shiguo Lian, Guanrong Chen, Albert Cheung, and Zhiquan Wang. A chaotic-neural-network-based encryption algorithm for JPEG2000 encoded images. In *Advances in Neural Networks - ISNN 2004: International Symposium on Neural Networks Proceedings, Part II*, *Lecture Notes in Computer Science* **3174**, pages 627–632, 2004.

[27] Jui-Cheng Yen and Jiun-In Guo. A new hierarchical chaotic image encryption algorithm and its hardware architecture. In *Proc. 1998 Ninth VLSI DESIGN/CAD Symposium*, 1998.

[28] Jui-Cheng Yen and Jiun-In Guo. A new chaotic image encryption algorithm. In *Proc. (Taiwan) National Symposium on Telecommunications*, pages 358–362, 1998.

[29] Jui-Cheng Yen and Jiun-In Guo. A new MPEG/encryption system and its VLSI architecture. In *Proc. Int. Symposium on Communications*, pages 215–219, 1999.

[30] Jiun-In Guo, Jui-Cheng Yen, and Jen-Chieh Yeh. The design and realization of a new chaotic image encryption algorithm. In *Proc. 1999 International Symposium on Communications*, pages 210–214, 1999.

[31] Jui-Cheng Yen and Jiun-In Guo. A chaotic neural network for signal encryption/decryption and its VLSI architecture. In *Proc. 10th VLSI Design/CAD Symposium*, pages 319–322, 1999.

[32] Jui-Cheng Yen and Jiun-In Guo. A new image encryption algorithm and its VLSI architecture. In *Proc. IEEE Workshop on Signal Processing Systems*, pages 430–437, 1999.

[33] Jui-Cheng Yen and Jiun-In Guo. Efficient hierarchical chaotic image encryption algorithm and its VLSI realisation. *IEE Proc. – Vis. Image Signal Process.*, 147(2):167–175, 2000.

[34] Scott Su, Alvin Lin, and Jui-Cheng Yen. Design and realization of a new chaotic neural encryption/decryption network. In *Proc. IEEE Asia-Pacific Conference on Circuits and Systems*, pages 335–338, 2000.

[35] Jui-Cheng Yen and Jiun-In Guo. A new chaotic key-based design for image encryption and decryption. In *Proc. IEEE Int. Symposium on Circuits and Systems (ISCAS'2000)*, volume 4, pages 49–52, 2000.

[36] Jui-Cheng Yen and Jiun-In Guo. The design and realization of a chaotic neural signal security system. *Pattern Recognition and Image Analysis (Advances in Mathematical Theory and Applications)*, 12(1):70–79, 2002.

[37] Hun-Chen Chen, Jui-Cheng Yen, and Jiun-In Guo. Design of a new cryptography system. In *Proc. PCM'2002*, *Lecture Notes in Computer Science*, vol. 2532, pages 1041–1048. Springer-Verlag, Berlin, 2002.

[38] Jui-Cheng Yen and Jiun-In Guo. Design of a new signal security system. In *Proc. IEEE Int. Symposium on Circuits and Systems*, volume 4, pages 121–124, 2002.

[39] Hun-Chen Chen, Jiu-Cheng Yen, and Jiun-In Guo. Design of a new cryptography system. In *Advances in Multimedia Information Processing - PCM 2002: Third IEEE Pacific Rim Conference on Multimedia Proc.*, *Lecture Notes in Computer Science*, volume 2532, pages 1041–1048, 2002.

[40] Hun-Chen Chen, Jiu-Cheng Yen, and Jiun-In Guo. Design of a new cryptography system. In *Advances in Multimedia Information Processing - PCM 2002: Third IEEE Pacific Rim Conference on Multimedia Proceedings*, *Lecture Notes in Computer Science* **2532**, pages 1041–1048, 2002.

[41] Hun-Chen Chen and Jui-Cheng Yen. A new cryptography system and its VLSI realization. *J. Systems Architecture*, 49(7–9):355–367, 2003.

[42] Jui-Cheng Yen and Jiun-In Guo. The design and realization of a new domino signal security system. *Journal of the Chinese Institute of Electrical Engineering*, 10(1):69–76, 2003.

[43] Hun-Chen Chen, Jiun-In Guo, Lin-Chieh Huang, and Jui-Cheng Yen. Design and realization of a new signal security system for multimedia data transmission. *EURASIP J. Applied Signal Processing*, 2003(13):1291–1305, 2003.

[44] Jui-Cheng Yen, Hun-Chen Chen, and Shin-Shian Jou. A new cryptographic system and its VLSI implementation. In *Proc. IEEE Int. Symposium on Circuits and Systems*, volume 2, pages 221–224, 2004.

[45] Andreas Pommer. *Selective Encryption of Wavelet-Compressed Visual Data.* PhD thesis, Department of Scientific Computing, University of Salzburg, Austria, June 2003.

[46] Jiangtao Wen, Michael Severa, Wenjun Zeng, Maximilian H. Luttrell, and Weiyi Jin. A format-compliant configurable encryption framework for access control of video. *IEEE Trans. Circuits and Systems for Video Technology*, 12(6):545–557, 2002.

[47] Wenjun Zeng and Shawmin Lei. Efficient frequency domain selective scrambling of digital video. *IEEE Trans. Multimedia*, 5(1):118–129, 2003.

[48] Nikolaos G. Bourbakis and Apostolos Dollas. SCAN-based compression-encryption-hiding for video on demand. *IEEE Multimedia*, 10(3):79–87, 2003.

[49] Mohan S. Kankanhalli and Teo Tian Guan. Compressed-domain scrambler/descrambler for digital video. *IEEE Trans. Consumer Eletronics*, 48(2):356–365, 2002.

[50] Yossi Matias and Adi Shamir. A video scrambling technique based on space filling curve (extended abstract). In *Advances in Cryptology – Crypto'87*, *Lecture Notes in Computer Science*, volume 293, pages 398–417, 1987.

[51] Lei Tang. Methods for encrypting and decrypting MPEG video data efficiently. In *Proc. 4th ACM Int. Conference on Multimedia*, pages 219–229, 1996.

[52] Bharat Bhargava, Changgui Shi, and Sheng-Yih Wang. MPEG video encryption algorithms. *Multimedia Tools and Applications*, 24(1):57–79, 2004.

[53] Sang Uk Shin, Kyeong Seop Sim, and Kyung Hyune Rhee. A secrecy scheme for MPEG video data using the joint of compression and encryption. In *Information Security: Second Int. Workshop (ISW'99) Proc.*, *Lecture Notes in Computer Science*, volume 1729, pages 191–201, 1999.

[54] Franco Chiaraluce, Lorenzo Ciccarelli, Ennio Gambi, Paola Pierleoni, and Maurizio Reginelli. A new chaotic algorithm for video encryption. *IEEE Trans. Consumer Eletronics*, 48(4):838–844, 2002.

[55] Melih Pazarci and Vadi Dipçin. A MPEG2-transparent scrambling technique. *IEEE Trans. Consumer Eletronics*, 48(2):345–355, 2002.

[56] Dahua Xie and C.-C. Jay Kuo. An enhanced MHT encryption scheme for chosen plaintext attack. In *Internet Multimedia Management Systems IV*, *Proc. SPIE*, volume 5242, pages 175–183, 2003.

[57] Chung-Ping Wu and C.-C. Jay Kuo. Fast encryption methods for audiovisual data confidentiality. In *Multimedia Systems and Applications III*, *Proc. SPIE*, volume 4209, pages 284–295, 2001.

[58] Chung-Ping Wu and C.-C. Jay Kuo. Efficient multimedia encryption via entropy codec design. In *Security and Watermarking of Multimedia Contents III, Proc. SPIE*, volume 4314, pages 128–138, 2001.

[59] Xiaolin Wu and Peter W. Moo. Joint image/video compression and encryption via high-order conditional entropy coding of wavelet coefficients. In *Porc. IEEE Conference on Multimedia Computing and Systems (CMS'99)*, pages 908–912, 1999.

[60] Shiguo Lian, Xiaofan Wang, Jinsheng Sun, and Zhiquan Wang. Perceptual cryptography on wavelet-transform encoded videos. In *Proc. IEEE Int. Symp. on Intelligent Multimedia, Video and Speech Processing (ISIMP'2004)*, pages 57–60, 2004.

[61] Shiguo Lian, Jinsheng Sun, and Zhiquan Wang. Perceptual cryptography on JPEG2000 compressed images or videos. In *Proc. Int. Conf. Computer and Information Technology (CIT'2004)*, pages 78–83. IEEE Computer Society, 2004.

[62] Shiguo Lian, Jinsheng Sun, Zhongxin Li, and Zhiquan Wang. A fast mpeg4 video encryption scheme based on chaotic neural network. In *Neural Information Processing: ICONIP 2004 Proceedings*, *Lecture Notes in Computer Science* **3316**, pages 720–725, 2004.

[63] J.-I. Guo, J.-C. Yen, and H.-F. Pai. New voice over Internet protocol technique with hierarchical data security protection. *IEE Proc. – Vis. Image Signal Process.*, 149(4):237–243, 2002.

[64] J.-I. Guo, Chiun-Chau Lin, Min-Chih Tsai, and Sheng-Wei Lin. An efficient voice over Internet protocol technique combining the speech data encryption and G.729 error recovery. In *Proc. Int. Computer Symposium (ICS'2002)*, 2002.

[65] S. Sridharan, E. Dawson, and B. Goldburg. Speech encryption in the transform domain. *Electronics Letters*, 26(10):655–657, 1990.

[66] S. Sridharan, E. Dawson, and B. Goldburg. Fast Fourier transform based speech encryption system. *IEE Proc. – Communications, Speech and Vision*, 138(3):215–223, 1991.

[67] B. Goldburg, S. Sridharan, and E. Dawson. Design and cryptanalysis of transform-based analog speech scramblers. *IEEE Journal on Selected Areas in Communications*, 11(5):735–744, 1993.

[68] Michael Bertilsson, Ernest F. Brickell, and Ingemar Ingemarson. Cryptanalysis of video encryption based on space-filling curves. In *Advances in Cryptology – EuroCrypt'88, Lecture Notes in Computer Science*, volume 434, pages 403–411, 1989.

[69] Iskender Agi and Li Gong. An empirical study of secure MPEG video transmission. In *Proc. ISOC Symposium on Network and Distributed Systems Security (SNDSS'96)*, pages 137–144, 1996.

[70] Jinn-Ke Jan and Yuh-Min Tseng. On the security of image encryption method. *Information Processing Letters*, 60(5):261–265, 1996.

[71] Lintian Qiao and Klara Nahrstedt. Is MPEG encryption by using random list instead of ZigZag order secure? In *Proc. IEEE Int. Symposium on Consumer Electronics (ISCE'97)*, pages 226–229, 1997.

[72] Justin H. Dolske. Secure MPEG video: Techniques and pitfalls. available online at http://www.dolske.net/old/gradwork/cis788r08/, June 1997.

[73] Lintian Qiao and Klara Nahrsted. Comparison of MPEG encryption algorithms. *Computers & Graphics*, 22(4):437–448, 1998.

[74] Lintian Qiao. *Multimedia Security and Copyright Protection*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1998.

[75] Takeyuki Uehara and Reihaneh Safavi-Naini. Chosen DCT coefficients attack on MPEG encryption schemes. In *Proc. IEEE Pacific-Rim Conference on Multimedia (IEEE-PCM'2000)*, pages 316–319, 2000.

[76] Chin-Chen Chang and Tai-Xing Yu. Cryptanalysis of an encryption scheme for binary images. *Pattern Recognition Letters*, 23(14):1847–1852, 2002.

[77] Xiaoyu Zhao, Gang Chen, Dan Zhang, Xiaohong Wang, and Guangchang Dong. Decryption of pure-position permutation algorithms. *Journal of Zhejiang University SCIENCE*, 5(7):803–809, 2004.

[78] Xiliang Liu and Ahmet M. Eskicioglu. Selective encryption of multimedia content in distribution networks: Challenges and new directions. In *Proc. IASTED Int. Conference on Communications, Internet and Information Technology (CIIT'2003)*, 2003.

[79] Tom D. Lookabaugh, Douglas C. Sicker, David M. Keaton, Wang Y. Guo, and Indrani Vedula. Security analysis of selectively encrypted MPEG-2 streams.

In *Multimedia Systems and Applications VI*, *Proc. SPIE*, volume 5241, pages 10–21, 2003.

[80] Shujun Li and Xuan Zheng. Cryptanalysis of a chaotic image encryption method. In *Proc. IEEE Int. Symposium on Circuits and Systems*, volume II, pages 708–711, 2002.

[81] Shujun Li and Xuan Zheng. On the security of an image encryption method. In *Proc. IEEE Int. Conference on Image Processing (ICIP'2002)*, volume 2, pages 925–928, 2002.

[82] Hao Bai-Lin. *Starting with Parabolas: An Introduction to Chaotic Dynamics*. Shanghai Scientific and Technological Education Publishing House, Shanghai, China, 1993. (In Chinese).

[83] Shujun Li. *Analyses and New Designs of Digital Chaotic Ciphers*. PhD thesis, School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China, June 2003. Available online at http://www.hooklee.com/pub.html (in both Chinese and English).

[84] Shujun Li, Guanrong Chen, and Xuan Zheng. Chaos-based encryption for digital images and videos. In Borko Furht and Darko Kirovski, editors, *Multimedia Security Handbook*, chapter 4. CRC Press, LLC, 2004. preprint is available at http://www.hooklee.com/pub.html.

[85] Tohru Kohda and Kazuyuki Aihara. Chaos in discrete systems and diagnosis of experimental chaos. *Trans. IEICE*, E 73(6):772–783, 1990.

[86] The editorial committee of Modern Applied Mathematics Handbooks. *The Handbook of Probability Theory and Stochastic Process*, volume 3 of *Modern Applied Mathematics Handbooks*. Tsinghua University Press, Beijing, China, July 2000. (In Chinese).

[87] Vinodkrishnan Kulathumani. Voice over IP: Products, services and issues. available online at http://www.cse.ohio-state.edu/~jain/cis788-99/voip_products/index.html, November 23, 1999.

[88] Chengqing Li, Shujun Li, Dan Zhang, and Guanrong Chen. Cryptanalysis of a chaotic neural network based multimedia encryption scheme. In *Advances in Multimedia Information Processing - PCM 2004 Proceedings, Part III*, *Lecture Notes in Computer Science* **3333**, pages 418–425. Springer-Verlag, 2004.

[89] Chengqing Li, Xinxiao Li, Shujun Li, and Guanrong Chen. Cryptanalysis of a multistage encryption system. In *Proceedings of 2005 IEEE Int. Symposium on Circuits and Systems*, pages 880–883, 2005.

[90] Chengqing Li, Shujun Li, Guanrong Chen, Gang Chen, and Lei Hu. Cryptanalysis of a new signal security system for multimedia data transmission. *EURASIP J. on Applied Signal Processing*, 2005(8):1–12, 2005.

[91] Chengqing Li, Shujun Li, Dan Zhang, and Guanrong Chen. Chosen-plaintext cryptanalysis of a clipped-neural-network-based chaotic cipher. In *Advances in Neural Networks - ISNN 2005 Proceedings, Part II*, *Lecture Notes in Computer Science* **3497**, pages 630–636. Springer-Verlag, 2005.

[92] Shujun Li, Chengqing Li, Guanrong Chen, and Xuanqin Mou. Cryptanalysis of the RCES/RSES image encryption scheme. Cryptology ePrint Archive: Report 2004/376, available online at http://eprint.iacr.org/2004/376, 2004.

[93] Shujun Li, Chengqing Li, Guanrong Chen, Dan Zhang, and Nikolaos G. Bourbakis. A general cryptanalysis of permutation-only multimedia encryption algorithms. Cryptology ePrint Archive: Report 2004/374, available online at http://eprint.iacr.org/2004/374, 2004.

[94] Chengqing Li, Shujun Li, Dan Zhang, and Guanrong Chen. Cryptanalysis of a data security protection scheme for VoIP. submitted to *IEE Proceedings-Vis. Image Signal Process*, 2004.

[95] Chengqing Li, Shujun Li, and Der-Chyuan Lou. On the security of the Yen-Guo's domino signal encryption algorithm (DSEA). accepted by *Elsevier Journal of Systems and Software*, 2005.

Publications Related to This Thesis

[1] Chengqing Li, Shujun Li, Dan Zhang, and Guanrong Chen. Cryptanalysis of a chaotic neural network based multimedia encryption scheme. *Lecture Notes in Computer Science* **3333**, pages 418–425. Springer-Verlag, 2004.

[2] Chengqing Li, Xinxiao Li, Shujun Li, and Guanrong Chen. Cryptanalysis of a multistage encryption system. In *Proceedings of 2005 IEEE Int. Symposium on Circuits and Systems*, pages 880–883, 2005.

[3] Chengqing Li, Shujun Li, Guanrong Chen, Gang Chen, and Lei Hu. Cryptanalysis of a new signal security system for multimedia data transmission. *EURASIP J. on Applied Signal Processing*, 2005(8):1277–1288, 2005.

[4] Chengqing Li, Shujun Li, Dan Zhang, and Guanrong Chen. Chosen-plaintext cryptanalysis of a clipped-neural-network-based chaotic cipher. *Lecture Notes in Computer Science* **3497**, pages 630–636. Springer-Verlag, 2005.

[5] Chengqing Li, Shujun Li, and Der-Chyuan Lou. On the security of the Yen-Guo's domino signal encryption algorithm (DSEA). *Elsevier Journal of Systems and Software*, 79(2):253–258, 2006.

[6] Shujun Li, Chengqing Li, Guanrong Chen, and Xuanqin Mou. Cryptanalysis of the RCES/RSES image encryption scheme. Cryptology ePrint Archive: Report 2004/376, available online at http://eprint.iacr.org/2004/376, 2004.

[7] Shujun Li, Chengqing Li, Guanrong Chen, Dan Zhang, and Nikolaos G. Bourbakis. A general cryptanalysis of permutation-only multimedia encryption algorithms. Cryptology ePrint Archive: Report 2004/374, available online at http://eprint.iacr.org/2004/374, 2004.

[8] Chengqing Li, Shujun Li, Dan Zhang, and Guanrong Chen. Cryptanalysis of a data security protection scheme for VoIP. *IEE Proceedings-Vis. Image Signal Process*, 153(1):1–10, 2006.