Chapter 1

Introduction

The idea of *linear cryptanalysis* was first introduced by Matsui and Yamagishi [7] in 1992 in an attack on FEAL [8]. The techniques used in this attack were refined by Matsui and used on DES [9] in a theoretical attack on the full 16-round DES requiring 2^{47} known plaintext/ciphertext pairs [6]. He then performed an experiment to break the full 16-round DES with high success probability using 2^{43} known plaintext/ciphertext pairs [10]. After that Kaliski and Robshaw investigate the use of *multiple linear approximations* to improve the success rate of linear cryptanalysis and to reduce the data complexity [11]. In 1994, Nyberg demonstrated that the success rate of Matsui's Algorithm 2 is underestimated by using the *approximate linear hull* [12].

In this thesis, we focus on the estimation of the number of test pairs, and the attack strategies on *Substitution-Permutation Networks* (SPN). We try to estimate that how many test pairs do we need, such that the success rate of the attack is sufficiently high. And we list some strategies of linear cryptanalysis to attack an SPN structure.

This thesis is organized as follows. In Chapter 2, we survey the background of cryptology, the block ciphers, and the introduction of the attacks. In Chapter 3, we present linear cryptanalysis on the SPN structure. In Chapter 4, we show the steps of the attack. We estimate the number of test pairs, and list some attack strategies. In Chapter 5, we use these strategies to attack a simple SPN structure. In Chapter 6 we summarize the results in this thesis.

Chapter 2

Background

2.1 Cryptography

Cryptography is the study of mathematical techniques related to aspects of *information security services* [1, 13, 2]. A lot of information security services can be identified, but the following four objects are considered to be foundational:

1. *Secrecy* (also called *privacy* or *confidentiality*): Ensures that the information in a computer system and transmitted information are accessible only for reading by authorized parties. An attacker may be able to view certain data (e.g., via a wiretap), but can not extract meaningful information from it.

2. *Data Integrity*: Ensures that only authorized parties are able to modity computer system assets and transmit information. Modification includes insertion, deletion, and substitution.

3. *Authentication*: Ensures that the origin of a message or electronic document is correctly indentified, with an assurance that the identity is not false. Authentication is usually subdivided into two major classes: *entity authentication* and *data origin authentication*. Data origin authentication implicitly provides data integrity (for if a message is modified, the source has changed).

4. *Non-repudiation*: Requires that neither the sender nor the receiver of a message be able to deny the transmission. Non-repudiation is important in situations in which disputes may arise over prior transactions. Protocols that ensure non-repudiation typically require the involvement of a trusted third party.

The term *cryptanalysis* refers to techniques used to break cryptographic techniques. *Cryptology* is the field of study that encompasses both cryptography and cryptanalysis.

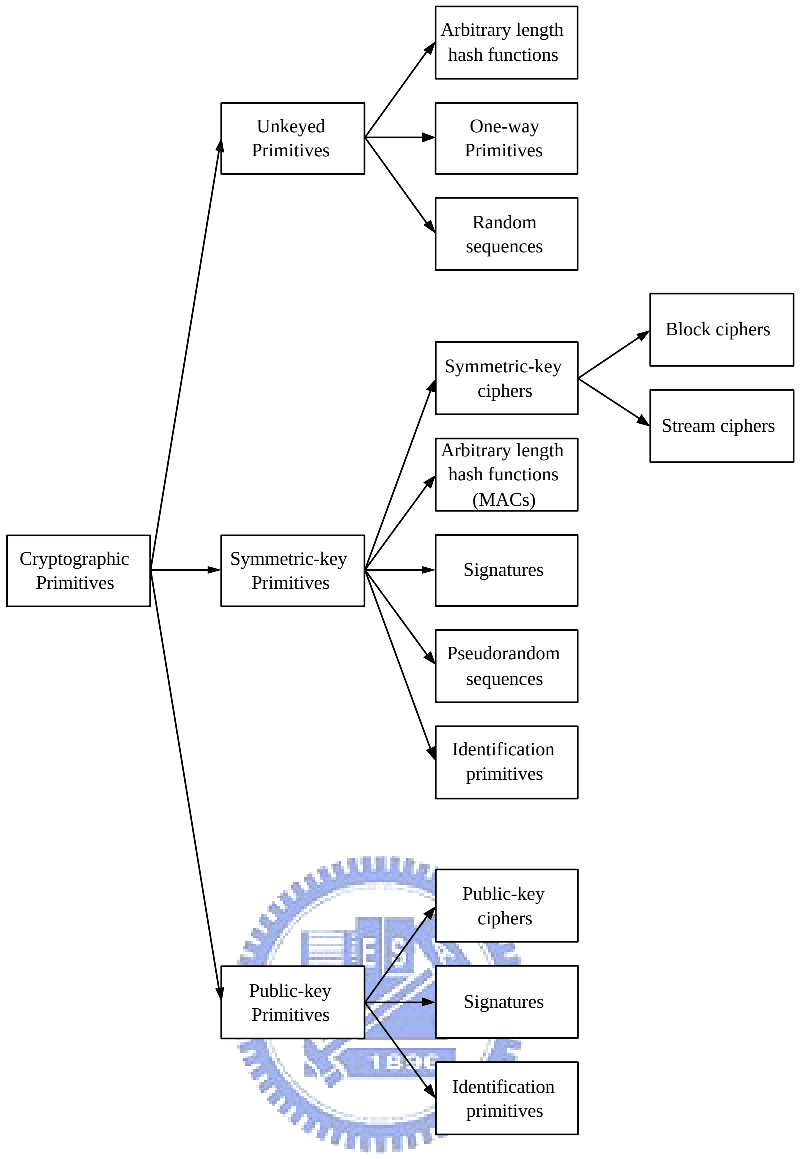


Figure 2.1: A taxonomy of cryptographic primitives (from [1, 2])

2.2 Cryptographic Primitives

The buliding blocks of the cryptographic techniques used to provide information security services are called *cryptographic primitives* (*tools*). Most primitives are functions whose inputs and outputs are elements of certain spaces of finite-length binary strings. Examples of primitives include encryption schemes, hash functions, and digital signature schemes. Figure 2.1 gives a schematic listing of the primitives considered and how they relate.

In the following sections we will introduce ome primitives in Figure 2.1. The remaining primitives are outside the scope of this thesis.

2.3 Ciphers

Figure 2.2 shows the basic operation of a cipher. Suppose the sender has a large message to send to the receiver. The sender divides this message into smaller pieces called *plain-*

texts, which are fixed length. Each plaintext (P) is input into an *encryption algorithm*, which takes an *encryption key*, K_E, as a parameter; the resulting output is called a *ciphertext* (C). The ciphertext is sent over the insecure channel, and the receiver recovers the plaintext using a *dcryption algorithm*, which also takes a *decryption key*, K_D, as a parameter. Then the plaintexts are reassembled into the original message. The term cipher (or *encryption scheme*) refers to the (parameterized) encryption/decryption algorithms.

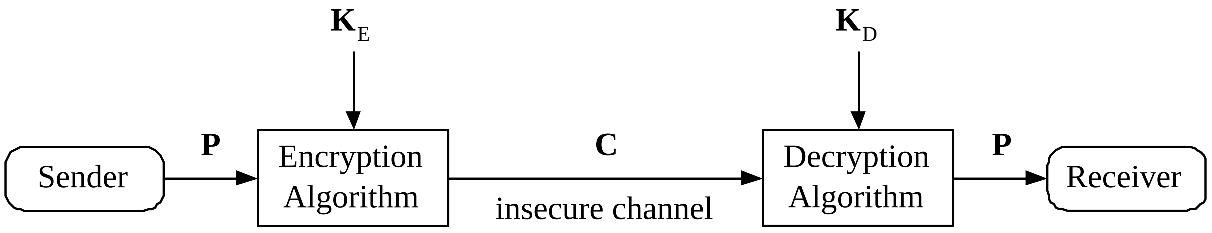


Figure 2.2: Operation of a cipher

2.3.1 Symmetric-Key and Public-Key Ciphers

There are two main categories of ciphers. In a *symmetric-key cipher*, the keys K_E and K_D are equal, or are easily derived from each other. In this thesis we will assume that $K_E = K_D = K$. Clearly that the sender and receiver have to establish a shared key when using a symmetric-key cipher — this is called the *key-distribution problem*.

In a *public-key cipher* [14] (also called *asymetric-key cipher*), each communicating party has a key pair (K_E, K_D) which are different. the encryption key K_E, also called the *public key*, can be widely distributed, while the decryption key K_D, also called the *pribate key*, is only known to the receiver.

Public-key ciphers provide an elegant solution to the key distribution problem. But public-key ciphers are much slower than symmetric-key cipher (e.g., 1/1000 the speed [5]), and require much longer key bits to achieve the same level of security. Therefore, hybrid techniques incorporating symmetric-key and public-key ciphers are common. For example, the sender can generate a key K to be used in a symmetric-key cipher. This key is then encrypted with the receiver's public key, and sent over the channel. The receiver decrypts K with his private key, and then they can transmit the messages using a symmetric-key cipher with the key K.

2.3.2 Block and Stream Ciphers

In Figure 2.1, symmetric-key ciphers can be categorized into *block ciphers* and *stream ciphers*. A block cipher is a bijective mapping form $\{0,1\}^N$ to $\{0,1\}^N$, with the key $K \in \{0,1\}^{N_K}$ (N is the *block size* and N_K is the *key length*). For a block cipher, a given plaintext will always map to the same ciphertext with a fixed key; this straightforward application of a block cipher is called *electronic codebook (ECB) mode*. There are four "modes for operation" in total for the block ciphers (FIPS Pub 81 in December 1980 [15]). The corresponding ciphertext will be different when using other three modes.

A stream cipher breaks a message to be encrypted into much smaller plaintexts, typically individual bits, $x_1, x_2, \ldots$. And the key K is expanded into a *keystream*, $z_1, z_2, \ldots$. The i^{th} ciphertext is obtained by combining x_i and z_i according some rules, often the XOR operation.

2.4 Block Cipher Architectures

Claude Shannon [16] gave two principles of the block ciphers in 1949: *confusion* and *diffusion*. Confusion is the obscuring of the relationship between the plaintext and the ciphertext, and is often a nonlinear *substitution* on a small subblock. Diffusion involves spreading out patterns in the plaintext so that thay are no longer detectable in the ciphertext, and is often a *linear transformation* (sometimes called *diffusion layer*) of subblock connections [17].

Substitution-permutation networks (SPN) [18] and *Feistel networks* [19] are two main block cipher architectures. They both use substitution and linear transformation to implement Shannon's principles. They also are *product ciphers* and *iterated ciphers*. A product cipher is constructed by composing two or more encryption operations, and is stronger that each of its constituent operations. An iterated cipher is a product cipher that consists of repeated application of the same encryption step, called a *round* [1]. In general, different keying material is used in each round.

2.4.1 Substitution-Permutation Networks

An R-round substitutino-permutation network (SPN) [18] requires $(R+1)$ N-bit subkeys, $K^1, K^2, \ldots, K^R, K^{R+1}$. Each round consists of three *layers*: *key-mixing*, *substitution*, and *linear transformation*. In the key-mixing layer, the N-bit ruond input is bitwise XORed with the subkey for that round. In the substitution layer, the resulting block is partitioned into M subblocks of size n $(N = Mn)$, and each subblock becomes the input to a bijective $n \times n$ *substitution box* (*s-box*), which is a bijective mapping from $\{0,1\}^n$ to $\{0,1\}^n$. In the linear transformation layer, the output from the substitution layer is processed through an invertible N-bit linear transformation. We present the linear transformation as an invertible $N \times N$ binary matrix, and use $\mathcal{L}$ to denote it. The linear transformation is usually omitted from the last round, because it won't strengthen the security of the SPN if we include the linear transformation layer. After R rounds, the output is XORed with the final subkey, K^{R+1}, to form the ciphertext. Figure 2.3 gives an example SPN with $N = 16$, $M = n = 4$, and $R = 3$.

The decryption algorithm is to run the SPN "backwards." At first the ciphertext is XORed with K^{R+1}, and then from round R down to round 1, with the inverse linear transformation, inverse s-boxes, and subkey K^r.

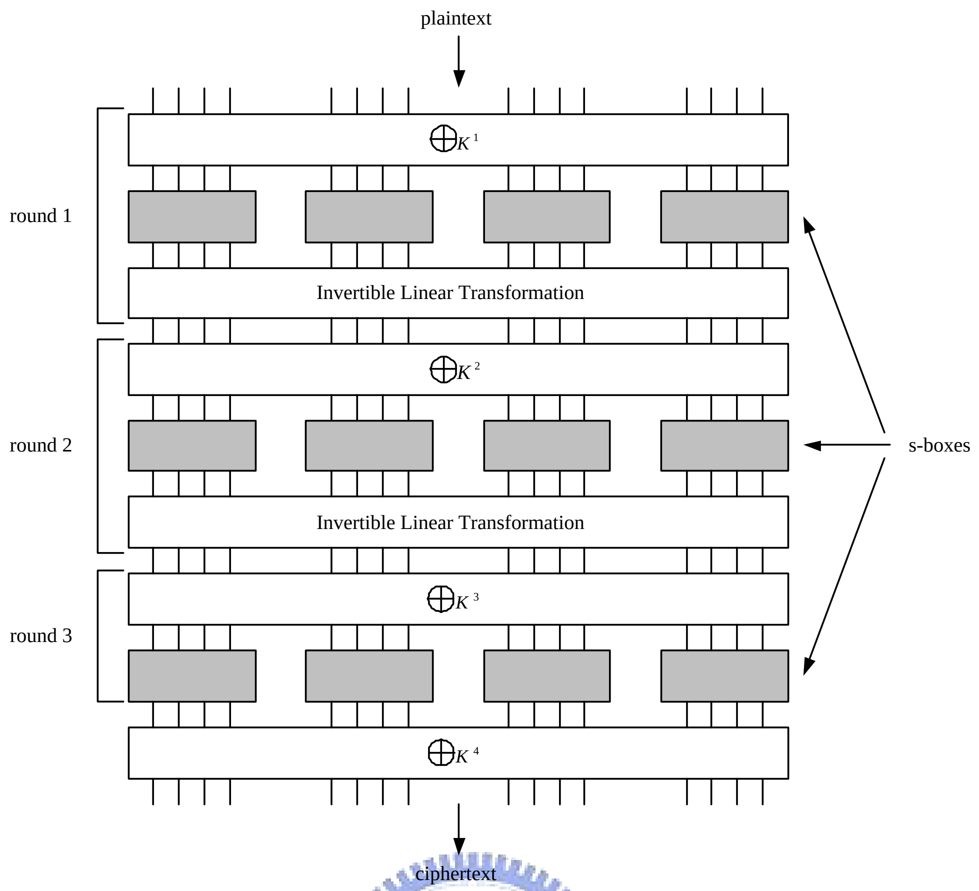


Figure 2.3: An SPN structure with $N = 16$, $M = n = 4$, $R = 3$

2.4.2 Feistel Networks

A Feistel network [19] is a block cipher that modifies *half* of the current block in each round. An R-round Feistel network needs R subkeys, $K^1, K^2, \ldots, K^R$. Let the left and right halves of the N-bit input to round t be denoted x_L^t and x_R^t. The right half, x_R^t, and the round key, K^t, will be the inputs to a *round function*, $f_t : \{0,1\}^{N/2} \to \{0,1\}^{N/2}$. The output from f_t will be XORed with x_L^t to form X_R^{t+1}, the right half of the next round; and x_L^{t+1} will be x_R^t without changed. This swapping of half blocks occurs in every round , except the last round. With the notations above, the plaintext will be $P = \langle x_L^1, x_R^1 \rangle$ and the ciphertext will be $C = \langle x_L^{R+1}, x_R^{R+1} \rangle$. Figure 2.4 shows the basic Feistel network structure.

There are many approaches to design Feistel network round functions. A common theme is to incorporate the basic features of an SPN round, i.e., some arrangement of s-boxes and linear transformations [19, 20, 21, 22, 23]. However, a Feistel network round function does not need to be invertible, allowing greater flexibility to be designed.

While implementation, one of the main advantages of the Feistel network structure is that encryption and decryption operations are the same. A ciphertext can be decrypted by processing it througn the encryption algorithm, but reversing the order of the round

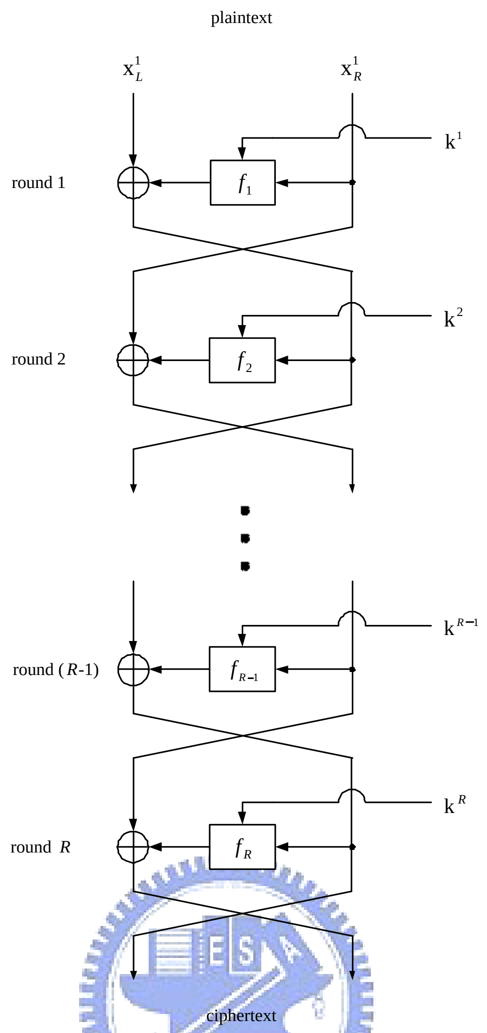


Figure 2.4: Basic Feistel network architecture

functions and corresponding subkeys. We do not need to generate or store the inverse components.

Schneier and Kelsey introduced the concept of *unbalanced Feistel networks* (UFNs) [24]. In a UFN, x_L^t and x_R^t are not equal in size (conventional Feistel networks are call *balanced* in [24]). If the lengths of x_L^t and x_R^t are u and v bits, $N_L + N_R = N$, then $f_t : \{0,1\}^{N_R} \to \{0,1\}^{N_L}$. the inputs to the next round, x_L^{t+1} and x_R^{t+1} are defined as

$$x_L^{t+1} \| x_R^{t+1} = x_R^t \| (f_t(x_R^t, K^t) \oplus x_L^t),$$

where $\|$ is the concatenation operator. Clearly this is a modified version of the structure in Figure 2.4. Schneier and Kelsey give preliminary arguments that in certain cases a UFN may have increased resistance to certain attacks. Variations of the UFN approach are used in ciphers such as CAST-256 [25] and MARS [26].

2.4.3 Other Block Cipher Architectures

Besides SPNs and Fesitel networks, there are lots of ciphers that use other structures. However, most retain the basic concept of constructing a cipher from repeated rounds.

We give two exapmples here.

The first one is IDEA (*International Data Encryption Algorithm*) [27], earlier called IPES [28, 29] (Improved Proposed Encryption Standard), which is a 8-round block cipher. IDEA has a 64-bit input, the round input is split into four 16-bit words, and these are combined with each other and with six 16-bit subkeys using a combination of binary operations on $\{0,1\}^{16}$ from different algebraic groups. Figure 2.5 show the structure of IDEA. IDEA has been extensively analyzed and widely implemented [17].

Another one is RC6 [30], which has a 128-bit block size and consists of 20 rounds. RC6 can be viewed as two 64-bit Feistel networks operating in parallel, with interactions occuring in each round. RC6 uses *data-dependent rotations* — bitwise rotations of data words in which the amount of ratation depends on other intermediate values.

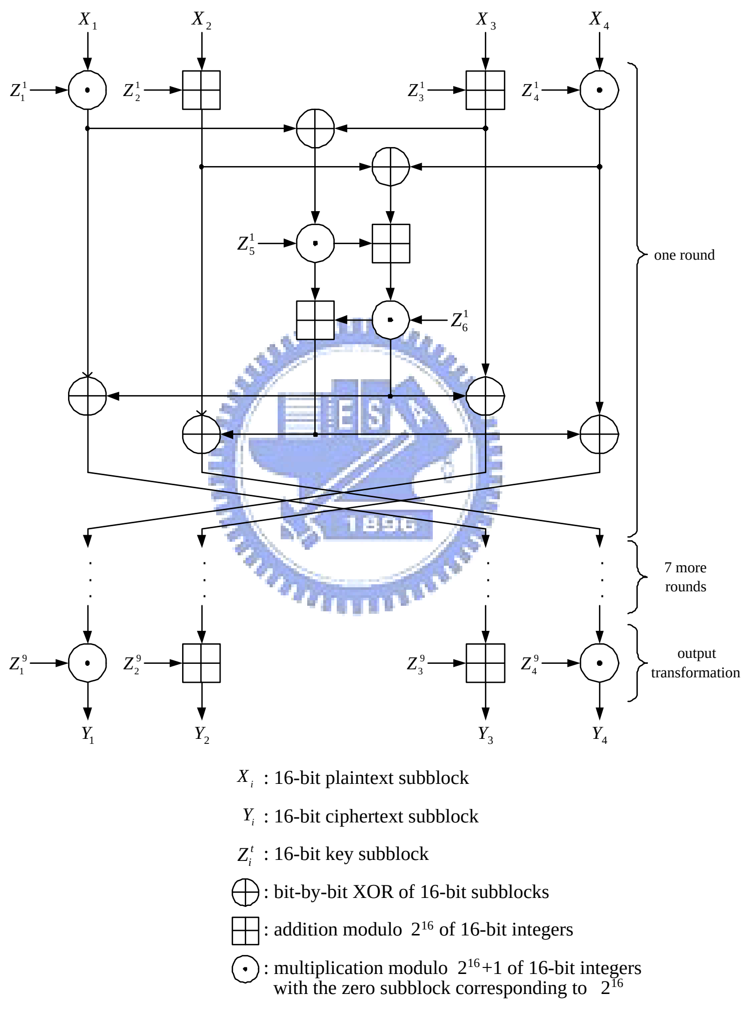

X_1
X_2
X_3
X_4
Z_1^1
Z_2^1
Z_3^1
Z_4^1
Z_5^1
Z_6^1
one round
7 more rounds
Z_1^9
Z_2^9
Z_3^9
Z_4^9
output transformation
Y_1
Y_2
Y_3
Y_4
X_i : 16-bit plaintext subblock
Y_i : 16-bit ciphertext subblock
Z_i^r : 16-bit key subblock
: bit-by-bit XOR of 16-bit subblocks
: addition modulo 2^{16} of 16-bit integers
: multiplication modulo $2^{16}+1$ of 16-bit integers with the zero subblock corresponding to 2^{16}

Figure 2.5: Encryption process of IDEA

2.5 Block Cipher Standards

Standardization is very important for people in the world when communicating with others using ciphers. Here we discuss three important standards.

2.5.1 Data Encryption Standard (DES)

The first major initiative was a call for proposals in 1973 for a cipher standard by the National Bureau of Standards (NBS, now the National Institute of Standards and Technology(NIST)) of the U.S. Department of Commerce. But cryptography was still in its infancy at this time. The NBS received only one candidate — a block cipher called Lucifer [18, 31], developed by Horst Feistel and his colleagues at IBM in 1971. The NBS published a modified vesion of Lucifer, called the Data Encryption Standard [9] (DES). DES is a 16-round Feistel network with a 64-bit block size and a 56-bit key (the small key size was a sourse of criticism at the beginning [32]). The publication of DES marked the beginning of the widespread study of block ciphers. Many cryptanalysis are developed to find weaknesses in DES [33, 34, 35, 6, 36]. In addition, many DES-like block ciphers are proposed and studied [37, 38, 39, 8].

2.5.2 Advanced Encryption Standard (AES)

On January 2, 1997, NIST began a process of choosing a replacement for DES, called the Advanced Encryption Standard [40] (AES). AES were required to be block ciphers with block size of 128 bits and key lengths of 128, 192, and 256 bits. In August 1999, five of the candidates were chosen by NIST as finalists: MARS [26], RC6 [30], Rijndael [41, 42], Serpent [43], and Twofish [44]. On October 2, 2000, Rijndael was selected to be the Advanced Encryption Standard, and published in the Federal Register on December 4, 2001 [45].

Rijndael is an SPN with 16 8×8 s-boxes in each substitution layer, and all s-boxes are identical. The linear transformation consists of two steps: a byte permutation, and the parallel application of four copies of highly diffusive 32-bit linear transformation. Figure 2.6 shows a single round of Rijndael.

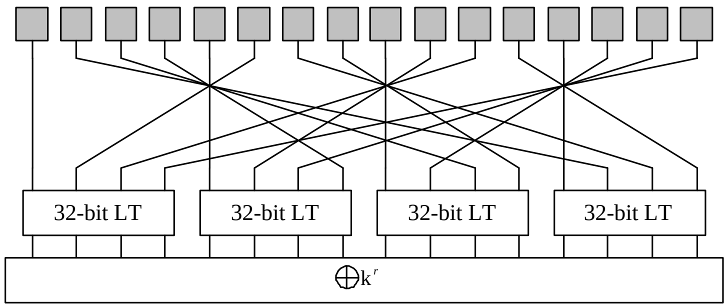


Figure 2.6: One round of Rijndael (the AES)

Rijndael actually supports block sizes and key lengths from 128 to 256 bits, increments of 32 bits. However, the AES only uses a 128-bit block size and key lengths of 128, 192, and 256 bits. The number of rounds varies according to the block size and key length, as shown in Table 2.1.

128-bit block size			
key length (bit)	128	192	256
number of rounds	10	12	14

Table 2.1: The relation between key lengths and the numbers of rounds in AES

2.6 Attacks on Block Ciphers

There are a lot of attacks on block ciphers. Generally, the goal of an attack is to decrypt any ciphertext. A successful attack is to derive the key (a *total break*), or to construct an algorithm to decrypt ciphertexts without the key (*global deduction*) [46].

The *attack model* specifies the information available to the attacker [1]. The most comn types of attack models are enumerated as follows:

1. *Ciphertext-only attack*: attacker possesses one or more ciphertexts.

2. *Known-plaintext attack*: attacker possesses one or more plaintexts and the corresponding ciphertexts.

3. *Chosen-plaintext/ciphertext attack*: attacker can choose a set of plaintexts/ciphertexts, and ask the corresponding ciphertexts/plaintexts.

4. *Adaptive chosen-plaintext/ciphertext attack*: attacker can submit plaintexts/ciphertexts for encryption/decryption, with the freedom to base later choices on the results of earlier submissions.

The *data complexity* of an attack is the number of data required. The *time complexity* of an attack is the number of steps required, where a "step" is often a single encryption, or some other appropriate computational unit [17].

We describe some attacks on block ciphers. We will usr N_K to denote the number of bits in the key.

2.6.1 Exhaustive Key Search

For a known $\langle$plaintext, ciphertext$\rangle$ pair, $\langle P, C\rangle$, *exhaustive key search* is to encrypt P with each of the 2^{N_K} keys, and discard the keys that do not produce the corresponding ciphertext, C. A small number of $\langle$plaintext, ciphertext$\rangle$ pairs are required to identify the correct key. Exhaustive key search is usually considered the "benchmark" against which other attacks are measured.

2.6.2 Linear Cryptanalysis

Linear Cryptanalysis, which is introduced by Matsui in 1993 [6], is a known-plaintext attack (ciphertext-only if plaintexts consist of random ASCII codes) that is considered to be one of the most powerful attacks on block ciphers. Linear cryptanalysi was the first attack actually implemented to break DES [10]. Matsui used 2^{43} known $\langle$plaintext, ciphertext$\rangle$ pairs to recover 2^{26} key bits, and then used the exhaustive search the remaining 2^{30} key bits. Linear cryptanalysis needs the existence of the relatively large deviation of the value on a the input bits and output bits of an s-box. In Chapter 3, we will give a detailed description of linear cryptanalysis on SPN.

2.6.3 Differential Cryptanalysis

Differential cryptanalysis is a chosen-plaintext attack presented by Biham and Shamir in 1990 [47, 48, 49, 33]. A differential-like attack was also published by Mruphy in 1990, which was applied to FEAL [50]. Differentail cryptanalysis was the first attack able to break DES faster than exhaustive key search, with data complexity 2^{47} and time complexity 2^{37} [33]. The main difference from linear cryptanalysis is that, efferential cryptanalysis involves comparing the XOR of two input to the XOR of the corresponding two outputs.

Chapter 3

Linear Cryptanalysis on SPN

In this chapter we will talk about the things need to be known and need to be done before we break a SPN cryptosystem. First of all, we will make an introduction about the original idea of Matsui's linear cryptanalysis in Section 3.1. And then we will In Sections 3.2 and 3.3 we will study the *linear approximation*, *linear characteristics*, and a related important lemma, Matsui's *Piling-up Lemma*. Last of all we talk about some techniques on linear cryptanalysis.

Notations

We first describe the notations and definitions we will use through this thesis. We let P be denoted as the plaintext, and the corresponding ciphertext is C, with the key K. For each plaintext/ciphertext pair, we call it a *text pair*, and $\mathcal{N}_L$ be the number of the test pairs we need when attack.

For an SPN structure, we use the following notations (see Figure 3.1):

- N: block size
- M: the number of s-boxes in one round
- n: size of a s-box
- N_K: key length
- K^t: round key of the round t
- $S(\cdot)$: s-box mapping function
- $\mathcal{L}$: linear transformation mapping function

The following definations are *bias* and *linear probability* [51]. These definitions are very important for us to find the best linear expression.

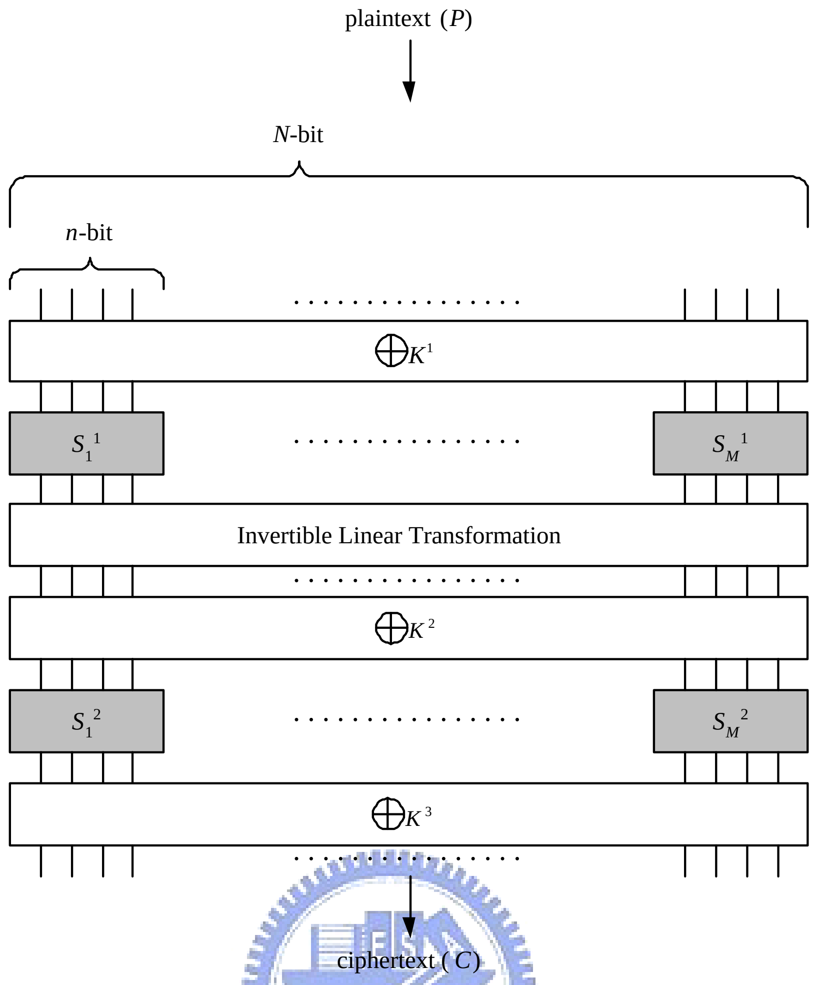


Figure 3.1: Notations

Definition 3.1. *Let* $\mathbf{X}$ *be a random variable, whose value is* 0 *with probability* p *or* 1 *with probability* $1-p$*. Then the* bias *of* $\mathbf{X}$*is defied as*

$$\varepsilon = p - \frac{1}{2}.$$

Definition 3.2 ([51]). *Let* $B : \{0,1\}^d \to \{0,1\}^d$ *be bijective, and let* $\mathbf{a}, \mathbf{b} \in \{0,1\}^d$ *be fixed. If* $\mathbf{X} \in \{0,1\}^d$ *is a uniformly distributed random variable, then the linear probability* $LP(\mathbf{a}, \mathbf{b})$ *is defined as*

$$LP(\mathbf{a}, \mathbf{b}) \overset{def}{=} (2 \cdot \Pr_{\mathbf{X}}\{\mathbf{a} \bullet \mathbf{X} = \mathbf{b} \bullet B(\mathbf{X})\} - 1)^2. \tag{3.1}$$

If B *is parameterized by a key,* K*, we write* $LP(\mathbf{a}, \mathbf{b}; K)$*, and the expected liner probability* $ELP(\mathbf{a}, \mathbf{b})$ *is defined as*

$$ELP(\mathbf{a}, \mathbf{b}) \overset{def}{=} \exp_{\mathbf{K}}[LP(\mathbf{a}, \mathbf{b}; \mathbf{K})], \tag{3.2}$$

where $\mathbf{K}$ *is a random variable uniformly distributed over the space of keys.*

The values $\mathbf{a}$ and $\mathbf{b}$ in Definition 3.2 are called input and output *masks* (or *selections* *pattern* [42]). For our purposes, the bijective mapping B will be an s-box, a single round, or a sequence of consecutive encryption rounds.

The LP values lie in the interval $[0, 1]$. If $\mathbf{X}$ and $B(\mathbf{X})$ are uncorrelated, then $\mathbf{a} \bullet \mathbf{X}$ and $\mathbf{b} \bullet B(\mathbf{X})$ will be equal exactly half the time, i.e., $\Pr_{\mathbf{X}}\{\mathbf{a} \bullet \mathbf{X} = \mathbf{b} \bullet B(\mathbf{X})\} = 1/2$, and the corresponding LP value will be 0. If $LP(\mathbf{a}, \mathbf{b}) = 1$, then $\Pr_{\mathbf{X}}\{\mathbf{a} \bullet \mathbf{X} = \mathbf{b} \bullet B(\mathbf{X})\}$ will be either 1 or 0. $LP(\mathbf{a}, \mathbf{b}) = 0$ if either $\mathbf{a} = \mathbf{0}$ and $\mathbf{b} \neq \mathbf{0}$, or $\mathbf{a} \neq \mathbf{0}$ and $\mathbf{b} = \mathbf{0}$.

The following lemma derives immediately from *Parseval's Theorem*.

Lemma 3.3 ([52]). *Let $B : \{0, 1\}^d \to \{0, 1\}^d$ be a bijective mapping parameterized by a key, K, and let $\mathbf{a}, \mathbf{b} \in \{0, 1\}^d$. Then*

$$\sum_{\mathbf{x} \in \{0,1\}^d} LP(\mathbf{a}, \mathbf{x}; K) = \sum_{\mathbf{x} \in \{0,1\}^d} LP(\mathbf{x}, \mathbf{b}; K) = 1,$$
$$\sum_{\mathbf{x} \in \{0,1\}^d} ELP(\mathbf{a}, \mathbf{x}) = \sum_{\mathbf{x} \in \{0,1\}^d} ELP(\mathbf{x}, \mathbf{b}) = 1.$$

3.1 Principle of Linear Attack

The purpose of Linear Attack is to find the following linear expression, which holds with probability $p \neq 1/2$ for randomly given plaintext P, the corresponding ciphertext C and the fixed secret key K:

$$P[i_1, i_2, \ldots, i_a] \oplus C[j_1, j_2, \ldots, j_b] = K[k_1, k_2, \ldots, k_c], \tag{3.3}$$

where $i_1, i_2, \ldots, i_a, j_1, j_2, \ldots, j_b$, and $k_1, k_2, \ldots, k_c$ denote fixed bit locations.

Since both sides of equation (3.3) essentially represent one-bit information, the magnitude of $|p - 1/2|$ expressed the effectiveness. We will refer to the most effective linear approximate expression (i.e., $|p - 1/2|$ is maximal) as the best expression and its probability as the best probability. Once we succeed in reaching an effective linear expression, it is possible to determine one key bits $K[k_1, k_2, \ldots, k_c]$ by the following algorithm based one the maximum likelihood method:

Algorithm 1 ([6]).

Step 1. Let T be the number of plaintexts such that the left side of equation (3.3) is equal to zero.

Step 2. If $T > N/2$ (N denotes the number of plaintexts),

(1) then guess $K[k_1, k_2, \ldots, k_c] = 0$ (when $p > 1/2$) or 1 (when $p < 1/2$),

(2) else guess $K[k_1, k_2, \ldots, k_c] = 1$ (when $p > 1/2$) or 0 (when $p < 1/2$).

In Algorithm 1, we guess the value of the linear expression $K[k_1, k_2, \ldots, k_c]$. But it is hard if we only guess the XORed value of the key bits. Therefore, for a practical known-plaintext attack, we make use of the best expression of $(n-1)$-round cipher, i.e., we regard the first round or the final round as having been deciphered using K_1 or K_{n+1} in SPN. Matsui then use the folling type of expression which holds with the best probability of $(n-1)$-round DES cipher:

$$\begin{aligned} P[i_1, i_2, \ldots, i_a] &\oplus C[j_1, j_2, \ldots, j_b] \\ &\oplus F_n(C_L, K_n)[l_1, l_2, \ldots, l_d] = K[k_1, k_2, \ldots, k_c], \end{aligned} \tag{3.4}$$

Accord to euqation (3.4), the following maximum likelihood method can be applied to deduce K_n and $K[k_1, k_2, \ldots, k_c]$:

Algorithm 2 ([6]).

Step 1. For each candidate $K_n^{(i)}$ $(i = 1, 2, \ldots)$ of K_n, let T_i be the number of plaintexts such that the left side of equation (3.4) is equal to zero.

Step 2. Let T_{max} be the maximal value and T_{min} be the minimal value of all T_i's.

(1) If $|T_{max} - N/2| > |T_{min} - N/2|$, then adopt the key candidate cooresponding to T_{max} and guess $K[k_1, k_2, \ldots, k_c] = 0$ (when $p > 1/2$) or 1 (when $p < 1/2$).

(2) If $|T_{max} - N/2| < |T_{min} - N/2|$, then adopt the key candidate cooresponding to T_{min} and guess $K[k_1, k_2, \ldots, k_c] = 1$ (when $p > 1/2$) or 0 (when $p < 1/2$).

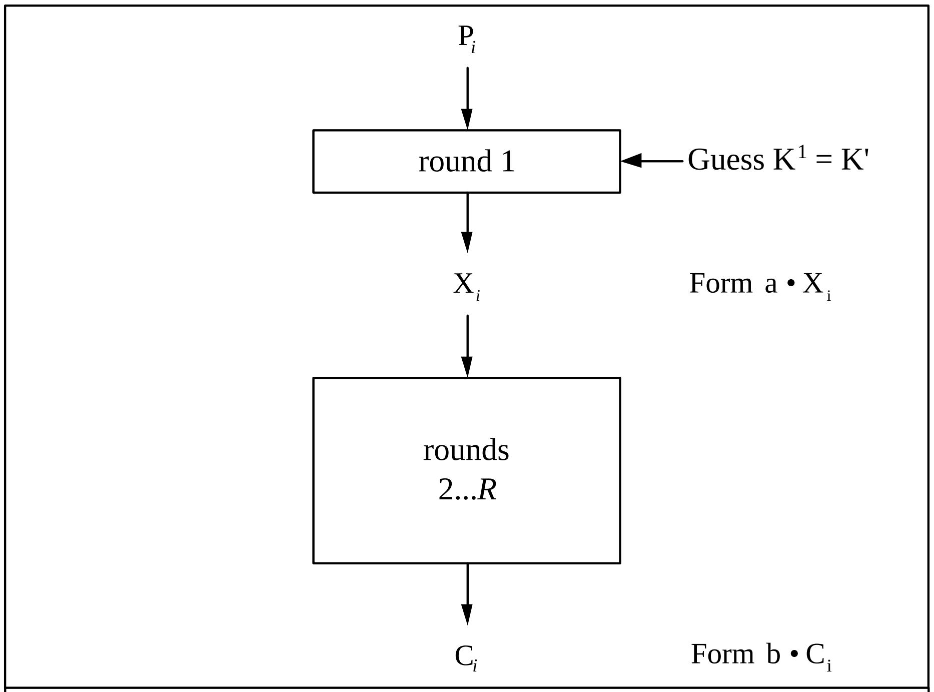
This algorithm is very improtant and most of the linear cryptanalysis on other cryptosystems are based on Matsui's Algorithm 2. The next aim is to find the linear expression of equation (3.4) which will enhance the success rate of Algorithm 2.

As applied Algorithm 2 to SPNs, the attack proceeds as in Figure 3.2. Note that P, C, and X are plaintext, ciphertext, and intermediate input to round 2, and $\mathbf{a}$ and $\mathbf{b}$ are input and output masks.

We view rounds $2 \ldots R$ as a single key-dependent function mapping $\{0,1\}^N \rightarrow \{0,1\}^N$, where $\tilde{K} = \langle K^1, k^2, \ldots, K^T \rangle$ is the key being used. Ideally, we want to precompute masks $\mathbf{a}, \mathbf{b} \in \{0,1\}^N \backslash \mathbf{0}$ that maximize $LP(\mathbf{a}, \mathbf{b}; \tilde{K})$. We will study it in Section 3.3.

3.2 Linear Approximations of S-boxes

In this section we study linear approximation of S-boxes. Our approach is to investigate the probability that a value on an input bit coincides with a value on an output bit. More



1. Find N_L known test pairs: $<P_1, C_1>, \ldots, <P_{NL}, C_{NL}>$
2. Guess $K^1 = K'$. Encrypt each P_i through round 1 to obtain X_i
3. If $a \bullet X_i = b \bullet C_i$ then increment the counter Counter[K']
4. Choose the K' which maximizes |Counter[K']-$N_L/2$|

Figure 3.2: Summary of linear cryptanalysis (Algorithm 2)

generally, it is useful to deal with not only on bit position but also an XORed value of several bit positions.

We start with the following definition:

Definition 3.4 ([6]). *For a given S-box $S : \{0,1\}^n \rightarrow \{0,1\}^n$, and let $\alpha \in \{0,1\}^n$, $\beta \in \{0,1\}^n$, we define $N_S(\alpha, \beta)$ as the number of times out of all the input patterns of S, such that an XORed value of the input bits masked by α coincides with an XORed value of the output bits masked by β. That is to say,*

$$N_S(\alpha, \beta) \overset{def}{=} \#\{X \,|\, \forall X \in \{0,1\}^n, \alpha \bullet X = \beta \bullet S(X)\}, \tag{3.5}$$

where $\alpha \bullet X$ denotes the inner product of α and X over $GF(2)$.

Accroding to Definition 3.2 and 3.4, we can define that, for an s-box S,

$$LP^S(\alpha, \beta) = (2 \cdot \frac{N_S(\alpha, \beta)}{2^n} - 1)^2.$$

Example 1: Suppose S is a bijective 4×4 s-box:

X	$S(X)$
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

It is easily to see that $N_S(8, 15) = 2$.

From the example 1, we can see that the second input bit (from left) of S coincides with an XORed value of all output bits with probability $2/16 = 0.125$. According the SPN structure, if S is the leftest s-box in the first round, then we have the following linear expression:

$$P[2] \oplus K^1[2] = S(P[1, 2, 3, 4])[1, 2, 3, 4],$$

that is,

$$P[2] \oplus S(P[1, 2, 3, 4])[1, 2, 3, 4] = K^1[2].$$

For an s-box S, we can make a table of all the $N_S(\alpha, \beta)$ values for all α and β. This is called the *linear approximation table*. Table 3.1 is a linear approximation table of the s-box given in example 1.

According to the linear approximation table, we choose the one such that $LP^S(\alpha, \beta)$ (recall that $LP^S(\alpha, \beta) = (2 \cdot \dfrac{N_S(\alpha, \beta)}{2^n} - 1)^2)$ is maximal. That is, the probability of the XORed value of the input and output bits is correlated; this is good for us to attack the system. For example, the most effective linear approximation in table 3.1 are $N_S(1, 7)$, $N_S(2, 14)$, $N_S(3, 9)$, and $N_S(8, 15)$.

3.3 Linear Characteristics

In the previous section, we studied how to find the best linear expression of one s-box. But what we need is a linear experssion with plaintext bits, ciphertext bits, and the key bits. Therefore we have to find several s-boxes to produce a linear expression like equation 3.3. Here is an example:

	β															
α	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	16	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
1	8	8	6	6	8	8	6	14	10	10	8	8	10	10	8	8
2	8	8	6	6	8	8	6	6	8	8	10	10	8	8	2	10
3	8	8	8	8	8	8	8	8	10	2	6	6	10	10	6	6
4	8	10	8	6	6	4	6	8	8	6	8	10	10	4	10	8
5	8	6	6	8	6	8	12	10	6	8	4	10	8	6	6	8
6	8	10	6	12	10	8	8	10	8	6	10	12	6	8	8	6
7	8	6	8	10	10	4	10	8	6	8	10	8	12	10	8	10
8	8	8	8	8	8	8	8	8	6	10	10	6	10	6	6	2
9	8	8	6	6	8	8	6	6	4	8	6	10	8	12	10	6
10	8	12	6	10	4	8	10	6	10	10	8	8	10	10	8	8
11	8	12	8	4	12	8	12	8	8	8	8	8	8	8	8	8
12	8	6	12	6	6	8	10	8	10	8	10	12	8	10	8	6
13	8	10	10	8	6	12	8	10	4	6	10	8	10	8	8	10
14	8	10	10	8	6	4	8	10	6	8	8	6	4	10	6	8
15	8	6	4	6	6	8	10	8	8	6	12	6	6	8	10	8

Table 3.1: Linear Approximation Table for example 1

Example 2: Figure 3.3 shows a simple SPN structure. In the first round, we choose the second s-box with input mask 8 and output mask 15. Then we have one expression:

$$P[1] \oplus K^1[1] = S_1^1(P[1,2,3,4])[1,2,3,4].$$

And then, following the arrows, all four s-boxes in round 2 are selected after permutation. So we have four more expressions now:

$$\begin{aligned}
P'[1] \oplus K^2[1] &= S_1^2(P'[1,2,3,4])[1,2,3,4],\\
P'[5] \oplus K^2[5] &= S_1^2(P'[5,6,7,8])[1,2,3,4],\\
P'[9] \oplus K^2[9] &= S_1^2(P'[9,10,11,12])[1,2,3,4],\\
P'[13] \oplus K^2[13] &= S_1^2(P'[13,14,15,16])[1,2,3,4].
\end{aligned}$$

Consequently, we can obtain the following expression by combining the expressions above and adding the final round key K^3:

$$P[1] \oplus C[1,\ldots,16] = K^1[1] \oplus K^2[1,5,9,13] \oplus K^3[1,\ldots,16].$$

From Figure 3.3, there is a path from plaintext side to ciphertext side, and we use it to fine a linear expression of the SPN. We call this path as a *trail*. From the figure above, we can see that a trail can be viewed as a sequence of input masks for each round. For example, the sequences in Figure 3.3 is $\langle(8000),(8888),(FFFF)\rangle$ (in hexadecimal). This is called the *linear characteristics* (or simply *characteristics*) of a trail.

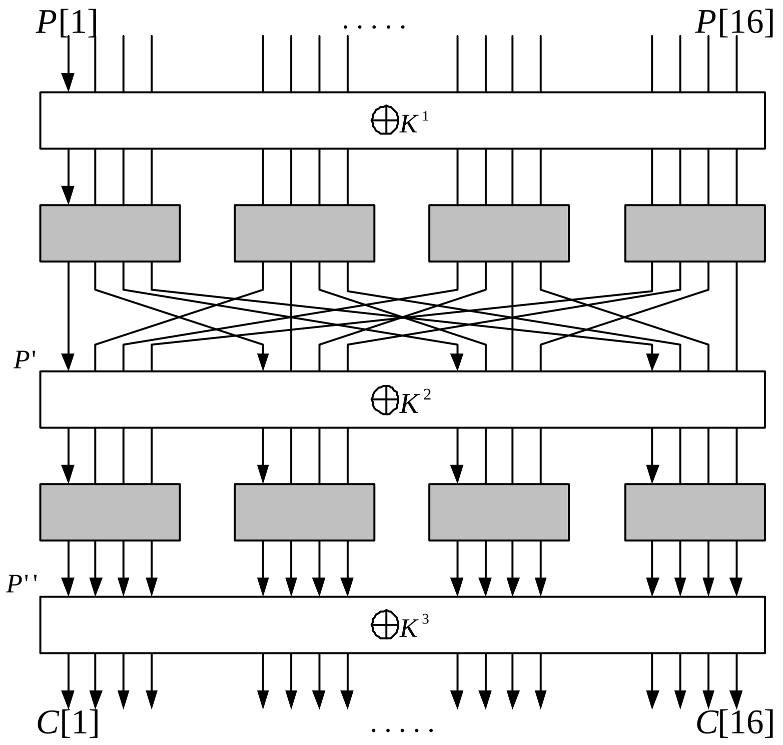


Figure 3.3: A trail of a SPN with $M = n = 4$ ($N = 16$), $R = 2$, and the permutation of Kam and Davida [3]

Definition 3.5 ([2]). *A one-round characteristic for rout t is a pair of N-bit masks, $\Omega^t = \langle \mathbf{a}^t, \mathbf{b}^t \rangle$; we view $\mathbf{a}^t$ and $\mathbf{b}^t$ as input and output masks for round t.*

Definition 3.6 ([2]). *A T-round characteristic for rounds $1 \ldots T$ is a $(T+1)$-tuple of N-bit masks, $\Omega = \langle \mathbf{a}^1, \mathbf{a}^2, \ldots, \mathbf{a}^T, \mathbf{a}^{T+1} \rangle$; we view $\mathbf{a}^t$ as the intput for round t $(1 \leq t \leq T)$, and $\mathbf{a}^{T+1}$ is the output mask for round T.*

What we want to know is the total probability of a trail (i.e., $LP(\mathbf{a}, \mathbf{b})$, where $\mathbf{a}$ and $\mathbf{b}$ are input and output masks of the SPN). To compute the total probability, we can use Matsui's Piling-up Lemma [6]:

Lemma 3.7 (Piling-up lemma [6]). *Let $\mathbf{X_i}$ $(1 \leq i \leq n)$ be independent random variables whose values are 0 with probability p_i or 1 with probability $1 - p_i$. Then the probability that $\mathbf{X_1} \oplus \mathbf{X_2} \oplus \cdots \oplus \mathbf{X_n} = 0$ is*

$$\frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \left(p_i - \frac{1}{2} \right). \tag{3.6}$$

Therefore we can treat different linear approximation expressions of a s-box (e.g., $P[2] \oplus S(X)[1,2,3,4] = K^1[2]$ we mentioned before) as distinct independent random variables $\mathbf{X_i}$, and $\Pr\{\mathbf{X_i} = 0\} = (N_S(\alpha, \beta)/2^m)$. Then the total probability can be calcultate easily.

By the Piling-up Lemma, we define the *linear characteristic probability* (LCP) and *expected linear characteristic probability* (ELCP):

Definition 3.8. *Let $\Omega^t = \langle \mathbf{a}^t, \mathbf{b}^t \rangle$ be a one-round characteristic for round t, and K^t is the subkey of round t. Let the M s-boxes of round t be enumerated from left to right as $S_1^t, S_2^t, \ldots, S_M^t$, and the input and output masks for S_i^t be denoted α_i^t and β_i^t. The LCP and ELCP of Ω^t are defined as*

$$LCP^t(\Omega^t) \overset{def}{=} \prod_{i=1}^{M} LP^{S_i^t}(\alpha_i^t, \beta_i^t), \tag{3.7}$$

$$ELCP^t(\Omega^t) \overset{def}{=} \exp_K[LCP^t(\Omega^t; K^t)]. \tag{3.8}$$

Definition 3.9. *Let $\Omega = \langle \mathbf{a}^1, \mathbf{a}^2, \ldots, \mathbf{a}^T, \mathbf{a}^{T+1} \rangle$ be a T-round characteristic for rounds $1 \ldots T$, and let $\tilde{K} = \langle K^1, k^2, \ldots, K^T \rangle$ be the vector of subkeys being used for these rounds. The LCP and ELCP of Ω are defined as*

$$LCP(\Omega) \overset{def}{=} \prod_{t=1}^{T} LCP^t(\mathbf{a}^t, \mathcal{L}(\mathbf{a}^t)), \tag{3.9}$$

$$ELCP(\Omega) \overset{def}{=} \prod_{t=1}^{T} ELCP^t(\mathbf{a}^t, \mathcal{L}(\mathbf{a}^t)). \tag{3.10}$$

Example 3: From Figure 3.3, we can obtain that $\Omega = \langle (8000), (8888), (FFFF) \rangle$. The linear characteristic probabilities are:

$$LCP^1(\langle (8000), (F000) \rangle) = \prod_{i=1}^{4} LP^{S_i^1}((8000), (F000)) = \frac{9}{16},$$

$$LCP^2(\langle (8888), (FFFF) \rangle) = \prod_{i=1}^{4} LP^{S_i^2}((8888), (FFFF)) = (\frac{9}{16})^4;$$

$$LCP(\langle (8000), (8888), (FFFF) \rangle) = \prod_{t=1}^{2} LCP^t(\Omega^t) = (\frac{9}{16})^5.$$

Choosing the Best Trail

As we said in Section 3.1, the attacker typically runs a search algorithm to find the trail, which has characteristic $\Omega = \langle \mathbf{a}^1, \mathbf{a}^2, \ldots, \mathbf{a}^T, \mathbf{a}^{T+1} \rangle$, with the key $\tilde{K} = \langle K^1, k^2, \ldots, K^T \rangle$, such that $LP(\mathbf{a}, \mathbf{a}^{T+1}; \tilde{K})$ is maximal.

But direct computation of $LP(\mathbf{a}, \mathbf{b}; \tilde{K})$ is generally infeasible, so we compute the expected value of $LP(\mathbf{a}, \mathbf{b}; \tilde{K})$, denoted $E_T[\mathbf{a}, \mathbf{b}]$:

$$E_T[\mathbf{a}, \mathbf{b}] \overset{def}{=} \exp[LP(\mathbf{a}, \mathbf{b}; \mathcal{K})], \tag{3.11}$$

where $\mathcal{K}$ is randomly chosen from the key set.

Harpes et al. [53] presented that

$$LP(\mathbf{a}, \mathbf{b}; \tilde{K}) \approx E_T[\mathbf{a}, \mathbf{b}],$$

and

$$E_T[\mathbf{a}, \mathbf{b}] \approx ELCP(\Omega).$$

Thus, we can find the trail by computing expected linear characteristic probability.

We need the information of the key while computing the ELCP value, but we do not have the key. Fortunately, we hae the following lemma:

Lemma 3.10 ([2]). *Let $1 \leq t \leq T$, and let $\mathbf{a}, \mathbf{b}, K^t \in \{0,1\}^N$. Then $LCP^t(\mathbf{a}, \mathbf{b}; K^t)$ is independent of K^t, and therefore*

$$LCP^t(\mathbf{a}, \mathbf{b}; K^t) = ELP^t(\mathbf{a}, \mathbf{b}). \tag{3.12}$$

Corollary 3.11 ([2]). *Let Ω be a T-round characteristic, and let $\tilde{K}$ be the vector of subkeys for the T rounds under consideration. Then $LCP(\Omega^t; \tilde{K}) = ELCP(\Omega)$.*

Therefore, we can find the best trail by computing the value $LCP(\Omega)$, and we choose the one with the maximal $LCP(\Omega)$.

3.4 Subkeys Attack

Once we have the trail and its linear characteristic probability (or bias), we can begin to extract the subkey bits of the first or the last round. Here we attack first round (extract subkey bits in K^1) for the example.

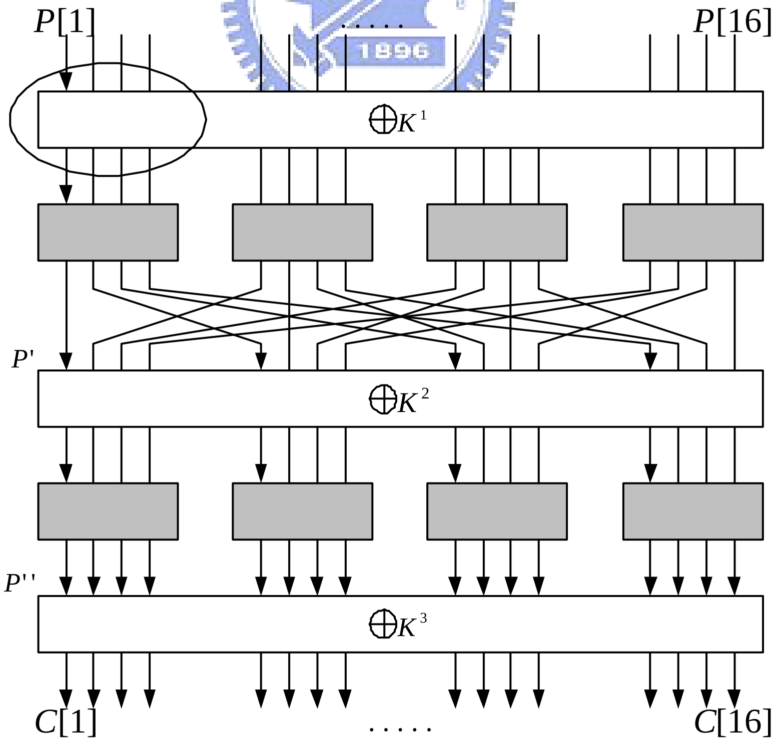


Figure 3.4: A trail for attack

In example 2, we find a tail in Figure 3.3 and its corresponding linear approximation expression. The subkey bits we are going to extract are those involved in the first round of the trail. For example, in the round 1 of Figure 3.4, only the first s-box has a nonzero input. Then the subkey bits begin extracted are the corresponding position of the input bits of the s-box, i.e., the circled part in Figure 3.4.

Since linear cryptanalysis is a known-plaintext attack, we need some test pairs (plaintext/ciphertext pairs), and suppose we have $\mathcal{N}_L$ pairs. We maintain a counter array for each possible subkeys. Then from the Figure 3.2, we partially encrypt the plaintext for each subkeys. If the expression in the figure holds, then we increment the corresponding counter of that subkey.

Last of all, we expect the counter, which is farthest from $\mathcal{N}_L/2$, is the most likely subkey.

3.5 Techniques of Linear Cryptanalysis

In this section we introduce some techniques of linear cryptanalysis. With these techniques, we can increase the success rate and reduce test pairs we need.

3.5.1 Key Ranking

Key ranking is a technique used by Matsui in applying linear cryptanalysis to DES [6]. Matsui used two linear approximation expressions, each of which provides candidate for 13 key bits. And he adopts the reliability of key candidates into consideration. The key candidates means that he stores not only the most likely key bits but also the i^{th} likely candidates. That is, he stores the key $K'_1, K'_2, \ldots$ in order where K'_i is the i^{th} likely dey bits. If K'_1 failed, that is, for a test pair $\langle P, C\rangle$, the result after we encrypt P by K'_1 is not equal to C, then K'_2 is used and so on until the correct one is found. With htis simple improvement, Matsui increased the success rate. In his test , the successfully attacked the 26 key bits of the full 16-round DES with 2^{43} test pairs. The remaining 30 key bits can be found by exhaustive key search. Compared to his original attack, more key bits are attacked with fewer test pairs needed.

3.5.2 Multiple Linear Approximations

Kaliski and Robshaw [11] proposed a new idea on linear cryptanalysis by using *multiple linear approximations*. Suppose that they have n linear approximations, which involve the same key bits but differ in the plaintext and ciphertext bits that they use. For each linear approximation, they assign a different weight a_i, where $a_i = \varepsilon_i / \sum \varepsilon_i$. Then for each possible key bits K'_j, $j = 1, 2, \ldots$, and for each linear approximation i, let T_i be the number of test pairs such that the left side of equation (3.3) is equal to 0. Then we calculate $U^j = \sum_{i=1}^{n} a_i T_i^j$ for each j. And the rest parts are just like the Matsui's Algorithm 2, i.e., we find which U^j is the farthest from $\mathcal{N}_L/2$ and we assume it to be the most likely key bits.

This technique is supposed to increase the success rate and reduce the data complexity. However, int their expriments, the increase of effectiveness on DES is somewhat limited. BIt this is still an important technique since it may be generally applicable to other block ciphers and be extremely effective in reducing data complexity.

Chapter 4

Attack Strategies on SPN

In this chapter we start to attack on SPN. We divide the whole attack process into three phases:

Trails-Finding Phase: Find all possible trails by given s-box and permution mappings.

Attacking Phase:

(1) Calculate the $ELCP(\mathbf{a}, \mathbf{b})$ value and find the best trail whose $ELCP(\mathbf{a}, \mathbf{b})$ value is maximal ($\mathbf{a}$ is the mask of plaintext and $\mathbf{b}$ is the mask of ciphertext).

(2) Evaluate the number of plaintext/ciphertext pairs we need for this trail.

(3) Attack and guess the key bits, and then save some other possible candidate subkeys.

Backtracking Phase: If the subkey we guessed is wrong, then use another candidate subkey.

Figure 4.1 shows the flow chart of attack process. In the following sections we will talk about each part.

4.1 Trails-Finding

Before the attack, we have to find all possible trails for the given s-box mappings and linear transformations. We can use a *recursive* algorithm to find the trails. First of all, for any input mask $\mathbf{a}^t$ for round t, we divide it into M sub-masks, $\alpha_1, \ldots, \alpha_M$, to be the input masks of the M s-boxes. An s-box for which the input and output masks are nonzero is called *active*. Second, for each active s-box with input sub-mask α_i, search the linear approximation table and find all possible output sub-masks, and then combine them. For example, if we use the s-box mapping in Example 1 and the input sub-mask is 1 (0001). From Table 3.1, there are 8 possible output sub-masks (the LP value is not equal to zero): $9, 10, \ldots, 15$.

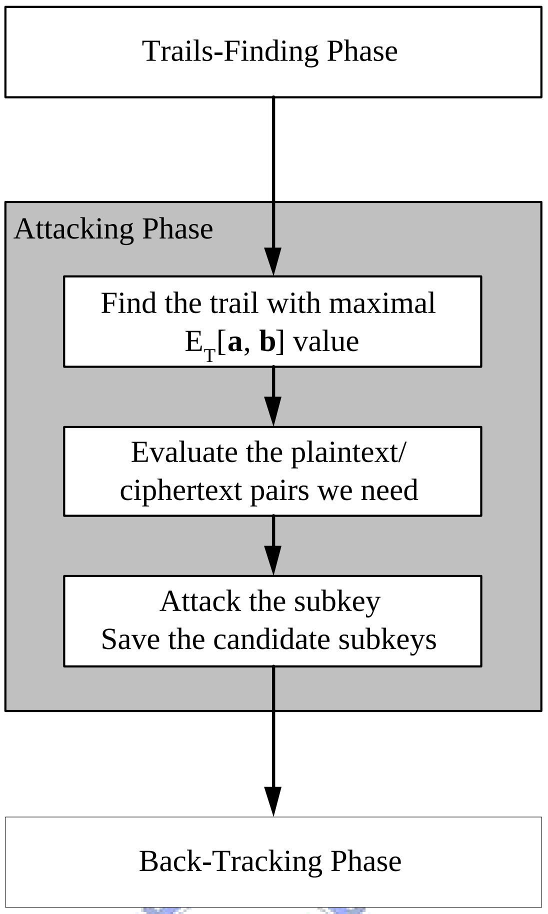

Trails-Finding Phase
Attacking Phase
Find the trail with maximal $E_T[\mathbf{a}, \mathbf{b}]$ value
Evaluate the plaintext/ ciphertext pairs we need
Attack the subkey
Save the candidate subkeys
Back-Tracking Phase

Figure 4.1: Flow chart of the attack

For effeciency, we can also set a *threshold value* to omit the output sub-masks with smaller LP value. In the previous example, if we set the threshold: "LP value must larger then 1/4." Then for the input sub-mask 1, there are only one possible output sub-mask now: 7.

After finding all possible output masks, we do the linear transformation to get the input mask for the next round, $\mathbf{a}^{t+1}$. If we are at the final round, then save the trail to a list and return to try next possible input mask.

But the number of all possible trails is too large and we will spend too much time on finding trails. Therefore, we only use the input masks such that the number of active s-boxes in the first round is 1. For example, we use the mask 0100, but do not use 0011. Thus, the number of the input masks for the first round will be reduced from $2^N (= 2^{Mn})$ to $M2^n$.

Here is the algorithm.

```
Algorithm 4.1.  TRAILS-FINDING($\mathbf{a}^t$, $t$)
  1. $(\alpha_1, \ldots, \alpha_M) \leftarrow \mathbf{a}^t$
  2. Find all possible output masks $\mathbf{b}_1, \mathbf{b}_2, \ldots$
  3. for each $\mathbf{b}_i$ do
     (1) $\mathbf{a}^{t+1} \leftarrow \mathcal{L}(\mathbf{b}_i)$
     (2) if $(t = R)$ then
           (i) save this trail to list
          (ii) return
     (3) else call TRAILS-FINDING($\mathbf{a}^{t+1}$, $t+1$)
```

After finding all the possible trails, we select a trail with maximal LCP value.

4.2 Mininum Number of test pairs

Here we want to know how many test pairs we need to attack for a given trail. Assume that the trail T has the maximal LCP value and its bias is ε. Now we want to know the mininum number of test pairs for this trail.

Let the corresponding linear approximation expression of T to be

$$P[p_1, p_2, \ldots, p_a] \oplus C[j_1, j_2, \ldots, j_b] = K[k_1, k_2, \ldots, k_c].$$

Let K^* be the real key and K' be the key we guessed. For each pair of plaintext P_i and ciphertext C_i, we define $\mathbf{I}_{i,K'} = P[p_1, p_2, \ldots, p_a] \oplus C[j_1, j_2, \ldots, j_b] \oplus K'[k_1, k_2, \ldots, k_c] \oplus 1$ to be a random variable.

Definition 4.1. *An experiment satisfied the following conditions is called a* binomial experiment*.*

1. *The experiment consists of a sequences of n tests, where n is fixed.*

2. *The tests are identical, and each of them can result*

Thus,

$$\mathbf{I}_{i,K'} \sim \text{Bin}(1, p), \tag{4.1}$$

where

$$p = \begin{cases} \frac{1}{2} + \varepsilon & , K' = K^*; \\ \frac{1}{2} & , K \neq K^*. \end{cases}$$

Suppose we need $\mathcal{N}_L$ pairs and let ${}_{K'} = \sum_{i=1}^{\mathcal{N}_L} \mathbf{I}_{i,K'}$ be a random variable. From equation (4.1), we know that ${}_{K'}$ is also a binomial random variable:

$${}_{K'} \sim \text{Bin}(\mathcal{N}_L, p), \tag{4.2}$$

where

$$p = \begin{cases} \frac{1}{2} + \varepsilon & , K' = K^*; \\ \frac{1}{2} & , K \neq K^*. \end{cases}$$

When $\mathcal{N}_L \to \infty$, we have

$$_{K'} \sim N(\mathcal{N}_L p, \mathcal{N}_L p(1-p)). \tag{4.3}$$

And then by *Central Limit Theorem*, we have the following equation:

$$\overline{}_{K'} = \frac{_{K'}}{\mathcal{N}_L} \sim N(p, \frac{p(1-p)}{\mathcal{N}_L}). \tag{4.4}$$

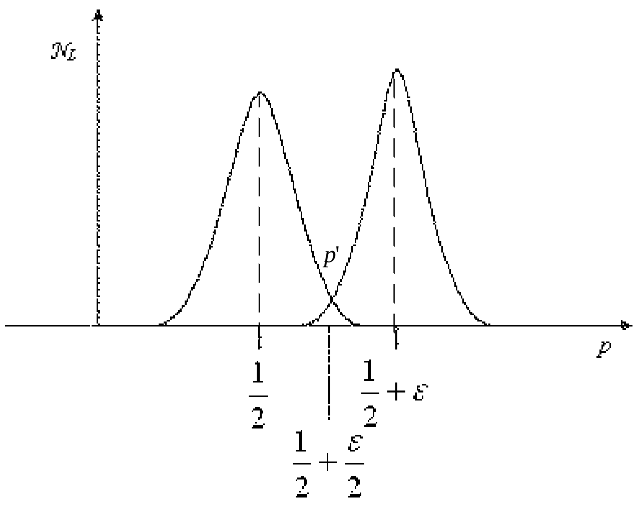


Figure 4.2: Two casess of distribution of $_{K'}$ with $\varepsilon > 0$

Figure 4.2 shows the distribution of $_{K'}$ with $\varepsilon > 0$. The left one shows the case when $K' \neq K^*$, and the right one shows another. In this figure, the two curves intersect at a point, denote p'. The the probability value of p' will be in the interval $(1/2, 1/2 + \varepsilon)$. If these two curves are the same, p' will be at the probability $1/2 + \varepsilon/2$.

In order to guess the key with heigher probability, we hope that the following two equations hold:

$$\Pr\{\overline{}_{K'} > \frac{1}{2} + \frac{\varepsilon}{2} \,|\, K' \neq K^*\} \leq \frac{\delta}{2}, \tag{4.5}$$

$$\Pr\{\overline{}_{K'} < \frac{1}{2} + \frac{\varepsilon}{2} \,|\, K' = K^*\} \leq \frac{\delta}{2}, \tag{4.6}$$

where $(1 - \delta)$ is the *success rate* of this attack using this trail.

Now we discuss the two cases separately.

Case 1: $K' \neq K^*$

From equation(4.4), we have

$$\overline{}_{K'} \sim N(\frac{1}{2}, \frac{1}{4\mathcal{N}_L}). \tag{4.7}$$

Applied into equation (4.5), it becomes

$$\begin{aligned}
&\Pr\{\overline{}_{K'} > \frac{1}{2} + \frac{\varepsilon}{2}\} \\
&= \Pr\{\overline{}_{K'} - \frac{1}{2} > \frac{\varepsilon}{2}\} \\
&= \Pr\{\frac{\overline{}_{K'} - \frac{1}{2}}{\sqrt{\frac{1}{4\mathcal{N}_L}}} > \frac{\frac{\varepsilon}{2}}{\sqrt{\frac{1}{4\mathcal{N}_L}}}\} \\
&= 1 - \Phi(\varepsilon\sqrt{\mathcal{N}_L}) \leq \frac{\delta}{2}.
\end{aligned} \tag{4.8}$$

If we want the success rate to be 95%, that is, $\delta = 0.05$, the equation (4.8) will become

$$\begin{aligned}
&1 - \Phi(\varepsilon\sqrt{\mathcal{N}_L}) \leq \frac{\delta}{2} = 0.025 \\
&\Rightarrow \Phi(\varepsilon\sqrt{\mathcal{N}_L}) \geq 0.975 = \Phi(1.96) \\
&\Rightarrow \varepsilon\sqrt{\mathcal{N}_L} \geq 1.96 \\
&\Rightarrow \varepsilon\sqrt{\mathcal{N}_L} \quad 2 \\
&\Rightarrow \mathcal{N}_L \quad 4\varepsilon^{-2}
\end{aligned} \tag{4.9}$$

If we set the success rate to be 99.9%, i.e., $\delta = 0.001$, then (4.8) will be

$$\begin{aligned}
&1 - \Phi(\varepsilon\sqrt{\mathcal{N}_L}) \leq \frac{\delta}{2} = 0.0005 \\
&\Rightarrow \Phi(\varepsilon\sqrt{\mathcal{N}_L}) \geq 0.9995 = \Phi(1.96) \\
&\Rightarrow \varepsilon\sqrt{\mathcal{N}_L} \geq 3.27 \\
&\Rightarrow \varepsilon\sqrt{\mathcal{N}_L} \quad 4 \\
&\Rightarrow \mathcal{N}_L \quad 16\varepsilon^{-2}
\end{aligned} \tag{4.10}$$

Case 2: $K' = K^*$

Similar to Case 1, we have

$$\overline{}_{K'} \sim N(\frac{1}{2} + \varepsilon, \frac{1 - 4\varepsilon^2}{4\mathcal{N}_L}). \tag{4.11}$$

Applied into equation (4.6), it will be

$$\begin{aligned}
&\Pr\{\overline{}_{K'} < \frac{1}{2} + \frac{\varepsilon}{2}\} \\
&= \Pr\{\overline{}_{K'} - \frac{1}{2} - \varepsilon > -\frac{\varepsilon}{2}\} \\
&= \Pr\{\frac{\overline{}_{K'} - \frac{1}{2} - \varepsilon}{\sqrt{\frac{1 - 4\varepsilon^2}{4\mathcal{N}_L}}} > \frac{-\varepsilon\sqrt{\mathcal{N}_L}}{\sqrt{1 - 4\varepsilon^2}}\} \\
&= \Phi(-\varepsilon\sqrt{\frac{\mathcal{N}_L}{1 - 4\varepsilon^2}}) \leq \frac{\delta}{2}.
\end{aligned} \tag{4.12}$$

$$\Rightarrow \Phi(-\varepsilon\sqrt{\frac{\mathcal{N}_L}{1-4\varepsilon^2}}) \leq \begin{cases} 0.025 & ,\delta = 0.05 \\ 0.0005 & ,\delta = 0.001 \end{cases}$$

$$\Rightarrow \varepsilon\sqrt{\frac{\mathcal{N}_L}{1-4\varepsilon^2}} \geq \begin{cases} 1.96 & ,\delta = 0.05 \\ 3.27 & ,\delta = 0.001 \end{cases}$$

$$\Rightarrow \varepsilon^2\frac{\mathcal{N}_L}{1-4\varepsilon^2} \quad \begin{cases} 4 & ,\delta = 0.05 \\ 16 & ,\delta = 0.001 \end{cases}$$

$$\Rightarrow \mathcal{N}_L \quad 4\varepsilon^{-2} - 16, \delta = 0.05 \tag{4.13}$$

$$\mathcal{N}_L \quad 16\varepsilon^{-2} - 64, \delta = 0.001 \tag{4.14}$$

From the equations (4.9), (4.10), (4.13), and (4.14), we can see that the results in Case 1 and 2 are very close. Table 4.1 is the success rate of Matsui's Algorithm, and the results are also similar to ours. Therefore, we will set $\mathcal{N}_L = 16\varepsilon^{-2}$.

$\mathcal{N}_L$	$2\varepsilon^{-2}$	$4\varepsilon^{-2}$	$8\varepsilon^{-2}$	$16\varepsilon^{-2}$
Success Rate	48.6%	78.5%	96.7%	99.9%

Table 4.1: The success rate of Matsui's Algorithm 2 ([6])

4.3 Attack Strategies

4.3.1 Basic Strategy

According the Matsui's Algorithm 2 (in Section 3.1), our basic idea of linear cryptanalysis is to guess one of the round keys, with a trail with maximal linear probability value. There are two ways for us to start the attack: we can guess the round keys from the first one, K^1, or the last one, K^{R+1}.

Attack from K^1 to K^{R+1}

Figure 4.3 shows the concept of this strategy. We first guess the first round key $K^1 = K'$, and them encrypt each plaintext P_i to X_i through round 1. Next we use the input and output masks, **a** and **b**, of the trail with maximal linear probability to check if the expression $\mathbf{a} \bullet X_i = \mathbf{b} \bullet C_i$ holds. If it holds, then we increase the counter of K', $Counter[K']$. In the final step, we choose the K' which maximizes $|Counter[K'] - \mathcal{N}_L/2|$. Once K^1 is known, round 1 can be stripped off, and linear cryptanalysis can be reapplied to obtain K^2, adn so on, until sll subkeys are known.

Algorithm 4.2 gives the detail steps of this strategy. Note that we denote T be the set of test pairs.

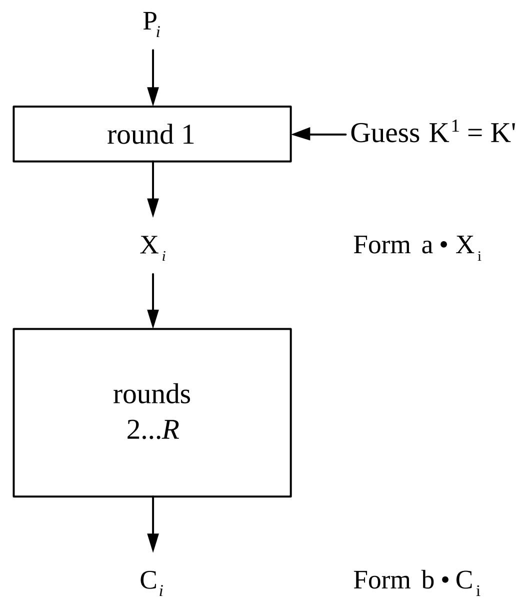


Figure 4.3: Attack strategy: from K^1 to K^{R+1} (from [4])

Algorithm 4.2. STRATEGY1($T, \mathcal{N}_L$)

```
1. for each possible round key $K'$ do
        $Counter[K'] \leftarrow 0$
2. for each test pair $\langle P_i, C_i \rangle \in T$ do
    (1) for each possible round key $K'$ do
        (i) Encrypt $P_i$ through round 1 to obtain $X_i$
        (ii) if $(\mathbf{a} \bullet X_i = \mathbf{b} \bullet C_i)$ then
                $Counter[K'] \leftarrow Counter[K'] + 1$
3. $max \leftarrow -1$
4. for each possible round key $K'$ do
    (1) $Counter[K'] \leftarrow |Counter[K'] - \mathcal{N}_L/2|$
    (2) if $(Counter[K'] > max)$ then
        (i) $max \leftarrow Counter[K']$
        (ii) $maxkey \leftarrow K'$
5. output($maxkey$)
```

Attack from K^{R+1} to K^1

Figure 4.4 is the concept of this strategy. This one is similar to previous one, except that we start from the last round key, K^{R+1}. The algotithm below shows the details of this

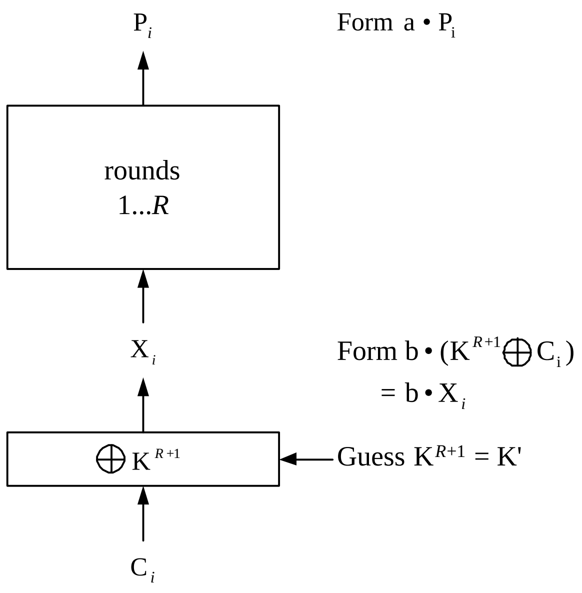


Figure 4.4: Attack strategy: from K^{R+1} to K^1 (from [5])

strategy.

Algorithm 4.3. STRATEGY2($T, \mathcal{N}_L$)

```
1. for each possible round key $K'$ do
       $Counter[K'] \leftarrow 0$
2. for each test pair $\langle P_i, C_i \rangle \in T$ do
   (1) for each possible round key $K'$ do
       (i) $X_i \leftarrow C_i \oplus K'$
       (ii) if $(\mathbf{a} \bullet P_i = \mathbf{b} \bullet X_i)$ then
               $Counter[K'] \leftarrow Counter[K'] + 1$
3. $max \leftarrow -1$
4. for each possible round key $K'$ do
   (1) $Counter[K'] \leftarrow |Counter[K'] - \mathcal{N}_L/2|$
   (2) if $(Counter[K'] > max)$ then
       (i) $max \leftarrow Counter[K']$
       (ii) $maxkey \leftarrow K'$
5. output($maxkey$)
```

4.3.2 Using Multiple Linear Approximations

As we described in Section 3.5.2, we need to find two or more trails that attack the same key bits when using multiple linear approximations. We choose the trails which their input masks $\mathbf{a}_i$ (or output masks $\mathbf{b}_i$) are the same. We use them because they attack the same key bits in K^1 (or K^{R+1} if we choose the same output masks). For each trail T_i with bias ε_i, we assign a weight $a_i = \varepsilon_i / \sum \varepsilon_i$. And the counters $Counter[K']$ are increased by a_i insdeed of 1. We also choose the key bits which is the farthest from $\mathcal{N}_L/2$.

The algotirhm shows below. Note that we denote T_L be the set of the trails we choose.

```
Algorithm 4.4.   MLA($T$, $\mathcal{N}_L$, $T_L$)
  1. for each possible round key $K'$ do
         $Counter[K'] \leftarrow 0$
  2. for each test pair $\langle P_i, C_i \rangle \in T$ do
     (1) for each possible round key $K'$ do
         (i) for each trail $T_i \in T_L$ do
             (a) $a_i \leftarrow \varepsilon_i / \sum_{i=1}^{n} \varepsilon_i$
             (b) Encrypt $P_i$ through round 1 to obtain $X_i$
             (c) if $(\mathbf{a} \bullet X_i = \mathbf{b} \bullet C_i)$ then
                     $Counter[K'] \leftarrow Counter[K'] + a_i$
  3. $max \leftarrow -1$
  4. for each possible round key $K'$ do
     (1) $Counter[K'] \leftarrow |Counter[K'] - \mathcal{N}_L/2|$
     (2) if $(Counter[K'] > max)$ then
          (i) $max \leftarrow Counter[K']$
         (ii) $maxkey \leftarrow K'$
  5. output($maxkey$)
```

4.4 Backtracking

In Section 3.5.1, we introduced the idea of candidate keys. Suppose we store r possible candidate keys whose counters are close to $\mathcal{N}_L/2$, where r is a flexible parameter could be modified. At the end, if the key we extracted tested to be wrong, we can go back to choose another candidate subkey systematically. Figure 4.5 shows this backtracking scheme. If we run out all possible candidate keys, then we declare this attack failed.

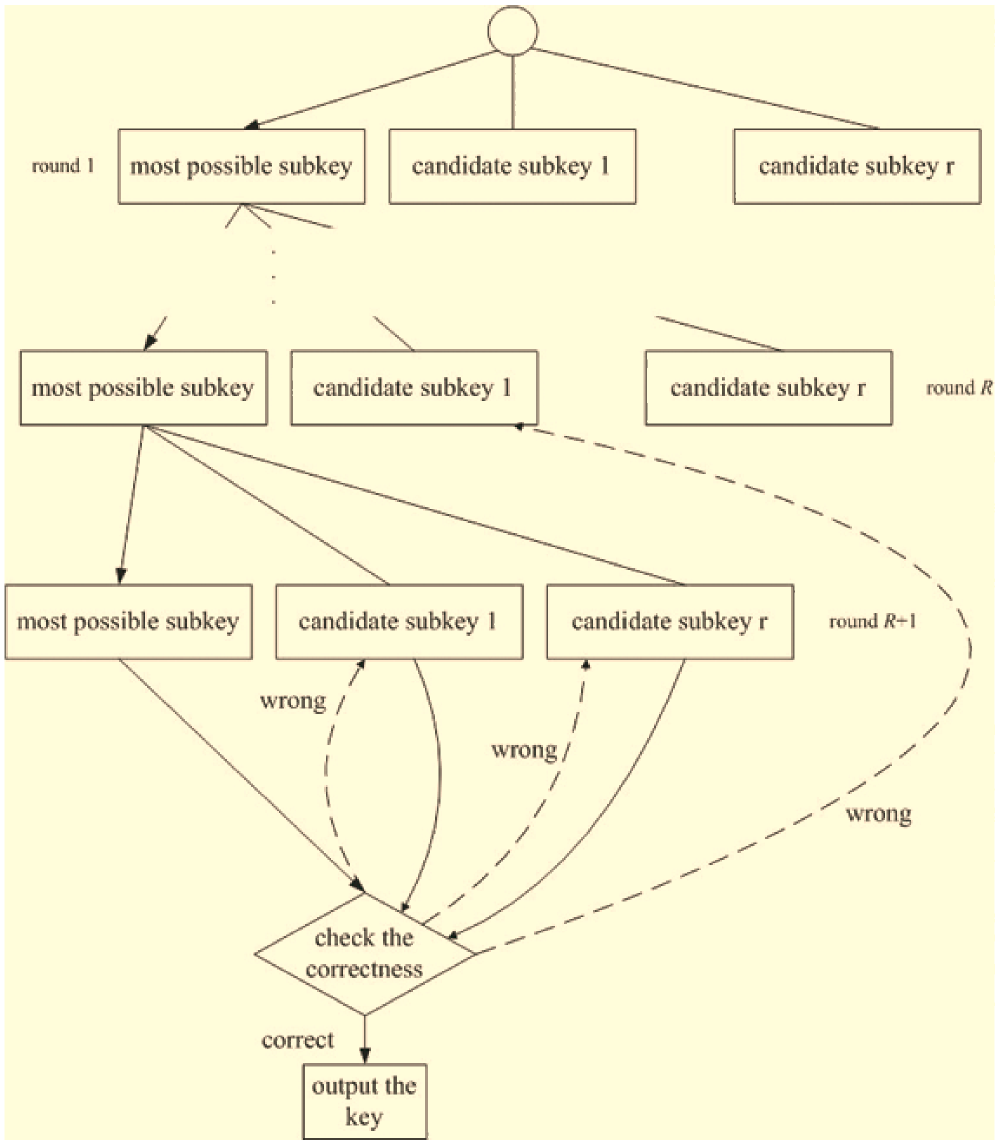


Figure 4.5: Backtracking Scheme

Chapter 5

Computer Experiments

Our experiment is to attack a simple SPN. The structure of this SPN is showed in Figure 5.1. It is a 4-round SPN with block size $N = 16$, s-box size $n = 4$, key length $N_K = 32$, and 4 s-boxes in a round. Table 5.1 shows the s-box and the permutation mappings. The subkey $K^t = K[t+1, \ldots, t+16]$.

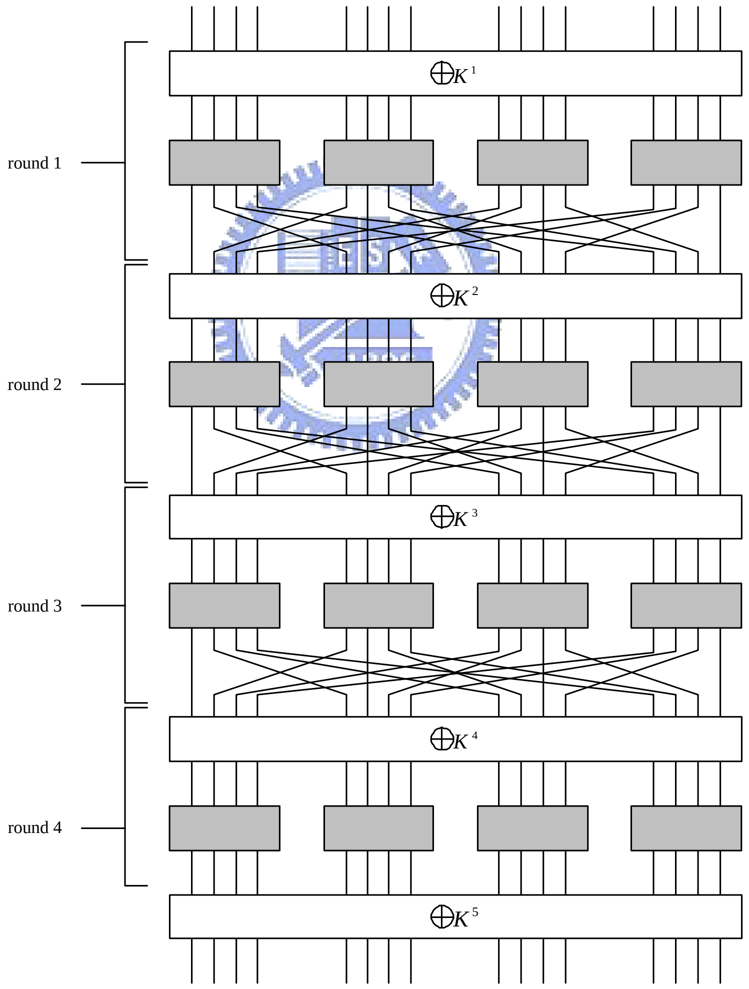


Figure 5.1: The SPN we will attack

We run the experiment on Pentium III 733 CPU with 256MB RAM under FreeBSD using C language. We simulate 100 tests with different keys. Each test we randomly

X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S(X)$	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7

X	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\mathcal{L}(X)$	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

Table 5.1: S-box and permutation mappings

choose around $8 \times \varepsilon^{-2}(= 16/LP(\mathbf{a}, \mathbf{b}))$ test pairs (data complexity). We set the number of candidate keys to be 1 to 10 for backtracking. The following table shows the difference between the two strategies.

Strategy	Data Complexity ($\mathcal{N}_L$)	Time Complexity (Average)
Basic (From K^1 to K^{R+1})	2272	1.79 mins
Muliple Linear Approximations	2272	8.11 mins

Table 5.2: Data complexity and time complexity

Number of Candidate Subkeys	1	2	3	4	5	6	7	8	9	10
Basic (From K^1 to K^{R+1})	6	12	22	34	44	52	62	70	78	96
Muliple Linear Approximations	8	20	27	36	46	55	65	72	82	98

Table 5.3: Success rate

Chapter 6

Conclusion

In this thesis, we discussed about the linear cryptanalysis on a simple substitution-permutation networks. T apply ilnear cryptanalysis, we first analyze the linear approximation of the s-box. From the table we know which input and output masks of the s-box act in a nonrandom fashion. Then we combine several s-boxes from their input and output masks to form a trail through the entire SPN. The characteristic of the trail tells us the linear characteristic probability and the linear expression of this trail. Then we start to attack the key bits by partially encrypting (decrypting) the test pairs, and count the number that the expression holds. At the end we choose the key bits whose counter is farthest from $\mathcal{N}_L/2$.

We then propose the attack algorithm of the SPN in Chapter 4. We first describe how to find the trails systematically, and set the threshold value to limit the number of the trails. Next we use the techniques in Statistics to compute the approximate value of $\mathcal{N}_L$. We then know that how many pairs de we need to achieve the success rate we want. Then we propose two kinds of strategies to attack SPN. The first one is from the basic idea of linear cryptanalysis. We give two versions of this strategy: from K^1 and from K^{R+1}. The other one we use the technique called multiple linear approximation. We use more then one trails that attack the same key bits to increase the success rate of the attack. Final we introduce the backtracking strategy. We store r candidate subkeys to improved our success rate. In Chapter 5, we try to attack a simple SPN and we show the data complexity, time complexity, and the success rate under different number of candidate subkeys.

In the future, we think there are two way to study. First one is to compue the number of test pairs, $\mathcal{N}_L$, more precisely. We need to analyze the distribution of $\mathcal{N}_L$, and try to find a formula to compute $\mathcal{N}_L$ easily. Second, we can compute the number of candidate subkeys, r. From the distribution of $\mathcal{N}_L$, we can estimate the probability that the real key lies in the first r candidates. We want to know the value r such that the probability is not too small. We hope that r will be not too large under some conditions in order to increase the success rate of the attack.

Bibliography

[1] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography.* CRC Press, 1997, ISBN: 0849385237.

[2] L. T. Keliher, "Linear cryptanalysis of substitution-permutation networks," Ph.D. dissertation, Queen's University, Kinston, Ontario, Canada, October 2003.

[3] J. B. Kam and G. I. Davida, "Structured design of substitution-permutation encryption networks," *IEEE Transactions on Computers*, vol. 28, no. 10, pp. 747–753, 1979.

[4] L. Keliher, H. Meijer, and S. Tavares, "New method for upper bounding the maximum average linear hull probability for SPNs," in *Advances in Cryptology - EUROCRYPT 2001*, ser. Lecture Notes in Computer Science, B. Pfitzmann, Ed., vol. 2045. Springer-Verlag, 2001, pp. 420–436.

[5] D. R. Stinson, *Cryptography: Theory and Practice*, 2nd ed. Chapman & Hall/CRC, 2002, ISBN: 1-58488-206-9.

[6] M. Matsui, "Linear cryptanalysis method for DES cipher," in *Advances in Cryptology - EUROCRYPT '93*, ser. Lecture Notes in Computer Science, T. Helleseth, Ed., vol. 765. Springer-Verlag, 1994, pp. 386–397.

[7] M. Matsui and A. Yamagishi, "A new method for known plaintext attack of FEAL cipher," in *Advances in Cryptology - EUROCRYPT '92*, ser. Lecture Notes in Computer Science, R. A. Rueppel, Ed., vol. 658. Springer-Verlag, 1993, pp. 81–91.

[8] A. Shimizu and S. Miyaguchi, "Fast data encipherment algorithm FEAL," in *Advances in Cryptology - EUROCRYPT '87*, ser. Lecture Notes in Computer Science, D. Chaum and W. L. Price, Eds., vol. 304. Springer-Verlag, 1988, pp. 267–280.

[9] FIPS 46, *Data Encryption Standard.* Federal Information Processing Standards Publication 46, U.S. Department of Commerce, NationalBureau of Standards, National Technical Information Service, Springfield, Virginia, 1977.

[10] M. Matsui, "The first experimental cryptanalysis of the Data Encryption Standard," in *Advances in Cryptology - CRYPTO '94*, ser. Lecture Notes in Computer Science, Y. G. Desmedt, Ed., vol. 839. Springer-Verlag, 1994, pp. 1–11.

[11] B. S. Kaliski Jr. and M. J. B. Robshaw, "Linear cryptanalysis using multiple approximations," in *Advances in Cryptology - CRYPTO '94*, ser. Lecture Notes in Computer Science, Y. G. Desmedt, Ed., vol. 839. Springer-Verlag, 1994, pp. 26–39.

[12] K. Nyberg, "Linear approximation of block ciphers," in *Advances in Cryptology - EUROCRYPT '94*, ser. Lecture Notes in Computer Science, A. De Santis, Ed., vol. 950. Springer-Verlag, 1995, pp. 439–444.

[13] W. Stallings, *Cryptography and Network Security: Priciples and Practice*, 2nd ed. Prentice Hall, 1998, ISBN: 0-13-869017-0.

[14] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, pp. 644–654, November 1976.

[15] FIPS 81, *DES Modes of Operation*. Federal Information Processing Standards Publication 81, U.S. Department of Commerce, NationalBureau of Standards, National Technical Information Service, Springfield, Virginia, 1980.

[16] C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.

[17] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd ed. John Wiley and Sons, 1996, ISBN: 0-471-12845-7.

[18] H. Feistel, "Cryptography and computer privacy," *Scientific American*, vol. 228, no. 5, pp. 15–23, 1973.

[19] H. Feistel, W. A. Notz, and J. L. Smith, "Some cryptographic techniques for machine to machine data communications," *Proceedings of the IEEE*, vol. 63, no. 11, pp. 1545–1554, 1975.

[20] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, "Camellia: A 128-bit block cipher suitable for multiple platforms — design and analysis," in *SAC 2000*, ser. Lecture Notes in Computer Science, D. R. Stinson and S. E. Tavares, Eds., vol. 2012. Springer-Verlag, 2001, pp. 39–56.

[21] M. Kanda, "Practical security evaluation against differential and linear cryptanalyses for Feistel ciphers with SPN round function," in *SAC 2000*, ser. Lecture Notes in Computer Science, D. R. Stinson and S. E. Tavares, Eds., vol. 2012. Springer-Verlag, 2001, pp. 324–338.

[22] M. Kanda, Y. Takashima, T. Matsumoto, K. Aoki, and K. Ohta, "A strategy for constructing fast round functions with practical security against differential and linear cryptanalysis," in *SAC '98*, ser. Lecture Notes in Computer Science, S. E. Tavares and H. Meijer, Eds., vol. 1556. Springer-Verlag, 1999, pp. 264–279.

[23] M. Kanda, S. Moriai, K. Aoki, H. Ueda, M. Ohkubo, Y. Takasima, K. Ohta, and T. Matsumoto, "E2 — a candidate cipher for AES," *The First Advanced Encryption Standard Candidate Conference, Proceedings, Ventura, California*, August 1998.

[24] B. Schneier and J. Kelsey, "Unbalanced Feistel networks and block cipher design," in *FSE*, ser. Lecture Notes in Computer Science, D. Gollmann, Ed., vol. 1039. Springer-Verlag, 1996, pp. 121–144.

[25] C. M. Adams, "The CAST-256 encryption algorithm," *The First Advanced Encryption Standard Candidate Conference, Proceedings, Ventura, California*, August 1998.

[26] C. Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S. M. Matyas Jr., L. O'Connor, M. Peyravian, D. Stafford, and N. Zunic, "MARS — a candidate cipher for AES," *The First Advanced Encryption Standard Candidate Conference, Proceedings, Ventura, California*, August 1998.

[27] X. Lai, *On the Design and Security of Block Ciphers, vol. 1 of ETH Series in Informatino Processing.* Hartung-Gorre Verlag Konstanz, 1992, ISBN: 3-89191-573-X.

[28] X. Lai and J. L. Massey, "A proposal for a new block encryption standard," in *Advances in Cryptology - EUROCRYPT '90*, ser. Lecture Notes in Computer Science, I. Damgård, Ed., vol. 473. Springer-Verlag, 1990, pp. 389–404.

[29] X. Lai, J. L. Massey, and S. Murphy, "Markov ciphers and differential cryptanalysis," in *Advances in Cryptology - EUROCRYPT '91*, ser. Lecture Notes in Computer Science, D. W. Davies, Ed., vol. 547. Springer-Verlag, 1991, pp. 17–38.

[30] R. L. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin, "The RC6 block cipher," *The First Advanced Encryption Standard Candidate Conference, Proceedings, Ventura, California*, August 1998.

[31] A. Sorkin, "Luciher, a cryptographic algorithm," *Cryptologia*, vol. 8, no. 1, pp. 22–41, 1984.

[32] W. Diffie and M. E. Hellman, "Exhaustive cryptanalysis of the NBS Data Encryption Standard," *Computer*, vol. 10, no. 6, pp. 74–84, June 1977.

[33] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard.* Springer-Verlag, 1993, ISBN: 0-387-97930-1.

[34] L. R. Knudsen, "Truncated and higher order differentials," in *Fast Software Encryption*, ser. Lecture Notes in Computer Science, B. Preneel, Ed., vol. 1008. Springer-Verlag, 1994, pp. 196–211.

[35] S. K. Langford and M. E. Hellman, "Differential-linear cryptanalysis," in *Advances in Cryptology - CRYPTO '94*, ser. Lecture Notes in Computer Science, Y. G. Desmedt, Ed., vol. 839. Springer-Verlag, 1994, pp. 17–25.

[36] S. Vaudenay, "An experiment on DES statistical cryptanalysis," in *ACM Conference on Computer and Communications Security*, 1996, pp. 139–147.

[37] C. M. Adams, "Constructing symmetric ciphers using the CAST design procedure," *Designs, Codes and Cryptography*, vol. 12, no. 3, pp. 283–316, 1997.

[38] L. Brown, J. Pieprzyk, and J. Seberry, "LOKI — a cryptographic primitive for authentication and secrecy applications," in *Advances in Cryptology - AUSCRYPT '90*, ser. Lecture Notes in Computer Science, J. Seberry and J. Pieprzyk, Eds., vol. 453. Springer-Verlag, 1990, pp. 229–236.

[39] J. Kilian and P. Rogaway, "How to protect des against exhaustive key search," in *Advances in Cryptology - CRYPTO '96*, ser. Lecture Notes in Computer Science, N. Koblitz, Ed., vol. 1109. Springer-Verlag, 1996, pp. 252–267.

[40] National Institute of Standards and Technology, "Announcing request for candidate algorithm nominations for the Advanced Encryption Standard (AES)," *Federal Register*, vol. 62, no. 177, pp. 48 051–48 058, September 1997.

[41] J. Daemen and V. Rijmen, "AES proposal: Rijndal," http://www.iaik.tu-graz.ac.at/research/krypto/AES/old/˜rijmen/rijndael/, 1999.

[42] ——, *The Design of Rijndael: AES — The Advanced Encryption Standard.* Springer-Verlag, 2002, ISBN: 3540425802.

[43] R. Anderson, E. Biham, and L. Knudsen, "Serpent: A flexible block cipher with maximum assurance," http://www.cl.cam.ac.uk/˜rja14/serpent.html, 1998.

[44] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, "Twofish: A 128-bit block cipher," http://www.schneier.com/paper-twofish-paper.html, June 1998.

[45] FIPS 197, *Advanced Encryption Standard (AES).* Federal Information Processing Standards Publication 197, U.S. Department of Commerce, NationalBureau of Standards, National Technical Information Service, Springfield, Virginia, 2001.

[46] L. R. Knudsen, "Block ciphers — analysis, design and applications," Ph.D. dissertation, University of Aarhus, Aarhus, Denmark, July 1994.

[47] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," in *Advances in Cryptology - CRYPTO '90*, ser. Lecture Notes in Computer Science, A. Menezes and S. A. Vanstone, Eds., vol. 537. Springer-Verlag, 1990, pp. 2–21.

[48] ——, “Differential cryptanalysis of DES-like cryptosystems,” *Journal of Cryptology*, vol. 4, no. 1, pp. 3–72, 1991.

[49] ——, “Differential cryptanalysis of the full 16-round DES,” in *Advances in Cryptology - CRYPTO ’92*, ser. Lecture Notes in Computer Science, E. F. Brickell, Ed., vol. 740. Springer-Verlag, 1992, pp. 487–496.

[50] S. Murphy, “The cryptanalysis of FEAL-4 with 20 chosen plaintexts,” *Journal of Cryptology*, vol. 2, no. 3, pp. 145–154, 1990.

[51] S. Vaudenay, “On the security of CS-Cipher,” in *FSE ’99*, ser. Lecture Notes in Computer Science, L. R. Knudsen, Ed., vol. 1636. Springer-Verlag, 1999, pp. 260–274.

[52] W. Meier and O. Staffelbach, “Nonlinearity criteria for cryptographic functions,” in *Advances in Cryptology - EUROCRYPT ’89*, ser. Lecture Notes in Computer Science, J.-J. Quisquater and J. Vandewalle, Eds., vol. 434. Springer-Verlag, 1989, pp. 549–562.

[53] C. Harpes, G. G. Kramer, and J. L. Massey, “A generalization of linear cryptanalysis and the applicability of Matsui’s Piling-up Lemma,” in *Advances in Cryptology - EUROCRYPT ’95*, ser. Lecture Notes in Computer Science, L. C. Guillou and J.-J. Quisquater, Eds., vol. 921. Springer-Verlag, 1995, pp. 24–38.

[54] Y. G. Desmedt, Ed., *Advances in Cryptology - CRYPTO ’94: 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 1994. Proceedings*, ser. Lecture Notes in Computer Science, vol. 839. Springer-Verlag, 1994.

[55] D. R. Stinson and S. E. Tavares, Eds., *Selected Areas in Cryptography: 7th Annual International Workshop, SAC 2000, Waterloo, Ontario, Canada, August 14–15, 2000, Proceedings:*, ser. Lecture Notes in Computer Science, vol. 2012. Springer-Verlag, 2001.