A Study on the Construction and Analysis of Substitution Boxes for Symmetric Cryptosystems

A Dissertation
Submitted to the Division of Electrical and Computer Engineering
and my Advisory Professors of Yokohama National University
In Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy

by
Kwangjo KIM
December 25, 1990

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Chief Advisor Professor Hideki IMAI

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Professor Yasunori DOHI

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Professor Rokuya ISHII

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Associate Professor Ryuji KOHNO

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Associate Professor Tsutomu MATSUMOTO

Abstract

S(ubstitution)-boxes are quite important components of modern symmetric cryptosystems (in particular, block ciphers) in the sense that S-boxes bring nonlinearity to block ciphers and strengthen their cryptographic security. An S-box is said to satisfy the strict avalanche criterion (SAC), if and only if for any single input bit of the S-box, the inversion of it changes each output bit with probability one half.

In this thesis, with the concrete proof of cryptographical properties of S-boxes satisfying the SAC, we propose a variety of provable construction methods for S-boxes satisfying the SAC. For Boolean S-boxes satisfying the SAC, we can construct and enlarge them by using concatenation, Kronecker (or direct) product, and dyadic shift. For bijective S-boxes satisfying the SAC, when an n-bit input Boolean function and an n-bit input bijective function satisfying the SAC are given, the combined function is proved to become an $(n+1)$-bit bijective function satisfying the SAC as well. Also, we propose one simple construction method to construct bijective functions satisfying the maximum order SAC.

Until now, bent functions have been given great attention in coding theory, logic synthesis and spread spectrum communications. We show that there exists an interesting relationship between bent functions and Boolean functions satisfying the (maximum order) SAC. All Boolean functions satisfying the maximum order SAC are always bent and all bent functions satisfy at least the 0-th order SAC.

For practical applications, we apply these cryptographically useful functions to construct DES-like S-boxes according to our design criteria including the possibility of differential attack. Compared with DES S-boxes, we found that our designed DES-like S-boxes exhibit better cyptographic properties than those of DES S-boxes. Moreover, as an experimental work of symmetric cryptosystems, we examine the statistical properties of DES-like cryptosystems (FEAL-4, FEAL-8, Multi2, DES, and s^2DES). s^2DES is the DES-like cryptosystem in which all the S-boxes of DES are replaced by our designed 8 DES-like S-boxes. These experiments again indicate to us that our designed DES-like S-boxes have good cryptographic performances.

Finally this thesis will aid us in designing and analyzing block cipher algorithms directly and stream cipher algorithms indirectly.

Acknowledgements

I would like to heartily thank my chief advisor, Professor Hideki Imai, for his guidance, encouragement, and support during my 3-year research work. I am also grateful to Professor Yasunori Dohi, Professor Rokuya Ishii, and Associate Professor Ryuji Kohno for their advice.

I would like to express my special thanks to Associate Professor Tsutomu Matsumoto for his useful suggestions and helpful comments.

I would like to acknowledge many foreign students in Imai Research Laboratory and, in particular, Mr. Manuel Cerecedo (from Spain) and Mr. Young Yoon (from Canada) for their proof reading of my first draft and correcting of its grammatical mistakes.

Special appreciation and thanks go to my wife, Jongsun Kang, for her patience and understanding throughout the course of this endeavor. Moreover, I feel thankful to my two lovely sons, 6-year-old Sunghak Kim and 4-year-old Jaehak Kim for growing up favorably.

Finally, financial support from the Japanese Ministry of Education (Monbusho) and my workplace in Korea, Electronics and Telecommunications Research Institute (ETRI), is gratefully acknowledged.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We are now living a so-called era of information society. The exchange of information by electronic means such as Facsimile, teletex, electronic mail, mobile telephones, *etc.* has been enormously increased in the last two decades.

When valuable or secret data are stored or transmitted, they are frequently protected physically through the use of safes, armed couriers, shielded cables, and the like. Today, however, such measures often become inapplicable, insufficient or uneconomical. Consequently, other appropriate and efficient techniques should be employed. Cryptography has evolved into a natural technique to solve such problems under the existing circumstances.

Cryptography is the use of transformation of data with the intent of making the data useless to one's opponents. Such transformations can solve two major problems of information security: the privacy problem whereby an opponent is prevented from extracting information from a communication channel, and the authentication problem whereby an opponent is prevented from injecting false data into the channel or altering messages so that their meaning is changed. A transformation can be simplified into the mathematical expression $C = F(K, P)$ where P is a plaintext, C is a ciphertext, and K is a key which determines whether or not a function F works uniquely. F is called hereinafter *cryptosystem*.

There are two categories of modern cryptosystems : *symmetric* and *asymmetric* cryptosystems. A *symmetric cryptosystem* (or conventional cryptosystem, one-key cryptosystem, common key cryptosystem) illustrated in Fig. (1.1) can be characterized by the fact that a sender and receiver have to agree upon a common secret key K before the en-

crypted communication can take place. This key must be shared through the secure channel between the two communication entities and be identical for encryption and decryption. If n communication entities are assumed to be in a network, each entity should record $\binom{n}{2}$ keys for secure communications. Thus, symmetric cryptosystems inevitably have a key management problem in a large communication network.

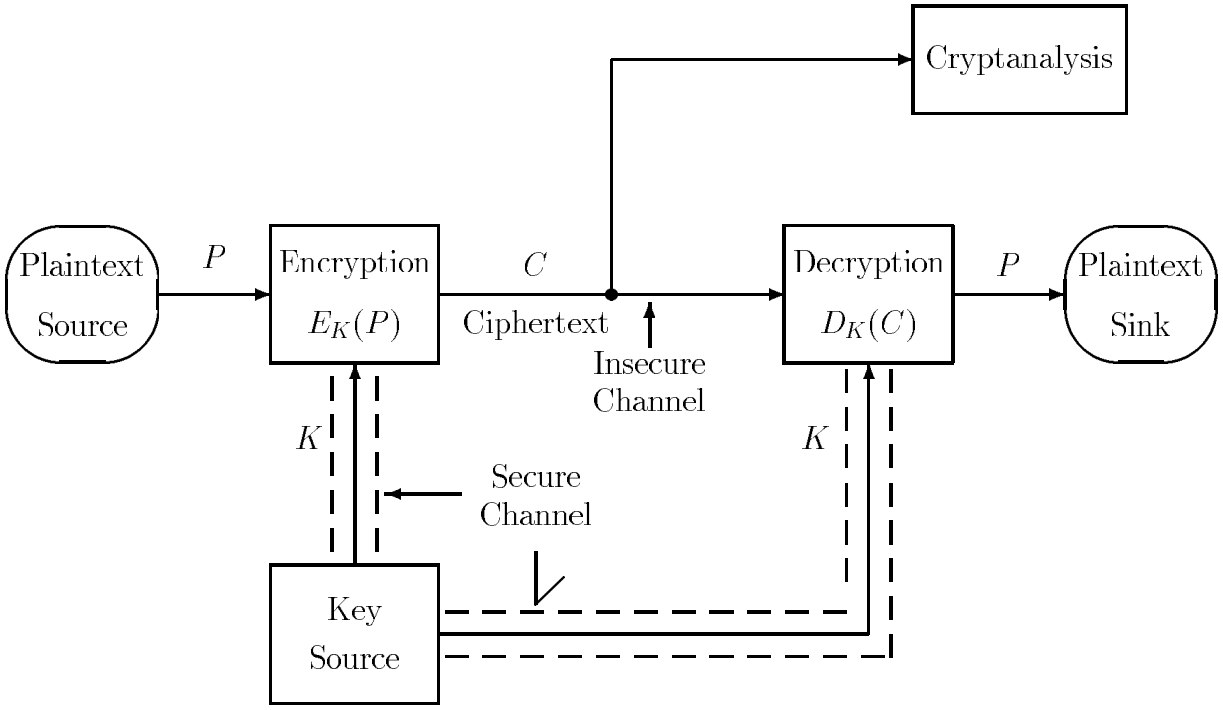

Figure 1.1: Systematic model of a symmetric cryptosystem

In an asymmetric cryptosystem (or public key cryptosystem, two-key cryptosystem), on the other hand, encryption is done with a public key whereas decryption uses a distinct secret key. Asymmetric cryptosystems use so-called trapdoor one-way functions. These functions are characterized by the fact that it is easy, given the public information key to compute the value of such a function for any argument, but for virtually all function values, it is extremely difficult to determine its inverse argument unless the trapdoor information is known. This thesis limits itself only to symmetric cryptosystems.

Let us briefly review the historical background of symmetric cryptosystems. In 1949, Shannon [69] proposed the outstanding notion of mixing transformation which randomly

distributes the meaningful messages uniformly over the set of all possible ciphertext messages. He formalized two basic transformations for symmetric cryptosystems. One is diffusion, according to which each bit of plaintext should influence many bits of ciphertext, so that the plaintext statistics are not recognizable in the ciphertext. The other is confusion which should make the interdependence of plaintext and ciphertext extremely difficult to analyze and describe. A practical mixing transformation can be implemented by alternatively applying permutation and substitution boxes.

In 1977, NBS (National Bureau of Standards) adopted "DES"(Data Encryption Standard) [17] which was proposed by IBM as a federal standard of commercial cryptosystem in the United States. NTT in 1987 proposed a new cryptosystem called "FEAL"(Fast Encryption ALgorithm) [52] whose purpose is to make encryption and decryption operation faster than DES when in the software or hardware implementation. FEAL has two types : FEAL-4 and FEAL-8 depending on the number of rounds. Moreover, Hitachi also proposed a symmetric cryptosystem called "Multi2" [76] in 1989. Also, Brown *et al* in Australia proposed one symmetric cryptosystem called "LOKI" [11] for banking securities in 1990.

It could be considered that all the above-mentioned symmetric cryptosystems are good practical design of the mixing transformation. We categorize these cryptosystems altogether as "DES-like cryptosystem". In DES-like cryptosystems, a substitution box (abbreviated as "S-box") is implemented by a logic circuit or a table lookup memory and a permutation is implemented by a one-to-one wiring. S-boxes bring nonlinearity to cryptosystems and strengthen their cryptographic security. This means that an S-box plays the most important role in DES-like cryptosystems.

In general, when we design a cryptosystem, we should consider the following basic problems:

- How can a cryptographer prove the security of a cryptosystem ?
- What kinds of statistical or mathematical structure(s) a cryptosystem has ?
- How can we check any property of a cryptosystem efficiently ?

At the cryptanalyst's side, for example, he(she) tries to cryptanalyze any cryptosystem by utilizing statistical or mathematical [26] structure(s) of a cryptosystem, *etc.*

Even if a cryptographer verifies the security of a cryptosystem, we have seen that some cryptosystems were broken [9] easily after their publication.

In principle, a cryptosystem can always be broken by an exhaustive key search. However, if the key set is large enough, such a search becomes computationally infeasible. Moreover, if the cryptosystem is not well designed, the key may be discovered with high probability by searching a much smaller set. Thus we consider that there is a need to perform as many statistical tests as possible to reveal any such weaknesses.

We will focus on a variety of design methodologies of S-boxes and their cryptographic significance and evaluate the goodness-of-fit of our designed S-boxes by substituting them into the DES S-boxes. The organization of this thesis is as follows:

In Chapter 2, we review the basic notions and operation modes of symmetric cryptosystems including block cipher and stream cipher. We summarize the cryptographic specifications of some published block cipher algorithms.

In Chapter 3, after suggesting the properties of S-box satisfying the Strict Avalanche Criterion (SAC) which means if and only if for any single input bit of the S-box, the inversion of it changes each output bit with probability one half, we propose a variety of provable construction methods for S-boxes satisfying the SAC. Also, we make clear the relationship between bent functions and Boolean functions satisfying the SAC. After classifying the S-boxes satisfying the SAC into balanced and unbalanced and unidirectional, bidirectional and self-bidirectional, we give how many functions exist under such classifications by computer search. In Chapter 4, we propose a structured methodology to construct DES-like S-boxes and evaluate their cryptographic significance by measuring their nonlinearity, dependence matrix, cross correlation of avalanche variables and pairs XOR distribution. In Chapter 5, as an experimental work of symmetric cryptosystem, we examine the statistical, quantitative and cyclic properties of DES-like cryptosystems including s^2DES. s^2DES is the DES-like cryptosystem in which all the S-boxes of DES are replaced by our designed S-boxes. The statistical properties of a cryptosystem are checked by partial input-output dependence test and randomness test including frequency test, serial test and runs test. For a quantitative measure of a cryptosystem, we perform binary derivative test and sequence complexity test of DES-like cryptosystems. We also examine the cyclic properties of DES-like cryptosystems when we reduce the original block size of cryptosystem into 16 bits, since it can be tractable

by a small computational-power machine. Finally, we state the conclusions of this thesis and suggest some open problems.

Chapter 2

An Overview of Symmetric Cryptosystems

Today's symmetric cryptosystem can be split up into two families : *block cipher* and *stream cipher*.

2.1 Block Cipher

In a block cipher, the plaintext is broken up into blocks of fixed length. Each plaintext block is mapped into a ciphertext block of the same length by means of a key-dependent substitution. As mentioned in Chapter 1, there are many published symmetric cryptosystems. Since DES is the most famous symmetric cryptosystem, we will briefly describe the internal structure of DES.

DES enciphers 64-bit blocks of data with a 56-bit key. The algorithm—which is used both to encipher and decipher—is illustrated in Fig. (2.1). A plaintext block P is first transposed under an initial permutation IP, giving $P_0 = IP(P)$. After it has passed through 16 iterations of a function f, it is transposed under the inverse permutation IP^{-1} to give the final result.

Between the initial and final transpositions, the algorithm performs 16 iterations of a function f that combines substitution and transposition. Let P_i denote the result of the i-th iteration, and let L_i and R_i denote the left and right halves of P_i respectively, *i.e.*, $T_i = L_i R_i$, where $L_i = t_1 t_2 \ldots t_{32}, R_i = t_{33} t_{34} \ldots t_{64}$ and t_k $(1 \leq k \leq 64)$ denotes the k-th bit of L_i or R_i. Then,

$$L_i \;=\; R_{i-1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

where K_i is a 48-bit round key produced by key scheduling. For the details of key scheduling of DES, refer to [17]. Note that after the last iteration, the left and right halves are not exchanged; instead the concatenated block $R_{16}L_{16}$ is input to the final permutation. This is necessary in order that the algorithm can be used both to encipher and decipher.

Fig. (2.2) shows a sketch of the function $f(R_{i-1}, K_i)$. First, R_{i-1} is expanded to a 48-bit block $E(R_{i-1})$ using the bit-selection table E. Next, the exclusive-or of $E(R_{i-1})$ and K_i is calculated and the result is broken into eight 6-bit blocks $B_1, B_2, \ldots, B_8$, *i.e.*,

$$E(R_{i-1}) \oplus K_i = B_1 B_2 \ldots B_8.$$

Each 6-bit block B_i is then used as input to a S-box S_i, which returns a 4-bit block $S_i(B_i)$. These blocks are concatenated together, and the resulting 32-bit block is transposed by the permutation P. Thus, the block returned by $f(R_{i-1}, K_i)$ is

$$P(S_1(B_1)S_2(B_2)\ldots S_8(B_8)).$$

Each S_j maps a 6-bit block $B_i = b_1 b_2 b_3 b_4 b_5 b_6$ into a 4-bit block as shown in **Appendix A.** This is done as follows: the integer corresponding to $b_1 b_6$ selects a row in the table, while the integer corresponding to $b_2 b_3 b_4 b_5$ selects a column. The value $S_i(B_i)$ is then a 4-bit representation of the integer in that row and column.

Decryption is performed using the same algorithm, except that K_{16} is used in the first iteration, K_{15} in the second, and so on, with K_1 used in the 16-th iteration. This is because the final permutation IP^{-1} is the inverse of the initial permutation IP, and

$$\begin{aligned} R_{i-1} &= L_i \\ L_{i-1} &= R_i \oplus f(L_i, K_i). \end{aligned}$$

Note that whereas the order of the keys is reversed, the algorithm itself is not.

Other published symmetric cryptosystems have a similar structure. However, the specific implementations of a f function are a little different. For example, the f function of FEAL is implemented by modular addition and 2-bit rotation operation and the f function of Multi2 by modular addition, subtraction and k-bit($k = 1, 2, 4, 8, 16$) rotation

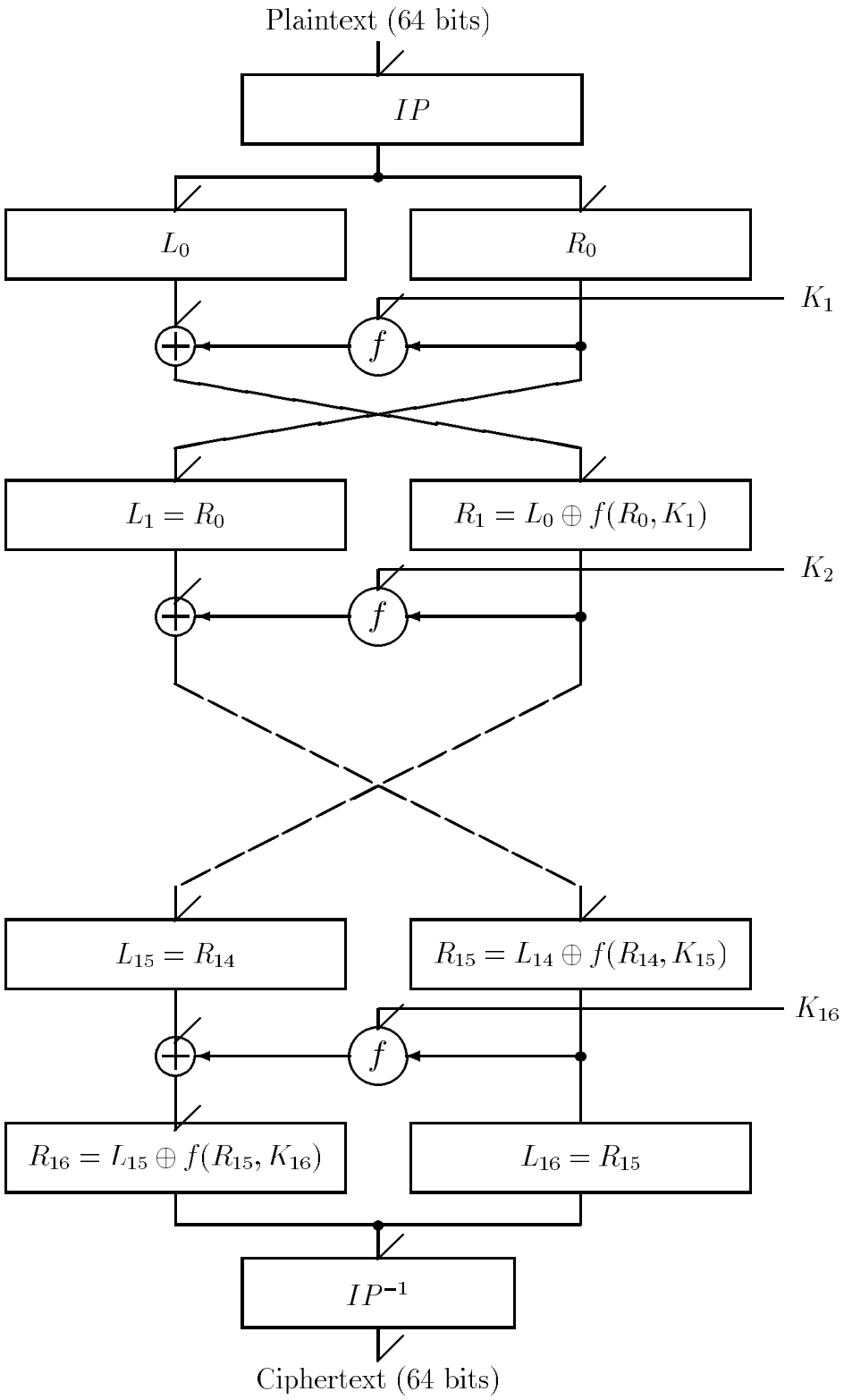

Figure 2.1: DES enciphering algorithm

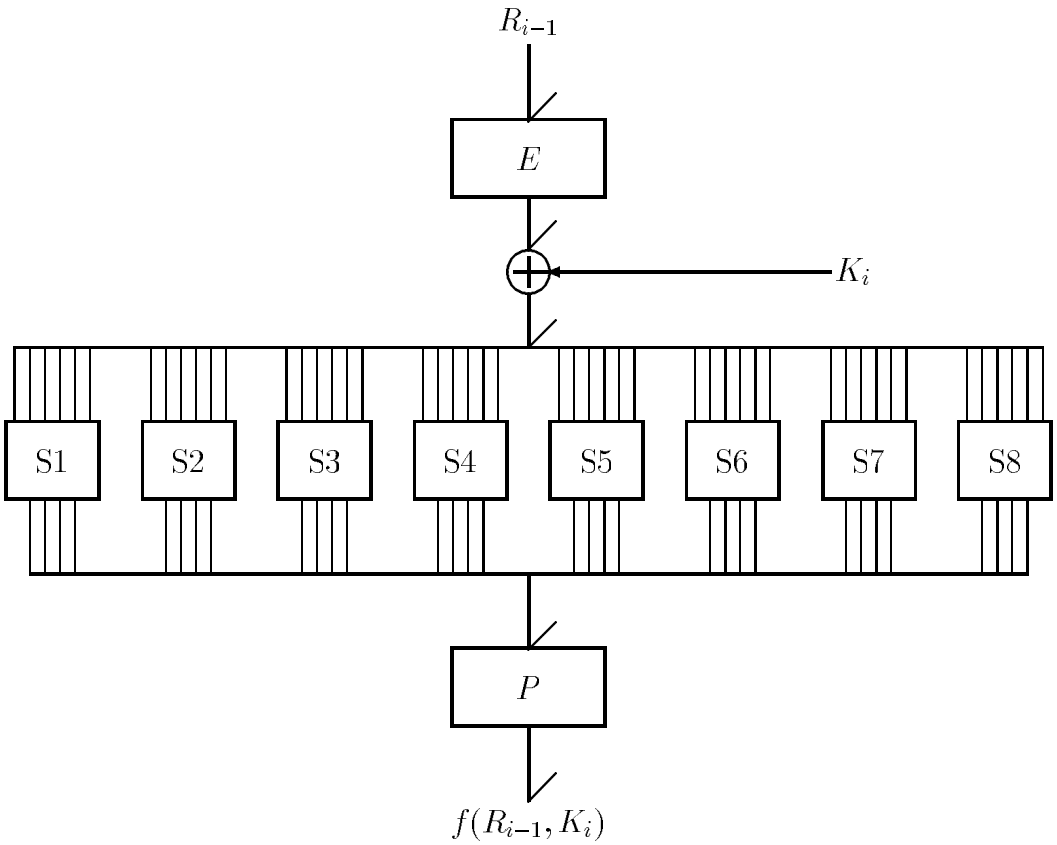

Figure 2.2: f function of DES

Table 2.1: Comparison of DES-like cryptosystems

	DES	FEAL-4	FEAL-8	Multi2	LOKI
Published Year	1977	1987	1988	1989	1990
Proposed by	IBM	NTT	NTT	Hitachi	Brown
Plaintext(bits)	64	64	64	64	64
Key(bits)	56	64	64	64,128	64
Number of round	16	4	8	8	16
Intermediate key(bits)	768	192	256	256	576
f-function	S-box	$+, \oplus, Rot_2$	$+, \oplus, Rot_2$	$+, \oplus, Rot_v*$	S-box

$*$: v has 1,2,4,8,16

operation. In Table 2.1, we compare the important characteristics of published DES-like cryptosystems.

There are four standard modes [18] to operate symmetric cryptosystems for their practical applications.

Electronic Codebook

The natural way of using a block cipher is often called the electronic codebook (ECB) mode and is illustrated in Fig. (2.3). It is the straightforward use of the algorithm to encipher one block.

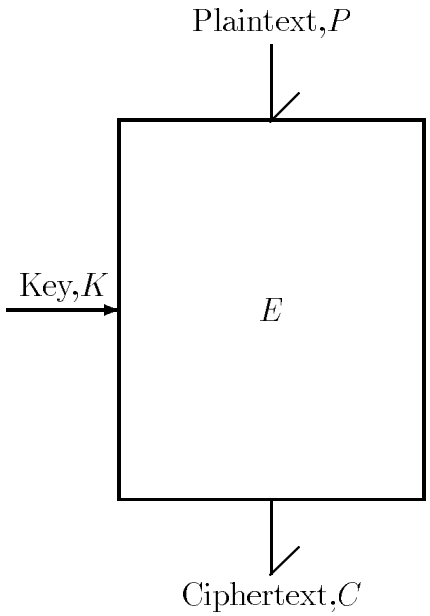

Figure 2.3: Electronic codebook mode

One of the major disadvantages of this mode is that inputting a given n-bit block of data results in the same n-bit block of cipher, *i.e.*, it acts exactly like a codebook. Of course, the consequence of this can be reduced by making n sufficiently large. Nevertheless, with highly formatted data, the same n-bit blocks may still occur relatively frequently with the obvious consequences. Increasing n introduces another disadvantage. No matter how short our message may be we always have to produce an n-bit block of

ciphertext and so, for short messages, increasing n might lead to message extension. Another possible disadvantage is error propagation. A single error in transmission is likely to cause a complete block of data to be in error. This is certainly not always a disadvantage but there are many situations where it is undesirable.

Cipher Feedback

Fig. (2.4) illustrates one of many ways in which a block cipher may be used to produce a cipher feedback(CFB) system. This is similar to the stream cipher shown in Fig. (2.7).

We begin with an n-bit input register filled with an n-bit block Initialization Vector(IV). This IV is then presented to the block cipher algorithm in place of data and an n-bit output block produced. We take k ($1 \leq k \leq n$) bit of output block and exclusive-or them to k-bit of plaintext data. In addition to being transmitted the resulting k-bit of ciphertext are also fedback to become the last k-bit of the n-bit input register. To make room for these new bits in the register, the original n-bit is moved k places to the left. This is often called k-bit CFB.

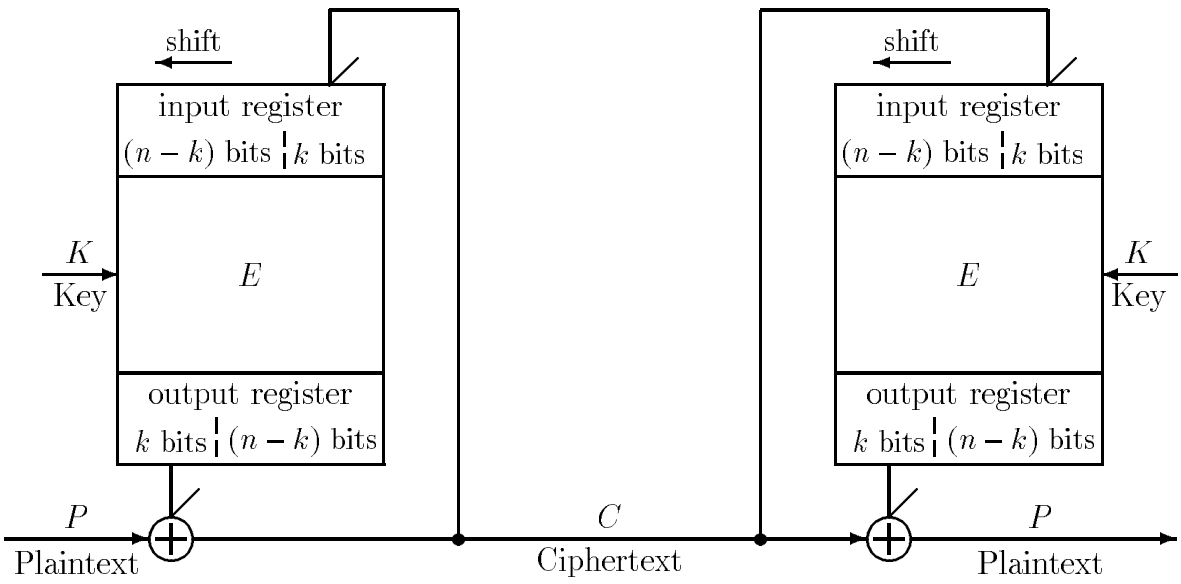

Figure 2.4: k-bit cipher feedback mode

This new block of n-bit is then presented to the block cipher algorithm and the

process continues in the same way. After the first few repetitions, since the input to the algorithm depends only on the ciphertext, this mode is self-synchronizing, but has the error propagation.

Cipher Block Chaining

The cipher bock chaining (CBC) mode is an alternative feedback method as shown in Fig. (2.5). In this mode, all plaintext is handled in n-bit blocks.

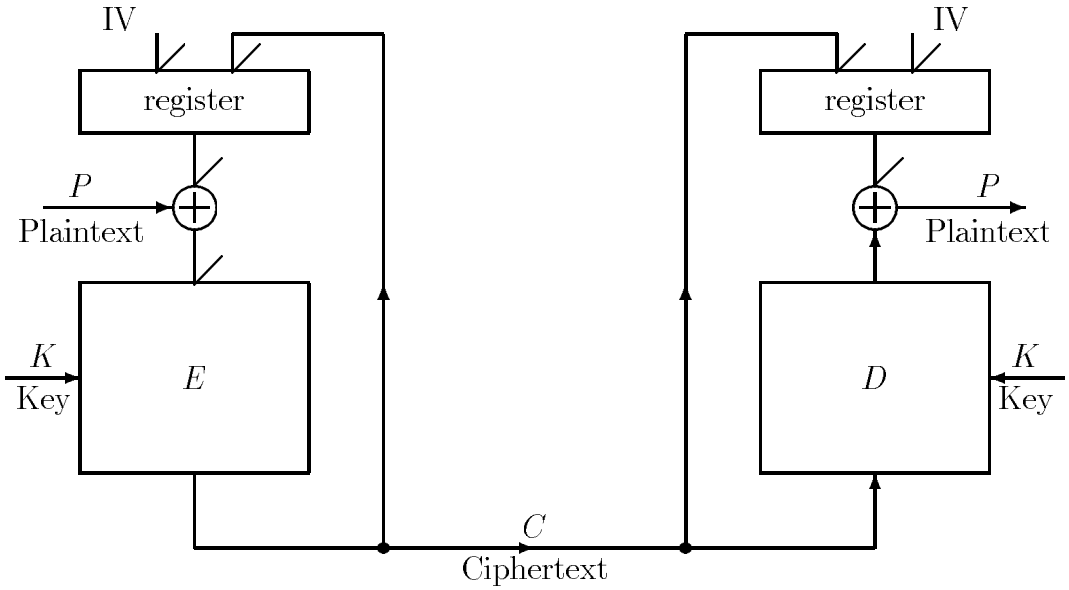

Figure 2.5: Cipher block chaining mode

To begin encryption, an n-bit IV is exclusive-ored with the first n-bit plaintext. The result is presented to the block cipher algorithm and an n-bit ciphertext block is produced in output register. As well as being transmitted, this ciphertext is fedback for bit-by-bit modulo 2-addition with the second plaintext block. The result of this addition passes through the block cipher algorithm as before. To decipher, the first block is presented to the block cipher algorithm and the resultant n-bit block is exclusive-ored with IV to produce the plaintext. From then onwards each ciphertext block is deciphered and then added to the previous ciphertext block. Block chaining is particularly useful if the cryptographer is seeking protection against deletions or the insertion of spurious information.

Output Feedback

This method is intended for applications in which the error-propagation properties of the CBC and CFB are troublesome.

Output feedback uses a cipher of the type invented by Vernam: only the pseudorandom source is new. In one respect, it resembles CFB operation because it can be applied to streams of k-bit ($1 \leq k \leq n$) plaintext. It has the property of all additive stream ciphers that errors in the ciphertext are simply transferred to corresponding bits of the plaintext output.

Fig. (2.6) shows k-bit OFB operation which resembles k-bit CFB operation in all respects except the place from which the feedback is taken.

The non-error propagation property of additive stream cipher carries a disadvantage with it. An active attack on the channel can produce controlled changes to the plaintext. Since OFB systems usually run continuously on a synchronous channel, the enemy may have no knowledge of where each message starts and ends, so the usefulness of this trick to the enemy might be limited. Since OFB mode does not have the self-synchronization property, it needs *ad hoc* initial and continuous synchronization schemes.

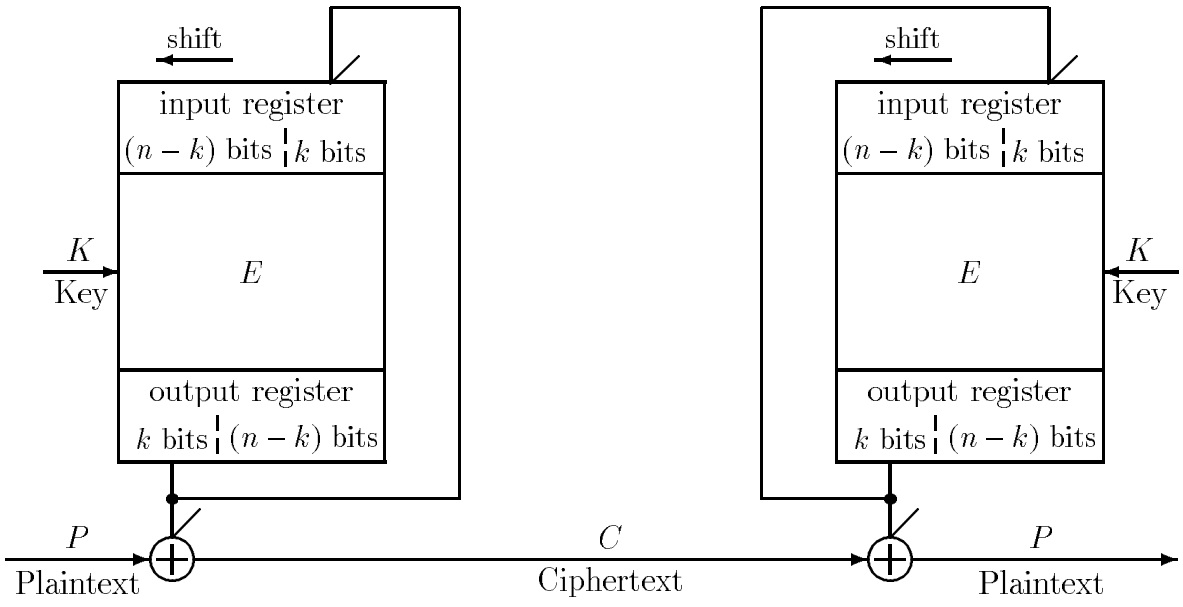

Figure 2.6: k-bit output feedback mode

Some variations of the standard mode summarized in this Section can be considered. Many variations are discussed in the open literature [39] [19] [46] [16].

2.2 Stream Cipher

In a typical stream cipher shown in Fig. (2.7), a plaintext in binary representation is added exclusive-or bit-by-bit to a binary keystream Z.

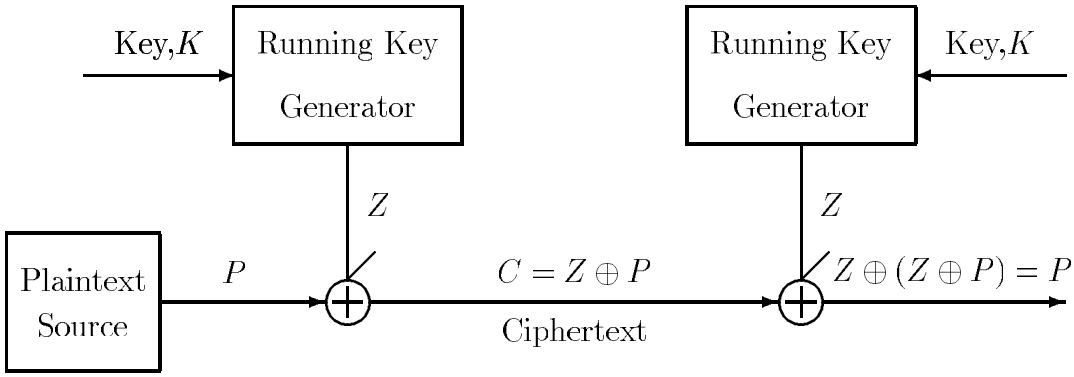

Figure 2.7: Structure of a stream cipher

A prerequisite for the security of a stream cipher is the unpredictability of the key stream. In other words, the keystream should be virtually indistinguishable from a stream of random bits. The deterministic rule under which the keystream is generated by the RKG (Running Key Generator) should be impossible to determine even after observations of a very long segment of the keystream.

In addition, it is required that the statistics of the pseudorandom keystream should be as similar as possible to those of a truly random sequence. However, it is very difficult to generate a truly random sequence.

A LFSR [29] (Linear Feedback Shift Register) which can be characterized by its linear recurring relation [82] is widely used to generate a pseudorandom sequence. A general form of LFSR is shown in Fig. (2.8).

Every periodic sequence (period p) can be generated by a LFSR of length $L < p$. The length of the shortest LFSR that can generate a given sequence is called its linear

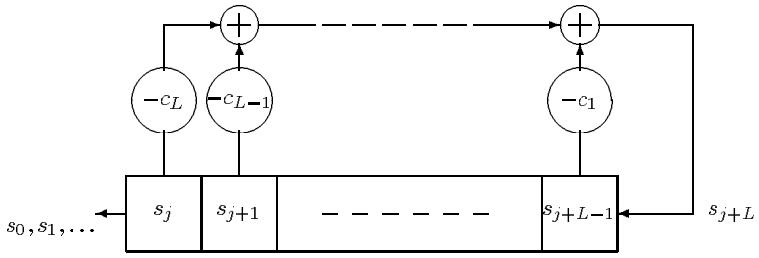

Figure 2.8: A general linear feedback shift register

complexity [65]. Massey [49] has proposed a systematic method to determine the linear complexity of a binary sequence.

If the keystream Z has a linear complexity L, then the observation of $2L$ bits of Z is sufficient to determine very easily the linear equivalent of the corresponding keystream generator. Security demands that the linear complexity (or linear span) of the keystream be large. A large linear span is therefore a necessary (but not sufficient) condition on the keystream of a secure stream cipher.

Many existing RKG's contain one or many LFSR's to obtain large linear complexity. Fig. (2.9) shows a RKG consisting of a single LFSR whose stages are tapped and combined by nonlinear function f and Fig. (2.10) shows a RKG with many LFSR's whose output sequences are combined by the function F. Moreover, Key [36] has given an attainable upperbound and lowerbound of linear complexity of nonlinearly filtered LFSR.

If a nonlinear combiner was chosen to be correlated (*e.g.*, J-K flip-flop [59], exclusive-or [27]) between a sequence of any LFSR, $\tilde{y}_i$ and its combined sequence $\tilde{z}$, that RKG can be broken by a statistical cryptanalysis [64] and ciphertext-only attack [72]. Thus, another security requirement for RKG is that a combining function F should be statistically independent among $\tilde{y}_i$. We call this requirement *correlation immunity*. Correlation-immunity can be modelled as in Fig. (2.11) in the information theoretical sense.

It can be formally defined as:

Definition 2.1 *[71] A memoryless nonlinear function F is said to be m-th order correlation immune if the mutual information between the output variable and any subset of*

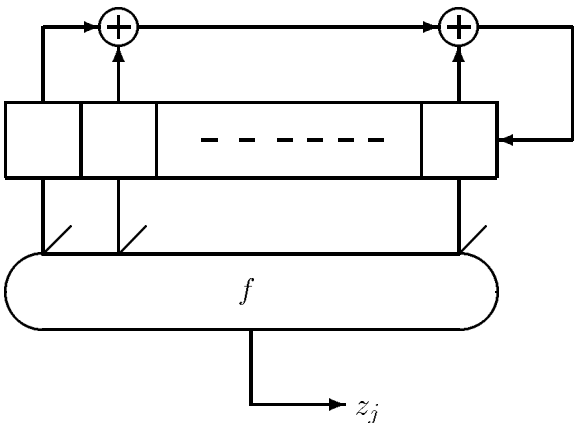

Figure 2.9: RKG with a single LFSR

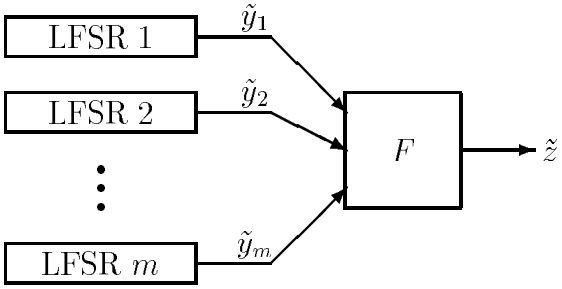

Figure 2.10: RKG with many LFSR's

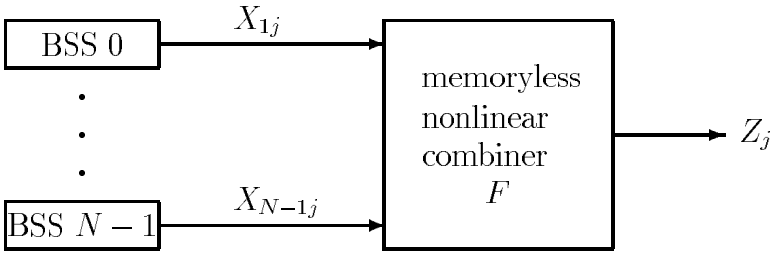

Figure 2.11: Information theoretic model of nonlinear combiner, BSS = Binary Symmetric Source

m input variables is zero, that is, if

$$I(Z; X_{i_1} X_{i_2} \ldots X_{i_m}) \;=\; 0$$

where $0 \leq i_1 < i_2 < \ldots < i_m \leq N-1$ *and* $I(X;Y)$ *is the mutual entropy between* X *and* Y.

If a nonlinear combiner is m-th order correlation-immune, it is not possible to correlate on any combination of m input sequence. But we need some tradeoff between correlation-immunity and nonlinearity. Siegenthaler [71] showed that there exists an important tradeoff between the attainable nonlinear order k[1] and the attainable level of correlation-immunity m,

$$k + m \;\leq\; N$$

where N is the number of input variables to F. When the random variable Z is required to be uniformly distributed, the tradeoff reads as

$$\begin{aligned} k+m &\leq N \quad \text{for } m = 0 \text{ or } m = N-1 \\ k+m &\leq N-1 \quad \text{for } 1 \leq m \leq N-2 \end{aligned}$$

Thus, the more correlation-immunity, the smaller the nonlinear order of F and consequently the smaller the linear complexity of the output sequence. Also, Xias and Massey [79] suggested the Walsh spectral characterization of correlation-immune combined functions. A variety of specific forms of a nonlinear combiner such as clock-controlled shift register [28] [74], stop-and-go generator [7], multiplexed sequences [32], *etc.* were suggested.

Therefore, in general a stream cipher is required to exhibit long period, randomness, high linear complexity (or linear span), unpredictability, nonlinearity and high correlation immunity.

[1]Defined as its maximum order when we express memoryless nonlinear combiner F in algebraic normal form.

Chapter 3

A Theory on Constructing Useful Substitution Boxes

3.1 Notation and Basic Definitions

The notation and the basic definitions used throughout this Chapter are summarized here.

- Z : the set of integers.
- Z_2^n : the n-dimensional vector space over the finite field $Z_2 = GF(2)$.
- $\oplus$: the addition over Z_2^n, or, the bit-wise exclusive-or.
- $wt(\cdot)$: Hamming weight function.
- $|\cdot|$: the cardinality of a set or the absolute value of a real number.

Definition 3.1 *For a positive integer n, define $\mathbf{c}_1^{(n)}, \mathbf{c}_2^{(n)}, \ldots \mathbf{c}_n^{(n)} \in Z_2^n$ by*

$$\begin{aligned} \mathbf{c}_1^{(n)} &= [0, 0, \ldots, 0, 0, 1] \\ \mathbf{c}_2^{(n)} &= [0, 0, \ldots, 0, 1, 0] \\ &\vdots \\ \mathbf{c}_n^{(n)} &= [1, 0, \ldots, 0, 0, 0]. \end{aligned}$$

Intuitively, $\mathbf{c}_i^{(n)}$ means an n dimensional vector with Hamming weight 1 at the i-th position.

Definition 3.2 $\mathbf{x} \cdot \mathbf{w}$ *denotes the dot product of* $\mathbf{x}$ *and* $\mathbf{w}$*, defined as*

$$\mathbf{x} \cdot \mathbf{w} = x_1 w_1 \oplus x_2 w_2 \oplus \cdots \oplus x_n w_n.$$

Definition 3.3 *For a function* $f: Z_2^n \rightarrow Z_2^m$*, denoted by* f_j $(1 \leq j \leq m)$ *the function* $Z_2^n \rightarrow Z_2$ *such that* $f(\mathbf{x}) = (f_m(\mathbf{x}), f_{m-1}(\mathbf{x}), \ldots, f_2(\mathbf{x}), f_1(\mathbf{x}))$*. We identify an element*

$$\mathbf{z} = (z_k, z_{k-1}, \ldots, z_2, z_1)$$

of Z_2^k *with an integer* $\sum_{i=1}^{k} z_i 2^{i-1}$*. To represent a function* $f: Z_2^n \rightarrow Z_2^m$*, we often use the integer tuple*

$$< f >= [f(0), f(1), f(2), \ldots, f(2^n - 1)]$$

and call it the integer representation of f*.*

This representation can be obtained by combining $< f_m >$*,* $< f_{m-1} >$*, ...,* $< f_2 >$*,* $< f_1 >$ *as*

$$< f >= \sum_{j=1}^{m} < f_j > \cdot 2^{j-1}.$$

3.2 Design Criteria

As mentioned before, an S-box in general is considered to be a table lookup memory or a Boolean function from Z_2^n to Z_2^m $(n > m)$ as shown in Fig. (3.1). We will discuss here the formal design criteria of S-boxes for cryptographic purposes.

3.2.1 Strict Avalanche Criterion

Feistel [22] has created one important criterion to design cryptographic functions.

Definition 3.4 (Avalanche effect) *A function* $f: Z_2^n \rightarrow Z_2^m$ *exhibits the avalanche effect if and only if*

$$\sum_{\mathbf{x} \in Z_2^n} wt(f(\mathbf{x}) \oplus f(\mathbf{x} \oplus \mathbf{c}_i^{(n)})) = m 2^{n-1}$$

for all i $(1 \leq i \leq n)$*.*

This means that an average of one half of the output bits change whenever a single input bit is complemented.

Kam and Davida [35] proposed the *completeness* condition that each output bit depends on all input bits of the substitution.

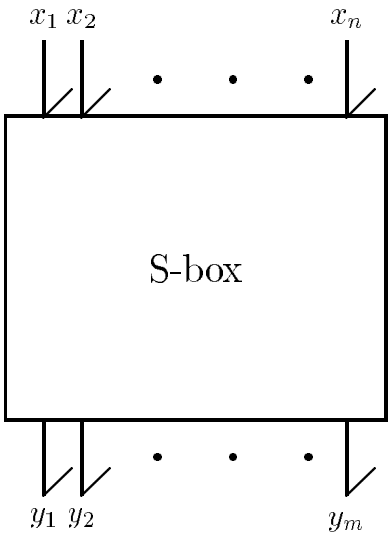

Figure 3.1: S-box

Definition 3.5 (Completeness) *A function $f: Z_2^n \to Z_2^m$ is complete if and only if*

$$\sum_{\mathbf{x} \in Z_2^n} f(\mathbf{x}) \oplus f(\mathbf{x} \oplus \mathbf{c}_i^{(n)}) > (0, 0, \ldots, 0)$$

for all i ($1 \leq i \leq n$), where both the summation and the greater-than are component-wise over Z^m.

This means that each output bit depends on all of the input bits. Thus, if it were possible to find the simplest Boolean expression for each output bit in terms of the input bits, each of those expressions would have to contain all of the input bits if the function is *complete*. Ayoub [3] suggested the probabilistic completeness of substitution-permutation encryption networks.

Webster and Tavares [78] introduced the *Strict Avalanche Criterion* (SAC) in order to combine the notions of the *completeness* and the *avalanche effect*.

Definition 3.6 (SAC, Strong S-box) *We say that a function f: $Z_2^n \to Z_2^m$ satisfies the SAC, or f is a strong S-box, if for all i ($1 \leq i \leq n$) there hold the following equations:*

$$\sum_{\mathbf{x} \in Z_2^n} f(\mathbf{x}) \oplus f(\mathbf{x} \oplus \mathbf{c}_i^{(n)}) = (2^{n-1}, 2^{n-1}, \ldots, 2^{n-1}). \tag{3.1}$$

In particular, if $f: Z_2^n \rightarrow Z_2$ satisfies the SAC, f is called a Boolean strong S-box.

If a function satisfies the SAC, each of its output bits should change with a probability of one half whenever a single input bit is complemented. Clearly, a strong S-box is *complete* and exhibits the *avalanche effect.*

If some output bits depend on only a few input bits, then, by observing a significant number of input-output pairs such as chosen plaintext attack, a cryptanalyst might be able to detect these relations and use this information to aid in the search for the key. And because any lower-dimensional space approximation of a mapping yields a wrong result in 25 % [4] of the cases, strong S-boxes play significant roles in cryptography.

Forré [23] extended this definition of the SAC into a higher order SAC.

Definition 3.7 (1-st order SAC) *A function $f: Z_2^n \rightarrow Z_2^m$ is said to satisfy the 1-st order SAC if and only if*

- *f satisfies the SAC, and*
- *every function obtained from f by keeping the i-th input bit constant and equal to c satisfies the SAC as well for every $i \in \{1, 2, \ldots, n\}$, and for $c = 0$ and $c = 1$.*

Naturally, the SAC defined in **Definition 3.6** can be said to be the 0-*th* order SAC too. To verify whether an n-bit input Boolean function satisfies the 1-*st* order SAC or not, at most $n + n \cdot (n-1)$ checks are required. n checks correspond the 0-*th* order SAC and $n \cdot (n-1)$ checks correspond the 1-*th* order SAC. This definition can be extended to the k-*th* order SAC where $1 \leq k \leq n-2$ if k input bits of $f(\mathbf{x})$ are kept constant.

Definition 3.8 (k-th order SAC) *A function $f: Z_2^n \rightarrow Z_2^m$ is said to satisfy the k-th order SAC if and only if*

- *f satisfies the (k-1)-th SAC, and*
- *any function obtained from f by keeping k of its input bits constant satisfies the SAC as well (this must be true for any choice of the positions and of the values of k constant bits).*

Therefore, verifying whether an n-bit input Boolean function satisfies the k-th order SAC or not requires at most $n + n \cdot (n-1) + \binom{n}{2}(n-2) + \ldots + \binom{n}{k}(n-k)$ checks.

Afterwards, when we talk of a function satisfying the SAC without specifying the order, the function at least satisfies the 0-th order SAC. Moreover, if an n-bit input Boolean function satisfies the (n-2)-th order SAC, the function is referred to as a Boolean function satisfying the maximum order SAC in the sense that the (n-2)-th order SAC is maximally achievable in Boolean functions.

3.2.2 Nonlinearity

It has been widely accepted as a basic criterion that any cryptosystem must be nonlinear. Rueppel [65] [66] has suggested that the nonlinearity of a Boolean function can be measured by the Hamming distance to the set of affine functions and is related to the Walsh transform $\hat{F}$ of $\hat{f}$: $Z_2^n \rightarrow \{1, -1\}$ according to

$$\delta(f) = 2^{n-1} - \frac{1}{2} max_{\mathbf{w}} |\hat{F}(\mathbf{w})|. \tag{3.2}$$

Meier and Staffelbach [50] suggested that the nonlinearity criterion for Boolean functions is classified in view of their suitability for cryptographic design. Their classification is set up in terms of the largest transformation group leaving a criterion invariant. One is the maximum distance to affine functions and the other is the maximum distance to linear structures. These two classifications are shown to be invariant under all affine transformations.

Pieprzyk [56] suggested the amount of nonlinearity and gave an equation to calculate the maximum achievable nonlinearity for an n-bit bijective function.

$$N_f = \begin{cases} \sum_{i=(n-2)/2}^{n-2} 2^i & \text{for } n = 2, 4, 6, \ldots \\ \sum_{i=(n-3)/2}^{n-3} 2^{i+1} & \text{for } n = 3, 5, 7, \ldots \end{cases} \tag{3.3}$$

When n is even, N_f equals to $2^{n-1} - 2^{n/2-1}$. Bent functions which will be defined in Section 3.3.4 always exhibit this maximum nonlinearity. Moreover, Pieprzyk [57] and Finkelstein [58] discussed the nonlinearity applications to cryptosystem design.

3.2.3 Cross Correlation of Avalanche Variables

Avalanche variables are the binary components of the so-called avalanche vectors, defined as the exclusive-or sums

$$\mathbf{V}_i = S(\mathbf{x}) \oplus S(\mathbf{x} \oplus \mathbf{c}_i^{(n)}),$$

where $S(\mathbf{x})$ denote an S-box. In [78], it is stated that, for a given set of avalanche vectors generated by the complementing of a single input bit, all the avalanche variables should be pairwise independent. Their degree of independence is measured by the correlation coefficient $\rho(A, B)$ which, for two random variables A and B, is given as

$$\rho(A, B) = \frac{cov(A, B)}{\sigma(A) \cdot \sigma(B)} \tag{3.4}$$

where

$$\begin{aligned}
cov(A, B) &= \text{covariance of } A \text{ and } B \\
&= E[AB] - E[A] \cdot E[B], \\
\sigma^2[A] &= \text{standard deviation of } A \\
&= E[A^2] - \{E[A]\}^2 \\
\sigma^2[B] &= \text{standard deviation of } B \\
&= E[B^2] - \{E[B]\}^2 \\
E[A], E[B] &= \text{the expected value or mean of A, B} \\
E[AB] &= \text{the expected value of the product A and B}
\end{aligned}$$

Since the value of the avalanche variable has 0 or 1, the value of the correlation coefficient was proved [77] to range from -1 to 1.

The intuitive meanings of this correlation coefficient are:

1. when -1, the avalanche variables are always complements of one another.
2. when 1, the avalanche variables are always identical.
3. when 0, the avalanche variables are independent.

In Section 4.4, $\rho_{i,j}(k)$ denotes the cross correlation between the i-th output bit and the j-th output bit of an avalanche vector V_k.

3.2.4 Bijection

We may or may not need this criterion depending on the structure of DES-like cryptosystems.

In order for an n-bit input function, f to be a bijection (*i.e.*, every possible input vector is mapped into an unique output vector.), Adams and Webster [1] suggested that it must satisfy a necessary and sufficient condition that any linear combination of a Boolean function has Hamming weight 2^{n-1}, *i.e.*,

$$wt(\sum_{i=1}^{n} a_i f_i) = 2^{n-1} \tag{3.5}$$

for any $a_i \in \{0,1\}$, $(a_1, a_2, \ldots a_n) \neq (0,0,\ldots,0)$, and $f = [f_1, f_2, \ldots, f_n]$. This condition allows us to say that every f_i is basically required to be 0/1 balanced for f to be a bijection.

3.3 Properties of S-box Satisfying the SAC

Here we discuss the cryptographic properties of S-boxes satisfying the SAC.

3.3.1 Basic Functions Satisfying the SAC

When the number of input is 2 in a Boolean function, we searched all Boolean functions and obtained the Boolean function satisfying the SAC. Table 3.1 shows all 2-bit input Boolean functions satisfying the SAC. The inside of $(\cdot)$ in the table is a hexadecimal representation of a function. We refer to these functions as "basic functions satisfying the SAC".

By employing these basic functions, we will construct a function satisfying the SAC in Section 3.4.

3.3.2 Some Functions Never Satisfying the SAC

A function that is linear or affine can be defined as:

Definition 3.9 (Linearity, Affinity) *A function f from Z_2^n into Z_2^m is affine if there exist an $n \times m$ matrix $\mathbf{A}_f$ over Z_2 and an m-dimensional vector $\mathbf{b}_f$ over Z_2 such that*

$$f(\mathbf{x}) = \mathbf{x}\mathbf{A}_f + \mathbf{b}_f$$

Table 3.1: Basic functions satisfying the SAC

No.	0123
1	0001(1)
2	0010(2)
3	0100(4)
4	1000(8)
5	0111(7)
6	1011(B)
7	1101(D)
8	1110(E)

where $\mathbf{x}$ *denotes the indeterminate* n*-dimensional vector. A function* f *is linear if it is affine with* $\mathbf{b}_f = \mathbf{0}$.

Hellman *et al* [31] suggested that any cryptosystem which implements linear or affine functions can be easily broken. This fact brings us to the question: Are there linear or affine functions satisfying the SAC ? The answer is of course "no".

Theorem 3.1 *A strong S-box is neither linear nor affine.*

Proof: Kam and Davida [35] showed that there are no *complete* affine functions, and as mentioned before a function which satisfies the SAC must be complete. Thus the conclusion is obvious. However, it is an easy task to give a direct proof:

Let f be an affine function :

$$f(\mathbf{x}) = \mathbf{x}\mathbf{A}_f \oplus \mathbf{b}_f.$$

Then, for each i $(1 \leq i \leq n)$ it holds that

$$\begin{aligned} & \sum_{\mathbf{x} \in Z_2^n} f(\mathbf{x}) \oplus f(\mathbf{x} \oplus \mathbf{c}_i^{(n)}) \\ = & \sum_{\mathbf{x} \in Z_2^n} \mathbf{x}\mathbf{A}_f \oplus \mathbf{b}_f \oplus \mathbf{x}\mathbf{A}_f \oplus \mathbf{c}_i^{(n)}\mathbf{A}_f \oplus \mathbf{b}_f \\ = & \sum_{\mathbf{x} \in Z_2^n} \mathbf{c}_i^{(n)}\mathbf{A}_f. \end{aligned}$$

Because each component of the above summation has either 0 or 2^n, thus f could not satisfy the definition of the SAC. □

And also it is easy to see that

Theorem 3.2 *For $n = 1$, or 2, any bijective function f from Z_2^n into Z_2^n never satisfies the SAC.*

Proof: By virtue of **Theorem 3.1** it is sufficient to show that f is affine. Since if $n = 1$ any function from Z_2 into Z_2 is apparently affine, we consider the case $n = 2$ in the following. When $n = 2$, f can be uniquely represented by

$$f(x_2, x_1) = \mathbf{a}_0 \oplus \mathbf{a}_1 x_1 \oplus \mathbf{a}_2 x_2 \oplus \mathbf{a}_3 x_1 x_2$$

where $\mathbf{a}_i \in Z_2^2 (i = 0, 1, 2, 3)$. Thus,

$$\left.\begin{array}{rcl} f(0) \oplus f(1) & = & \mathbf{a}_1 \\ f(0) \oplus f(2) & = & \mathbf{a}_2 \\ f(1) \oplus f(2) & = & \mathbf{a}_1 \oplus \mathbf{a}_2 \end{array}\right\} \tag{3.6}$$

$$\left.\begin{array}{rcl} f(2) \oplus f(3) & = & \mathbf{a}_1 \oplus \mathbf{a}_3 \\ f(1) \oplus f(3) & = & \mathbf{a}_2 \oplus \mathbf{a}_3 \\ f(0) \oplus f(3) & = & \mathbf{a}_1 \oplus \mathbf{a}_2 \oplus \mathbf{a}_3 \end{array}\right\} \tag{3.7}$$

Since f is *bijective*, none of the above six vectors are zero. From Eq. (3.6), we observe that $\mathbf{a}_1 \neq 0$, $\mathbf{a}_2 \neq 0$, $\mathbf{a}_1 \oplus \mathbf{a}_2 \neq 0$ which means that

$$\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_1 \oplus \mathbf{a}_2\} = \{1, 2, 3\}$$

The facts $\mathbf{a}_1 \neq \mathbf{a}_3$, $\mathbf{a}_2 \neq \mathbf{a}_3$, $\mathbf{a}_1 \oplus \mathbf{a}_2 \neq \mathbf{a}_3$ from Eq. (3.7) indicate that $\mathbf{a}_3 = 0$. Thus f must be affine. □

Thus in order to obtain *bijective* strong S-boxes, we must treat at least quadratic functions of at least three variables.

3.3.3 Property-conserving Transform

When $m = 1$, and $n = 3$ or 4, the experiments tell us that we can easily generate many strong S-boxes $f: Z_2^n \rightarrow Z_2$ by random search on an engineering workstation (SONY NWS810) in a few microseconds. But for the case of $n \geq 5$ it becomes rather difficult to efficiently generate single output strong S-boxes in the same computational environment.

Example 3.1 *For $n = 3$ and $m = 1$,*

$$< p >= [1, 0, 1, 1, 1, 0, 0, 0],$$

$$< q >= [1, 1, 1, 0, 0, 0, 1, 0],$$

$$< r >= [1, 1, 0, 1, 0, 1, 0, 0]$$

are integer representations of strong S-boxes p, q and r respectively. By complementing the output bit of the single output strong S-box p, q and r, we have

$$< p' >= [0, 1, 0, 0, 0, 1, 1, 1],$$

$$< q' >= [0, 0, 0, 1, 1, 1, 0, 1],$$

$$< r' >= [0, 0, 1, 0, 1, 0, 1, 1].$$

It is easy to check that all of these functions are strong S-boxes.

By the definition of the SAC and by the above observation, we can readily show the following.

Theorem 3.3 *Let e and g respectively denote an affine function from Z_2^n and Z_2^m into themselves with a permutation matrix and an arbitrary binary vector. Then, a function $f: Z_2^n \rightarrow Z_2^m$ satisfies the SAC if and only if the composite function $g \circ f \circ e: Z_2^n \rightarrow Z_2^m$ as illustrated in Fig. (3.2) satisfies the SAC.*

Proof: Since every component of the tuple in the right-hand side of Eq. (3.1) is the same, a permutation of the output bits and/or input bits of f does not affect whether or not f satisfies the SAC. Also, when we complement any output bit(s) and/or input bit(s) of f, the number of output bits from 1 to 0 and from 0 to 1 remains constant. This completes the proof. □

Given some single output strong S-boxes, we can generate multiple output strong S-boxes using the idea summarized in the above theorem. However, note that a strong S-box of $m = n$ generated by this method is not guaranteed to be *bijective*.

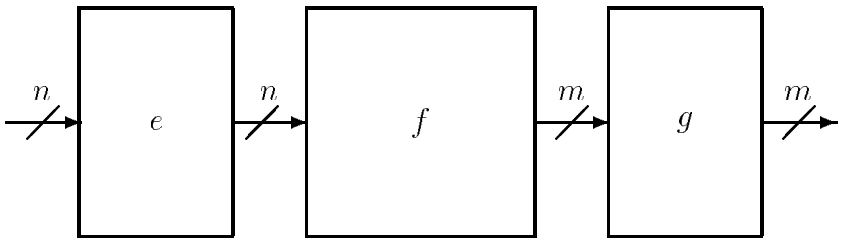

Figure 3.2: Illustrative description of **Theorem 3.3**

Example 3.2 *The 3-input 3-output S-box f defined by $f(\mathbf{x}) = (r(\mathbf{x}), p(\mathbf{x}), q'(\mathbf{x}))$ is strong, i.e., satisfies the SAC. Since*

$$< r >= [1,1,0,1,0,1,0,0],$$

$$< p >= [1,0,1,1,1,0,0,0],$$

$$< q' >= [0,0,0,1,1,1,0,1],$$

then, the integer representation of f is

$$< r > \cdot 4 + < p > \cdot 2 + < q' >= [6,4,2,7,3,5,0,1].$$

As a special case of **Theorem 3.3**, we can consider the input/output dyadic shift transform of any function satisfying the SAC for property-conserving transform.

Definition 3.10 *A function $f(\mathbf{x})$: $Z_2^n \rightarrow Z_2^m$ ($n \geq 2$) is given. Let k be any integer $0 \leq k \leq 2^n - 1$ and let l be any integer $0 \leq k \leq 2^m - 1$. We call $f(\mathbf{x} \oplus k)$ the input dyadic shift transform of f and $f(\mathbf{x}) \oplus l$ the output dyadic shift transform.*

Theorem 3.4 *If f is a function satisfying the SAC, then the input dyadic shift and/or the output dyadic shift transform of f also satisfies the SAC.*

Example 3.3 *When a bijective strong S-box $f(\mathbf{x}) : Z_2^3 \rightarrow Z_2^3$ is given below,*

$$< f >= [5,4,3,2,7,1,6,0].$$

When $k = 2$, the input dyadic shift of $< f >$ is

$$[3, 2, 5, 4, 6, 0, 7, 1]$$

and the output dyadic shift of $< f >$ is

$$[7, 6, 1, 0, 5, 3, 4, 2].$$

These functions satisfy the SAC too.

This transform can be applicable to a Boolean case too.

Thus we can conclude this section by stating that there are no difficulties in efficiently generating many strong S-boxes up to the 4-bit input case.

3.3.4 Relationship between Bent Functions and Boolean Functions Satisfying the SAC

Rothaus [63] coined the term "bent" and defined bent functions as follows:

Definition 3.11 (bent function) *A Boolean function $g(\mathbf{x})$: $Z_2^n \to Z_2$, $n = 2l$, is said to be bent if all the Fourier transform coefficients $G(\mathbf{w})$ of $(-1)^{g(\mathbf{x})}$ as defined for all $\mathbf{w} \in Z_2^n$*

$$G(\mathbf{w}) = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x} \in Z_2^n} (-1)^{g(\mathbf{x}) + \mathbf{x} \cdot \mathbf{w}} \tag{3.8}$$

have unit magnitude i.e.,

$$|G(\mathbf{w})| = 1. \tag{3.9}$$

Bent functions have been given great attention in coding theory [47], threshold logic design [80], spread spectrum communications [55] [73], *etc.* Also, the maximal families of bent sequences [42] and generalized bent functions [40] have been discussed in the open literature. Recently, bent functions are suggested to be closely related to cryptography [60] [45].

Definition 3.12 (Walsh Transform) *[5] If $f(\mathbf{x})$ is any real-valued function whose domain is the vector space Z_2^n, the Walsh transform of $f(\mathbf{x})$ is defined as*

$$F(\mathbf{w}) = \sum_{x \in Z_2^n} f(\mathbf{x}) \cdot (-1)^{\mathbf{x} \cdot \mathbf{w}}, \tag{3.10}$$

where $\mathbf{w} \in Z_2^n$.

The function $f(\mathbf{x})$ *can be recovered from* $F(\mathbf{w})$ *by the inverse Walsh transform :*

$$f(\mathbf{x}) = 2^{-n} \sum_{w \in Z_2^n} F(\mathbf{w}) \cdot (-1)^{\mathbf{x} \cdot \mathbf{w}}. \tag{3.11}$$

The Walsh transform and its inverse (both defined for real-valued functions) may be applied to Boolean functions if their values are the real values 0 and 1.

For symmetry reasons, it is often convenient to map a 0/1 valued Boolean function $f(\mathbf{x})$ into an 1/-1 valued Boolean function $\hat{f}(\mathbf{x})$. We denote this mapping as

$$\hat{f}(\mathbf{x}) = 1 - 2 \cdot f(\mathbf{x}) \text{ or } \hat{f}(\mathbf{x}) = (-1)^{f(\mathbf{x})}$$

Forré [23] has proved that we can check the SACness of $\hat{f}$ in term of its Walsh spectrum as follows:

Theorem 3.5 *A function* $\hat{f}(\mathbf{x})$: $Z_2^n \to \{1, -1\}$ *fulfills the SAC if and only if its Walsh transform* $\hat{F}(\mathbf{w})$ *satisfies*

$$\sum_{\mathbf{w} \in Z_2^n} (-1)^{c_i^{(n)} \cdot \mathbf{w}} \cdot (\hat{F}(\mathbf{w}))^2 = 0 \tag{3.12}$$

for all $i \in \{1, 2, \ldots, n\}$.

It is well known [63] [55] that bent functions have the following properties:

B1 Only exist for an even number of input bits.

B2 Always unbalanced. (*i.e.*, For n-bit input their Hamming weight is $2^{n-1} \pm 2^{n/2-1}$)

B3 The Walsh transform of a bent function is bent.

B4 Closed under linear or affine transform.

B5 Exhibit maximum nonlinearity as defined in Section 3.2.2.

We will state the following theorem between the relationship between bent functions and Boolean functions satisfying the SAC.

Theorem 3.6 *Let $\mathcal{A}_n$ denote the set of all n-bit input Boolean functions, $\mathcal{B}_n$ denote the set of n-bit input bent functions, and $\mathcal{S}_n$ denote the set of n-bit input Boolean functions satisfying the SAC. In particular, we denote the set of n-bit input Boolean functions satisfying the maximum order SAC by $\mathcal{S}_n^{max}$. The relationship between these sets for even n can be stated as*

$$\mathcal{S}_n^{max} \subseteq \mathcal{B}_n \subseteq \mathcal{S}_n \subset \mathcal{A}_n.$$

Proof: Since Adams and Tavares [2] proved that all Boolean functions satisfying the maximum order SAC are bent, it is clear that $\mathcal{S}_n^{max} \subseteq \mathcal{B}_n$.

We will prove that $\mathcal{B}_n \subseteq \mathcal{S}_n$. By **Definition 3.11**, we insert any bent function into Eq. (3.12) and check if the right hand side of the equation becomes zero. Since bent functions have unit magnitude, it is clear that bent functions always satisfy Eq. (3.12). Thus we complete the proof. □

Here we give examples of two cases. The function in Table 3.2 is bent and satisfying the SAC since it satisfies Eq. (3.9) and Eq. (3.12).

Table 3.2: A function is bent and satisfying the SAC

x_1/w_1	x_2/w_2	x_3/w_3	x_4/w_4	$f(\mathbf{x})$	$\hat{F}(\mathbf{w})$	$G(\mathbf{w})$
0	0	0	0	0	4	1
0	0	0	1	1	-4	-1
0	0	1	0	1	-4	-1
0	0	1	1	0	4	1
0	1	0	0	1	-4	-1
0	1	0	1	0	4	1
0	1	1	0	1	-4	-1
0	1	1	1	0	4	1
1	0	0	0	1	-4	-1
1	0	0	1	1	-4	-1
1	0	1	0	0	4	1
1	0	1	1	0	4	1
1	1	0	0	0	4	1
1	1	0	1	0	4	1
1	1	1	0	0	4	1
1	1	1	1	0	4	1

The function in Table 3.3 satisfies the SAC but is not bent since it satisfies Eq. (3.12) but does not satisfy Eq. (3.9). (*i.e.,* it did not have unit magnitude of its Fourier spectrum.)

Table 3.3: A function is not bent but satisfying the SAC

x_1/w_1	x_2/w_2	x_3/w_3	x_4/w_4	$f(\mathbf{x})$	$\hat{F}(\mathbf{w})$	$G(\mathbf{w})$
0	0	0	0	0	8	2
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	0	0	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	1	8	2
1	0	0	0	0	-8	-2
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	0	0
1	1	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	0	0	0	0
1	1	1	1	0	8	2

By computer search, we confirmed the cardinality of each set in Table 3.4. This Table supports **Theorem 3.6.** Note that Lloyd [44] has proved $|S_n^{max}|$ equals 2^{n+1}. All 4-bit input bent functions are listed in **Appendix B** for reference.

Table 3.4: The cardinality of the set

n	2	3	4	5	6
$\lvert\mathcal{A}_n\rvert$	16	256	65,536	2^{32}	2^{64}
$\lvert\mathcal{S}_n\rvert$	8	64	4,128	?	?
$\lvert\mathcal{B}_n\rvert$	8	NE	896	NE	$2^{32.3}$*
$\lvert\mathcal{S}_n^{max}\rvert$	8	16	32	64	128

NE : Not Existent, * : 5,425,430,528

3.4 Construction of S-boxes Satisfying the SAC

3.4.1 Boolean Functions

Next we discuss the expandable properties of Boolean strong S-boxes and present the recursive construction of Boolean strong S-boxes of arbitrary n and m.

Construction by Concatenation(I)

Let us construct $(n+1)$-bit input S-boxes using n-bit input S-boxes.

Definition 3.13 *For a function $f: Z_2^n \to Z_2$, an integer $k \in \{1, 2, \ldots, n\}$ and a constant $b \in Z_2$, define a function $\mathbf{D}_b^k[f]: Z_2^{n+1} \to Z_2$ by $\mathbf{D}_b^k[f](0, \mathbf{x}) = f(\mathbf{x})$ and $\mathbf{D}_b^k[f](1, \mathbf{x}) = f(\mathbf{x} \oplus \mathbf{c}_k^{(n)}) \oplus b$ for all $\mathbf{x} \in Z_2^n$.*

Definition 3.14 *For a function $f: Z_2^n \to Z_2^n$ such that*

$$f(\mathbf{x}) \;=\; (f_n(\mathbf{x}), f_{n-1}(\mathbf{x}), \ldots, f_1(\mathbf{x}))$$

and a function $g: Z_2^n \to Z_2$ and an integer $k \in \{1, 2, \ldots, n\}$, define the function $\mathbf{E}^k[g, f] : Z_2^{n+1} \to Z_2^{n+1}$ by

$$\mathbf{E}^k[g, f](\mathbf{y}) = (\mathbf{D}_1^k[g](\mathbf{y}), \mathbf{D}_0^k[f_n](\mathbf{y}), \mathbf{D}_0^k[f_{n-1}](\mathbf{y}), \ldots, \mathbf{D}_0^k[f_1](\mathbf{y}))$$

for all $\mathbf{y} \in Z_2^{n+1}$.

We can show that the constructed S-boxes have nice properties.

Theorem 3.7 *If a function $f: Z_2^n \to Z_2$ satisfies the SAC, then for any $k \in \{1, 2, \ldots, n\}$ and any $b \in Z_2$, $\mathbf{D}_b^k[f]$ also satisfies the SAC.*

Proof: Since f satisfies the SAC, it holds that

$$\sum_{\mathbf{x} \in Z_2^n} f(\mathbf{x}) \oplus f(\mathbf{x} \oplus \mathbf{c}_i^{(n)}) = 2^{n-1}$$

for any $i \in \{1, 2, \ldots, n\}$. Thus it also holds that

$$\begin{aligned} & \sum_{\mathbf{x} \in Z_2^n} f(\mathbf{x}) \oplus f(\mathbf{x} \oplus \mathbf{c}_i^{(n)}) \oplus 1 \\ = \;& 2^n - \sum_{\mathbf{x} \in Z_2^n} f(\mathbf{x}) \oplus f(\mathbf{x} \oplus \mathbf{c}_i^{(n)}) \\ = \;& 2^n - 2^{n-1} \\ = \;& 2^{n-1} \end{aligned}$$

To prove the theorem, we denote $\mathbf{D}_b^k[f]$ by g and show that for any $i \in \{1, 2, \ldots, n+1\}$,

$$\sum_{\mathbf{y} \in Z_2^{n+1}} g(\mathbf{y}) \oplus g(\mathbf{y} \oplus \mathbf{c}_i^{(n+1)}) = 2^n$$

(Case 1) $i \in \{1, 2, \ldots, n\}$.

$$\begin{aligned}
& \sum_{\mathbf{y} \in Z_2^{n+1}} g(\mathbf{y}) \oplus g(\mathbf{y} \oplus \mathbf{c}_i^{(n+1)}) \\
= & \sum_{\mathbf{x} \in Z_2^n} g(0, \mathbf{x}) \oplus g(0, \mathbf{x} \oplus \mathbf{c}_i^{(n)}) + \sum_{\mathbf{x} \in Z_2^n} g(1, \mathbf{x}) \oplus g(1, \mathbf{x} \oplus \mathbf{c}_i^{(n)}) \\
= & \sum_{\mathbf{x} \in Z_2^n} f(\mathbf{x}) \oplus f(\mathbf{x} \oplus \mathbf{c}_i^{(n)}) + \sum_{\mathbf{x} \in Z_2^n} (f(\mathbf{x} \oplus \mathbf{c}_k^{(n)}) \oplus b) \oplus (f((\mathbf{x} \oplus \mathbf{c}_i^{(n)}) \oplus \mathbf{c}_k^{(n)}) \oplus b) \\
= & \sum_{\mathbf{x} \in Z_2^n} f(\mathbf{x}) \oplus f(\mathbf{x} \oplus \mathbf{c}_i^{(n)}) + \sum_{\mathbf{x} \in Z_2^n} f(\mathbf{x} \oplus \mathbf{c}_k^{(n)}) \oplus f((\mathbf{x} \oplus \mathbf{c}_k^{(n)}) \oplus \mathbf{c}_i^{(n)}) \\
= & 2 \cdot \sum_{\mathbf{x} \in Z_2^n} f(\mathbf{x}) \oplus f(\mathbf{x} \oplus \mathbf{c}_i^{(n)}) \\
= & 2 \cdot 2^{n-1} \\
= & 2^n
\end{aligned}$$

(Case 2) $i = n + 1$

$$\begin{aligned}
& \sum_{\mathbf{y} \in Z_2^{n+1}} g(\mathbf{y}) \oplus g(\mathbf{y} \oplus \mathbf{c}_{n+1}^{(n+1)}) \\
= & \sum_{\mathbf{x} \in Z_2^n} g(0, \mathbf{x}) \oplus g(1, \mathbf{x}) + \sum_{\mathbf{x} \in Z_2^n} g(1, \mathbf{x}) \oplus g(0, \mathbf{x}) \\
= & 2 \cdot \sum_{\mathbf{x} \in Z_2^n} g(0, \mathbf{x}) \oplus g(1, \mathbf{x}) \\
= & 2 \cdot \sum_{\mathbf{x} \in Z_2^n} f(\mathbf{x}) \oplus f(\mathbf{x} \oplus \mathbf{c}_k^{(n)}) \oplus b \\
= & 2 \cdot 2^{n-1} \\
= & 2^n
\end{aligned}$$

Thus, we complete the proof. □

Example 3.4 *A function* $f: Z_2^3 \to Z_2$ *which satisfies the SAC is given as* $< f >= [1, 1, 0, 0, 0, 1, 0, 1]$. *Then,*

$$\begin{aligned}
< \mathbf{D}_0^1[f] > &= [1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0], \\
< \mathbf{D}_1^1[f] > &= [1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1].
\end{aligned}$$

By **Theorem 3.7**, *these expanded functions also satisfy the SAC.*

Construction by Concatenation(II)

In the last Section, we discussed how to enlarge two given distinct n-bit input Boolean functions satisfying the SAC into one $(n+1)$-bit input Boolean function satisfying the SAC. At this time we construct unique $(n+1)$-bit Boolean functions satisfying the SAC as many as possible.

Theorem 3.8 *When two distinct Boolean functions f_i and f_j: $Z_2^n \to Z_2$ satisfying the SAC are given, the concatenated Boolean function g: $Z_2^{n+1} \to Z_2$ i.e.,*

$$g = f_i || f_j \tag{3.13}$$

satisfies the SAC if and only if

$$wt(< f_i > \oplus < f_j >) = 2^{n-1}. \tag{3.14}$$

Proof: We will show that $\sum_{\mathbf{x} \in Z_2^{n+1}} g(\mathbf{x}) \oplus g(\mathbf{x} \oplus \mathbf{c}_k^{(n+1)}) = 2^n$ for any $k \in \{1, 2, \ldots, n+1\}$. Let us denote the $(n+1)$-bit input vector of g as $\mathbf{x} = (x_{n+1}, \mathbf{x}')$ where $\mathbf{x}' = (x_n, x_{n-1}, \ldots, x_1)$. Also let $\mathbf{x}_0$ denote $(0, \mathbf{x}')$ and let $\mathbf{x}_1$ denote $(1, \mathbf{x}')$.

Note that

$$g(\mathbf{x}) = \begin{cases} f_i(\mathbf{x}') & \text{if } x_{n+1} = 0, \\ f_j(\mathbf{x}') & \text{if } x_{n+1} = 1. \end{cases}$$

For $k = 1, 2, \ldots, n$,

$$\begin{aligned} \sum_{\mathbf{x} \in Z_2^{n+1}} g(\mathbf{x}) \oplus g(\mathbf{x} \oplus \mathbf{c}_k^{(n+1)}) &= \sum_{\mathbf{x} \in Z_2^{n+1}} g(\mathbf{x}_0) \oplus g(\mathbf{x}_0 \oplus \mathbf{c}_k^{(n+1)}) + \sum_{\mathbf{x} \in Z_2^{n+1}} g(\mathbf{x}_1) \oplus g(\mathbf{x}_1 \oplus \mathbf{c}_k^{(n+1)}) \\ &= \sum_{\mathbf{x}' \in Z_2^n} f_i(\mathbf{x}') \oplus f_i(\mathbf{x}' \oplus \mathbf{c}_{k'}^{(n)}) + \sum_{\mathbf{x}' \in Z_2^n} f_j(\mathbf{x}') \oplus f_j(\mathbf{x}' \oplus \mathbf{c}_{k'}^{(n)}) \\ &= 2^{n-1} + 2^{n-1} \\ &= 2^n. \end{aligned}$$

When $k = n+1$, let $< g_l >$ denote the left half of $< g >$ and let $< g_r >$ denote the right half of $< g >$. In this case, the condition of SACness can be alternatively checked like

$$wt(< g_l > \oplus < g_r >) = 2^{n-1}.$$

This is equivalent to Eq. (3.14).

Therefore g satisfies the SAC for all $k = 1, 2, \ldots, n, n+1$. We complete the proof. □

Example 3.5 *When* $< f_1 >= [0,0,0,1]$ *and* $< f_2 >= [0,0,1,0]$,

$$\begin{aligned} < g > &= < f_1 > || < f_2 > \\ &= [0,0,0,1,0,0,1,0] \end{aligned}$$

satisfies the SAC.

We applied recursively this method to enlarge basic functions satisfying the SAC to any bit input Boolean functions satisfying the SAC. We can generate 48 3-bit input Boolean functions satisfying the SAC from 8 basic functions satisfying the SAC and 1,440 4-bit input Boolean functions satisfying the SAC from 48 3-bit input Boolean functions satisfying the SAC. The results generated by this method are summarized in Table 3.5. In Table 3.5, A_n denotes the number of all n-bit input Boolean functions, B_n denotes the number of n-bit input Boolean functions satisfying the SAC and C_n denotes the number of n-bit input Boolean functions satisfying the SAC generated by this method.

Table 3.5: Generated results of unique Boolean functions satisfying the SAC

n	2	3	4	5
A_n	16	256	65,536	2^{32}
B_n	8	64	4,128	?
C_n	8	48	1,440	980,160

But, this method can not generate all functions satisfying the SAC. The following examples are two 3-bit input Boolean functions satisfying the SAC which this method could not generate.

$$[0,1,1,0,0,0,0,0],$$
$$[0,1,1,0,1,1,1,1].$$

From this example, we may consider that some parts of the functions might be constructed by employing the exclusive-or's of some basic functions satisfying the SAC before two functions are concatenated in this method.

Use of the Kronecker product

Let A and B be $m \times n$ matrix and $r \times k$ matrix respectively such that

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

and

$$B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1k} \\ b_{21} & b_{22} & \cdots & b_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ b_{r1} & b_{r2} & \cdots & b_{rk} \end{bmatrix}$$

then the Kronecker (or direct) product $A \otimes B$ is defined by

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix}$$

which is an $(mr) \times (nk)$ matrix. By using the Kronecker product, we can generate $2l$-bit input Boolean function satisfying the SAC from basic functions satisfying the SAC. At this moment we use a mapping $0 \rightarrow 1$ and $1 \rightarrow -1$. We denote this mapping as

$$\hat{f}(\mathbf{x}) = 1 - 2 \cdot f(\mathbf{x}) \text{ or } \hat{f}(\mathbf{x}) = (-1)^{f(\mathbf{x})}$$

where $f(\mathbf{x})$ is a 0/1 valued function.

Theorem 3.9 *When an 1/-1 valued basic function $\hat{f}$ satisfying the SAC is given, then an 1/-1 valued $2l$-bit ($l \geq 2$) input Boolean function $\hat{g}$ extended by*

$$< \hat{g}^l >= \underbrace{< \hat{f} > \otimes < \hat{f} > \otimes \cdots \otimes < \hat{f} >}_{l-1 \ times}$$

satisfies the SAC.

Proof: We can prove this theorem by mathematical induction.
(**Basis**) $l = 2$. $< \hat{g}^2 >=< \hat{f} > \otimes < \hat{f} >$. Because $\hat{f}(\mathbf{x})$ satisfies the SAC, it holds that $\sum_{\mathbf{x} \in Z_2^2} f(\mathbf{x}) \oplus f(\mathbf{x} \oplus \mathbf{c}_j^{(2)}) = 2$ or $\sum_{\mathbf{x} \in Z_2^2} \hat{f}(\mathbf{x}) \hat{f}(\mathbf{x} \oplus \mathbf{c}_j^{(2)}) = 0$ for $j = 1, 2$. Let

$< \hat{f} >= [x_0, x_1, x_2, x_3]$ and $x_i \in \{1, -1\}$ for $i = 0, 1, 2, 3$. Then, we can obtain that

$$\begin{aligned} x_0x_1 + x_2x_3 &= 0 \text{ and} \\ x_0x_2 + x_1x_3 &= 0. \end{aligned}$$

Also,

$$\begin{aligned} < \hat{g}^2 > &= < \hat{f} > \otimes < \hat{f} > \\ &= [x_0 < \hat{f} >, x_1 < \hat{f} >, x_2 < \hat{f} >, x_3 < \hat{f} >] \\ &= [x_0^2, x_0x_1, x_0x_2, x_0x_3, x_0x_1, x_1^2, x_1x_2, x_1x_3, \\ &\quad x_0x_2, x_1x_2, x_2^2, x_2x_3, x_0x_3, x_1x_3, x_2x_3, x_3^2] \end{aligned}$$

We must prove that $\sum_{\mathbf{x} \in Z_2^4} \hat{g}^2(\mathbf{x}) \hat{g}^2(\mathbf{x} \oplus \mathbf{c}_k^{(4)}) = 0$ for any $k \in \{1, 2, 3, 4\}$.
(Case 1) When $k = 1$,

$$\begin{aligned} &\sum_{\mathbf{x} \in Z_2^4} \hat{g}^2(\mathbf{x}) \hat{g}^2(\mathbf{x} \oplus \mathbf{c}_1^{(4)}) \\ = \; & x_0^2 \cdot x_0x_1 + x_0x_2 \cdot x_0x_3 + x_0x_1 \cdot x_1^2 + x_1x_2 \cdot x_1x_3 \\ & + x_0x_2 \cdot x_1x_2 + x_2^2 \cdot x_2x_3 + x_0x_3 \cdot x_1x_3 + x_2x_3 \cdot x_3^2 \\ = \; & x_0x_1 + x_2x_3 + x_0x_1 + x_2x_3 \\ & + x_0x_1 + x_2x_3 + x_0x_1 + x_2x_3 \\ = \; & 0. \end{aligned}$$

(Case 2) When $k = \{2, 3, 4\}$, by the same way we can obtain

$$\sum_{\mathbf{x} \in Z_2^4} \hat{g}^2(\mathbf{x}) \hat{g}^2(\mathbf{x} \oplus \mathbf{c}_k^{(4)}) = 0.$$

Therefore, $\hat{g}^2$ is a 4-bit input Boolean function satisfying the SAC.
(**Induction Hypothesis**) When $l = n$, we assume that

$$< \hat{g}^n >= \underbrace{< \hat{f} > \otimes < \hat{f} > \otimes \cdots \otimes < \hat{f} >}_{n-1 \;\; times}$$

satisfies the SAC, *i.e.*,

$$\sum_{\mathbf{x} \in Z_2^{2n}} \hat{g}^n(\mathbf{x}) \hat{g}^n(\mathbf{x} \oplus \mathbf{c}_j^{(2n)}) = 0$$

for all $j = \{1, 2, \ldots, 2n\}$. Let $< \hat{g}^n >= [x'_0, x'_1, \ldots, x'_{4^n-1}]$, then the following equations hold.

$$\begin{aligned} x'_0x'_1 + x'_2x'_3 + \cdots + x'_{4^n-2}x'_{4^n-1} &= 0 \quad (\diamondsuit) \\ x'_0x'_2 + x'_1x'_3 + \cdots + x'_{4^n-3}x'_{4^n-1} &= 0 \\ &\cdot \\ &\cdot \\ &\cdot \\ x'_0x'_{2^n} + x'_1x'_{2^n+1} + \cdots + x'_{2^n-1}x'_{4^n-1} &= 0 \end{aligned}$$

(**Induction**) When $l = n + 1$, we show that

$$\sum_{\mathbf{x} \in Z_2^{2n+2}} \hat{g}^{n+1}(\mathbf{x}) \hat{g}^{n+1}(\mathbf{x} \oplus \mathbf{c}_k^{(2n+2)}) = 0.$$

On the other hand,

$$\begin{aligned} < \hat{g}^{n+1} > &= < \hat{f} > \oplus < \hat{g}^n > \\ &= [x_0 < \hat{g}^n >, x_1 < \hat{g}^n >, x_2 < \hat{g}^n >, x_3 < \hat{g}^n >]. \end{aligned}$$

When $k = 1$, we can obtain that

$$\begin{aligned} \sum_{\mathbf{x} \in Z_2^{2n+2}} \hat{g}^{n+1}(\mathbf{x}) \hat{g}^{n+1}(\mathbf{x} \oplus \mathbf{c}_1^{(2n+2)}) &= x_0^2 \cdot (\diamondsuit) + x_1^2 \cdot (\diamondsuit) + x_2^2 \cdot (\diamondsuit) + x_3^2 \cdot (\diamondsuit) \\ &= 0. \end{aligned}$$

By the same way when $k = \{2, 3, \ldots, 2n + 2\}$, we can obtain

$$\sum_{\mathbf{x} \in Z_2^{2n+2}} \hat{g}^{n+1}(\mathbf{x}) \hat{g}^{n+1}(\mathbf{x} \oplus \mathbf{c}_k^{(2n+2)}) = 0.$$

Thus, we complete the proof.

□

Example 3.6 *An 1/-1 valued basic function satisfying the SAC is given below,*

$$< \hat{f} >= [1, 1, -1, 1].$$

Then,

$$\begin{aligned} <\hat{g}^2> &= <\hat{f}> \otimes <\hat{f}> \\ &= [1,1,-1,1,1,1,-1,1,-1,-1,1,-1,1,1,-1,1]. \end{aligned}$$

$<\hat{g}^2>$ *is an 1/-1 valued 4-bit input Boolean function satisfying the SAC.*

3.4.2 Bijective Functions

We state the necessary condition that a function resulting from the combination of one Boolean function and one bijective function be also bijective.

Theorem 3.10 *For a bijection* $f: Z_2^n \to Z_2^n$*, a function* $g: Z_2^n \to Z_2$*, and an integer* $k \in \{1, 2, \ldots, n\}$*, the function* $\mathbf{E}^k[g, f]: Z_2^{n+1} \to Z_2^{n+1}$ *is bijective.*

Proof: By the definition of $\mathbf{E}^k[g, f]$ (**Definition 3.14**), we have for any $\mathbf{x} \in Z_2^n$,

$$\begin{aligned} \mathbf{E}^k[g,f](0,\mathbf{x}) &= (g(\mathbf{x}), f(\mathbf{x})), \\ \mathbf{E}^k[g,f](1,\mathbf{x} \oplus \mathbf{c}_k^{(n)}) &= (g(\mathbf{x}) \oplus 1, f(\mathbf{x})). \end{aligned}$$

For any $\mathbf{u} \in Z_2^n$ and $\mathbf{v} \in Z_2^n$, let

$$\begin{aligned} A(\mathbf{u},\mathbf{v}) &= \mathbf{E}^k[g,f](0,\mathbf{u}) \oplus \mathbf{E}^k[g,f](0,\mathbf{v}), \\ B(\mathbf{u},\mathbf{v}) &= \mathbf{E}^k[g,f](1,\mathbf{u} \oplus \mathbf{c}_k^{(n)}) \oplus \mathbf{E}^k[g,f](1,\mathbf{v} \oplus \mathbf{c}_k^{(n)}), \\ C(\mathbf{u},\mathbf{v}) &= \mathbf{E}^k[g,f](0,\mathbf{u}) \oplus \mathbf{E}^k[g,f](1,\mathbf{v} \oplus \mathbf{c}_k^{(n)}). \end{aligned}$$

We have

$$\begin{aligned} A(\mathbf{u},\mathbf{v}) &= B(\mathbf{u},\mathbf{v}) \\ &= (g(\mathbf{u}) \oplus g(\mathbf{v}), f(\mathbf{u}) \oplus f(\mathbf{v})) \\ C(\mathbf{u},\mathbf{v}) &= (g(\mathbf{u}) \oplus g(\mathbf{v}) \oplus 1, f(\mathbf{u}) \oplus f(\mathbf{v})) \end{aligned}$$

Since f is *bijective*, $f(\mathbf{u}) \oplus f(\mathbf{v}) = 0$ if and only if $\mathbf{u} = \mathbf{v}$. Therefore, if $\mathbf{u} \neq \mathbf{v}$, we have $A(\mathbf{u},\mathbf{v}) = B(\mathbf{u},\mathbf{v}) \neq (0,0)$ and $C(\mathbf{u},\mathbf{v}) \neq (0,0)$. And if $\mathbf{u} = \mathbf{v}$, we have $A(\mathbf{u},\mathbf{v}) = B(\mathbf{u},\mathbf{v}) = (0,0)$ and $C(\mathbf{u},\mathbf{v}) = (1,0) \neq (0,0)$. Thus, $A(\mathbf{u},\mathbf{v})$ and $B(\mathbf{u},\mathbf{v})$ is equal to zero if and only if $\mathbf{u} = \mathbf{v}$, and $C(\mathbf{u},\mathbf{v})$ is never equal to zero for any $\mathbf{u}$ and $\mathbf{v}$. These facts show that for any $\mathbf{s} \in Z_2^{n+1}$ and $\mathbf{t} \in Z_2^{n+1}$, $\mathbf{E}^k[g,f](\mathbf{s}) = \mathbf{E}^k[g,f](\mathbf{t})$ if and only if $\mathbf{s} = \mathbf{t}$, or in other words, $\mathbf{E}^k[g,f]$ is *bijective*. □

Theorem 3.11 *If both a bijection* $f: Z_2^n \to Z_2^n$ *and a function* $g: Z_2^n \to Z_2$ *satisfy the SAC, then for any integer* $k \in \{1, 2, \ldots, n\}$*, the function* $\mathbf{E}^k[g, f]: Z_2^{n+1} \to Z_2^{n+1}$ *is a bijection satisfying the SAC.*

Proof: This theorem follows directly from **Theorems 3.7** and **3.10.** □

For explanatory purposes, we illustrate this method as shown in Fig. (3.3).

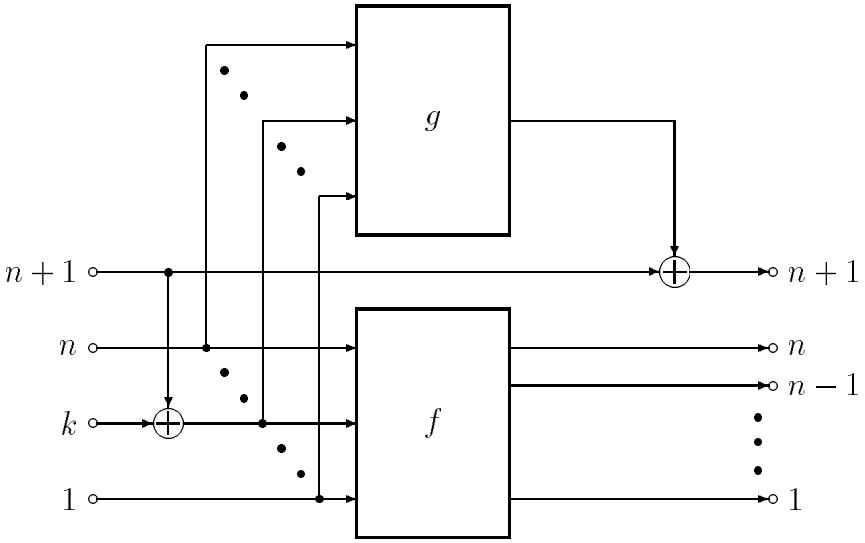

Figure 3.3: Construction method using f and g ($1 \leq k \leq n$).

Remark: Define $f_i: Z_2^n \to Z_2$ $(i = 1, 2, \ldots, n)$ by $f(\mathbf{x}) = (f_n(\mathbf{x}), f_{n-1}(\mathbf{x}), \ldots, f_1(\mathbf{x}))$ from the *bijection* $f: Z_2^n \to Z_2^n$ satisfying the SAC. Noting that f_i satisfies the SAC, **Theorem 3.11** tells us that given a *bijection* $f: Z_2^n \to Z_2^n$ that satisfies the SAC we can construct a *bijection* $\mathbf{E}^k[f_i, f]: Z_2^{n+1} \to Z_2^{n+1}$ satisfying the SAC using only f as illustrated in Fig. (3.4). □

Example 3.7 *When a strong S-box* $g: Z_2^3 \to Z_2$ *is [1,0,0,0,1,1,0,1] and a bijective strong S-box* $f: Z_2^3 \to Z_2^3$ *is [3,1,4,0,2,5,6,7],*

$$< \mathbf{D}_1^1[g] >= [1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1],$$

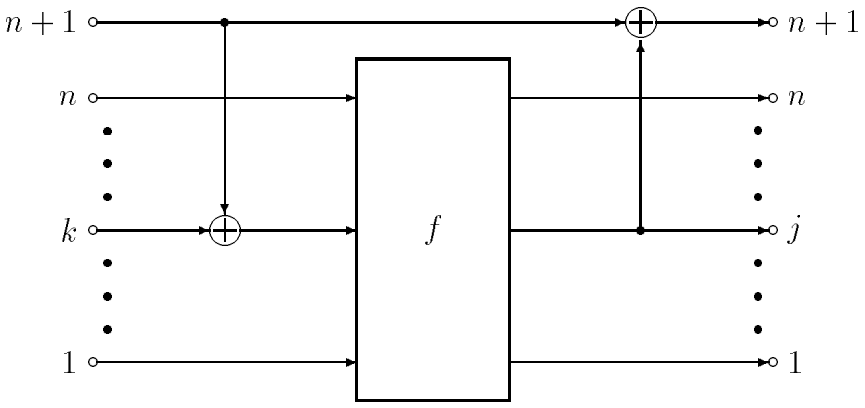

Figure 3.4: Construction method using only f ($1 \leq k, j \leq n$).

and

$$< \mathbf{D}_0^1[f] >= [3,1,4,0,2,5,6,7,1,3,0,4,5,2,7,6].$$

By **Theorem 3.11**, *we can get a strong bijective S-box :*

$$< \mathbf{E}^1[g,f] >= [11,1,4,0,10,13,6,15,9,3,8,12,5,2,7,14].$$

By applying **Theorem 3.11** recursively, we can get 6-bit input *bijective* strong S-boxes,

$$[4,53,16,57,43,45,2,6,12,55,63,33,8,26,30,51,$$
$$37,20,41,0,61,59,22,18,39,28,49,47,10,24,35,14,$$
$$21,36,25,48,13,11,38,34,23,44,1,31,58,40,19,62,$$
$$52,5,32,9,27,29,50,54,60,7,15,17,56,42,46,3],$$

and

$$[36,21,48,57,43,45,2,38,12,23,63,1,8,58,30,19,$$
$$37,20,9,0,29,27,22,50,39,60,49,15,10,56,35,46,$$
$$53,4,25,16,13,11,6,34,55,44,33,31,26,40,51,62,$$
$$52,5,32,41,59,61,18,54,28,7,47,17,24,42,14,3].$$

Therefore, we can generate arbitrary input size *bijective* strong S-boxes if we find 3-bit input *bijective* strong S-boxes.

3.4.3 Bijective Functions Satisfying the Maximum Order SAC

An n bit bijective function ensures that all possible 2^n n-bit input vectors will map to distinct output vectors (*i.e.*, the S-box is a permutation of the integers from 0 to $2^n - 1$) and ensures that all output bits appear once. This is a necessary condition for invertibility of the S-box which may or may not be important depending on the structure of the DES-like cryptosystem.

We can conceive that if there is a statistical dependence between some number of input bits fixed to a constant with the remaining input and output bits of S-boxes, a cryptanalyst will use this relationship to break a cryptosystem.

Therefore, to prevent this possible way of cryptanalysis, we propose the construction method of bijective S-boxes satisfying the maximum order SAC in this Section.

First we review the proposed construction methods of Boolean function satisfying the maximum order SAC.

[Method F]

Forré [23] proposed one recursive construction method which can generate Boolean functions satisfying the maximum order SAC. Her method can generate, for example, 3-bit input Boolean functions $g^{(3)}(\mathbf{x})$ satisfying the maximum SAC from the basic functions $f_i^{(2)}(\mathbf{x}), f_j^{(2)}(\mathbf{x})$ satisfying the SAC by applying Eq. (3.15) recursively.

Let $f^{(n)}: Z_2^n \to Z_2$ and $g^{(n+1)}: Z_2^{n+1} \to Z_2$,

$$g^{(n+1)}(\mathbf{x}) \;=\; x_{n+1} \cdot f_i^{(n)}(\mathbf{x}) + \overline{x_{n+1}} \cdot f_j^{(n)}(\mathbf{x}) \tag{3.15}$$

for integers $i, j \in \{1, 2, \ldots, 2^{n+1}\}$.

This method can generate all Boolean functions satisfying the maximum order SAC, but it is necessary to check whether an enlarged function satisfies the maximum order SAC or not.

[Method A]

Adams and Tavares [2] proposed one structured method where an enlarged Boolean function always satisfies the maximum order SAC. They utilized the idea of the construction method [80] of bent functions.

Let $f^{(n)}: Z_2^n \rightarrow Z_2$ represent a Boolean function satisfying the (n-2)-th order SAC. Furthermore, let $f_r^{(n)}$ be the bitwise reversal of $f^{(n)}$ (*i.e.*, if $< f^{(n)} >= [b_0, b_1, \ldots, b_{2^n-1}]$, $< f_r^{(n)} >= [b_{2^n-1}, b_{2^n-2}, \ldots, b_0]$) and let $\overline{f_r^{(n)}}$ be the bitwise complement of $f_r^{(n)}$.

If n is even,

$$g_1^{(n+1)} = f^{(n)} || f_r^{(n)} \tag{3.16}$$

$$g_2^{(n+1)} = f^{(n)} || \overline{f_r^{(n)}} \tag{3.17}$$

are two distinct (n+1)-bit input Boolean functions satisfying the (n-1)-th order SAC, and

$$\left.\begin{array}{rcl} h_1^{(n+2)} & = & f^{(n)} || f_r^{(n)} || \overline{f_r^{(n)}} || f^{(n)} \\ h_2^{(n+2)} & = & f^{(n)} || \overline{f_r^{(n)}} || f_r^{(n)} || f^{(n)} \\ h_3^{(n+2)} & = & f_r^{(n)} || f^{(n)} || f^{(n)} || \overline{f_r^{(n)}} \\ h_4^{(n+2)} & = & \overline{f_r^{(n)}} || f^{(n)} || f^{(n)} || f_r^{(n)} \end{array}\right\} \tag{3.18}$$

are four distinct (n+2)-bit input Boolean functions satisfying the n-th order SAC.

This construction method can be applied recursively to easily create Boolean functions satisfying the (n-2)-th order SAC for any n. Also, it was proved [2] that all Boolean functions satisfying the (n-2)-th order SAC are bent for even n. Since bent functions are 0/1 unbalanced, we can see that all Boolean functions satisfying the maximum order SAC are unbalanced for even n.

By using this method, when the number of input is 3 (or 4), we give all Boolean functions satisfying the maximum order SAC in Table 3.6 (or Table 3.7).

However, these two methods did not suggest how to construct a bijective function satisfying the maximum order SAC. We can extend these two methods to generate the bijective function satisfying the maximum order SAC.

From the 0/1 unbalance of bent functions and **Method A**, the following theorem is easily obtained.

Theorem 3.12 *Let n denote the number of input of a Boolean function and $n \geq 2$. When n is odd, half of all functions satisfying the maximum order SAC are balanced and the other half are unbalanced. Therefore, when n is odd, the total number of the balanced Boolean functions satisfying the maximum order SAC is 2^n. When n is even, all functions satisfying the maximum order SAC are unbalanced.*

Table 3.6: All 3-bit input Boolean functions satisfying the 1-st order SAC

No.	0123 4567
1	0001 1000(18)
2	0001 0111(17)
3	0010 0100(26)
4	0010 1011(2B)
5	0100 0010(42)
6	0100 1101(4D)
7	1000 0001(81)
8	1000 1110(8E)
9	0111 1110(7E)
10	0111 0001(71)
11	1011 1101(BB)
12	1011 0010(B2)
13	1101 1011(DB)
14	1101 0100(D4)
15	1110 0111(E7)
16	1110 1000(E8)

Table 3.7: All 4-bit input Boolean functions satisfying the 2-nd order SAC

No.	$01 \sim 15$	No.	$01 \sim 15$
1	(188E)	17	(7118)
2	(1871)	18	(71E7)
3	(1781)	19	(7E17)
4	(177E)	20	(7EE8)
5	(244D)	21	(B224)
6	(24B2)	22	(B2DB)
7	(2B42)	23	(BD2B)
8	(2BBD)	24	(BDD4)
9	(422B)	25	(D442)
10	(42D4)	26	(D4BD)
11	(4D24)	27	(DB4D)
12	(4DDB)	28	(DBB2)
13	(8117)	29	(E881)
14	(81E8)	30	(E87E)
15	(8E18)	31	(E78E)
16	(8EE7)	32	(E771)

Proof: Let F_n^{max}, $F_{n:odd}^{max}$, and $F_{n:even}^{max}$ respectively denote any n-bit input, odd-bit input, and even-bit input Boolean function satisfying the maximum order SAC.

For even n, it is known [63] [55] that each bent function has Hamming weight $2^{n-1} \pm 2^{n/2-1}$ (*i.e.*, unbalanced). So any bitwise complement of a bent function has Hamming weight $2^{n-1} \mp 2^{n/2-1}$.

(Case 1 : $F_{n:odd}^{max}$) We can see that the $(n+1)$-bit input function $g_1^{(n+1)}$ expanded by Eq. (3.16) has Hamming weight

$$2^{n-1} \pm 2^{n/2-1} + 2^{n-1} \pm 2^{n/2-1} \quad = \quad 2^n \pm 2^{n/2}.$$

Moreover, the $(n+1)$-bit input function $g_2^{(n+1)}$ expanded by Eq. (3.17) has Hamming weight

$$2^{n-1} \pm 2^{n/2-1} + 2^{n-1} \mp 2^{n/2-1} \quad = \quad 2^n.$$

Since any $F_{n:odd}^{max}$ can be expressed by $g_1^{(n+1)}$ or $g_2^{(n+1)}$, the number of balanced $F_{n:odd}^{max}$ equals the number of unbalanced $F_{n:odd}^{max}$. Moreover, since Lloyd [44] proved that the number of $F_n^{max} = 2^{n+1}$, the number of balanced $F_{n:odd}^{max} = 2^n$.

(Case 2 : $F_{n:even}^{max}$) All 4 $(n+2)$-bit input functions $h_i^{(n+2)}$ (for $i = 1, 2, 3, 4$) expanded by Eq. (3.18) have Hamming weight

$$2^{n-1} \pm 2^{n/2-1} + 2^{n-1} \mp 2^{n/2-1} + 2^{n-1} \pm 2^{n/2-1} + 2^{n-1} \pm 2^{n/2-1} \quad = \quad 2^{n+1} \pm 2^{n/2}.$$

Since any $F_{n:even}^{max}$ can be expressed by $h_1^{(n+2)}$, $h_2^{(n+2)}$, $h_3^{(n+2)}$, or $h_4^{(n+2)}$, all $F_{n:even}^{max}$ are unbalanced. We complete the proof. □

Thus, in order to obtain a bijective function satisfying the maximum order SAC, the number of input bits should be odd.

[Method K]

By using **Theorem 3.12**, we can generate all bijective functions satisfying the maximum order SAC. In order that n Boolean functions $f_1, f_2, \ldots, f_n$ satisfying the maximum order SAC become an n-bit bijective function, the following equation gives as a necessary and sufficient condition that n combined functions must be bijective and satisfying the maximum order SAC.

$$wt(\bigoplus_{i=1}^{n} a_i f_i) = 2^{n-1} \tag{3.19}$$

where $a_i \in \{0,1\}$ and $(a_1, a_2, \ldots, a_n) \neq (0, 0, \ldots, 0)$. This equation was described in Section 3.2.4 too.

In other words, the Hamming weight of any nonzero linear combinations of f_i should be 2^{n-1}. The verification of Eq. (3.19) requires $\binom{n}{2} + \binom{n}{3} + \ldots + \binom{n}{n}$ checks.

Therefore, after selecting n functions from a set of n-bit input (balanced) Boolean functions satisfying the maximum order SAC, we can generate all bijective functions satisfying the maximum order SAC by simply checking Eq. (3.19). **Method K** can be illustrated as Fig. (3.5).

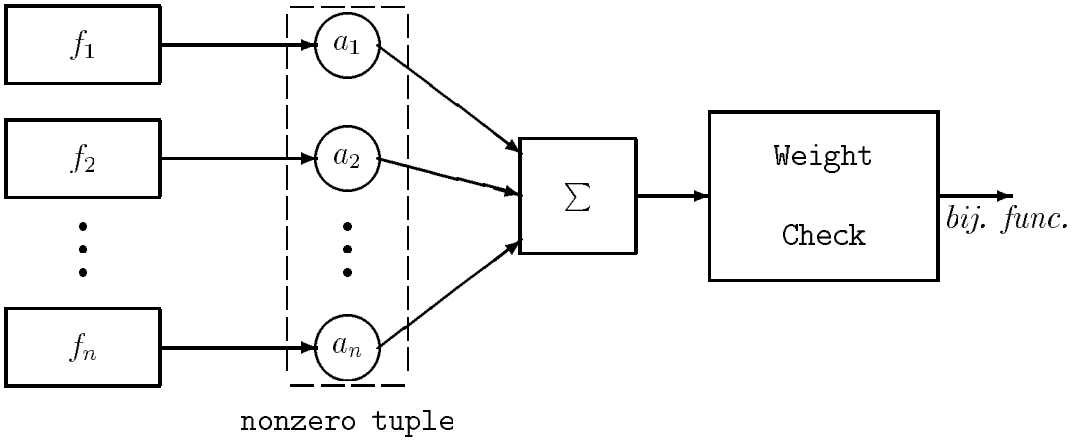

Figure 3.5: Illustrative description of **Method K**

When the number of input is 3, we give the practically generated results by employing **Method K**. The total number of balanced Boolean functions satisfying the 1-*st* order SAC is $2^n = 2^3 = 8$.

Let us first consider the combinatorial search by using **Method K**. We searched $\binom{8}{3} = 56$ combinatorial cases and obtained 32 3-bit bijective functions satisfying the 1-*st* order SAC. Table 3.8 shows examples of generated 3-bit bijective function by this method.

Next we tested all $8^3 = 512$ cases exhaustively and obtained 192 3-bit bijective functions satisfying the 1-*st* order SAC. The total number of 3-bit bijective functions satisfying the 1-*st* order SAC is verified to be identical with that of filtering from all 8!

Table 3.8: Combinatorial search result of 3-bit bijective functions satisfying 1-*st* order SAC

No.	0 1 2 3 4 5 6 7	No.	0 1 2 3 4 5 6 7
1	0 1 2 4 3 5 6 7	17	3 0 1 5 2 6 7 4
2	0 1 3 5 2 4 6 7	18	3 0 5 1 6 2 7 4
3	0 3 1 5 2 6 4 7	19	3 1 0 5 2 7 6 4
4	0 3 5 1 6 2 4 7	20	3 1 2 7 0 5 6 4
5	1 0 2 4 3 5 7 6	21	3 1 5 0 7 2 6 4
6	1 0 3 5 2 4 7 6	22	3 1 7 2 5 0 6 4
7	1 2 0 4 3 7 5 6	23	3 5 0 1 6 7 2 4
8	1 2 3 7 0 4 5 6	24	3 5 1 0 7 6 2 4
9	1 2 4 0 7 3 5 6	25	3 5 6 7 0 1 2 4
10	1 2 7 3 4 0 5 6	26	3 5 7 6 1 0 2 4
11	1 3 0 5 2 7 4 6	27	3 7 1 2 5 6 0 4
12	1 3 2 7 0 5 4 6	28	3 7 5 6 1 2 0 4
13	1 3 5 0 7 2 4 6	29	7 1 2 3 4 5 6 0
14	1 3 7 2 5 0 4 6	30	7 1 3 2 5 4 6 0
15	1 7 2 3 4 5 0 6	31	7 3 1 2 5 6 4 0
16	1 7 3 2 5 4 0 6	32	7 3 5 6 1 2 4 0

permutation. By the same way, we can generate all 5-bit bijective functions satisfying the 3-*rd* order SAC by combinatorial or exhaustive search from 5-bit input balanced Boolean function satisfying the 3-*rd* order SAC. The following functions are generated examples of 5-bit bijective functions satisfying the 3-*rd* order SAC.

$$[3, 4, 11, 19, 7, 31, 16, 23, 25, 1, 14,$$
$$9, 2, 5, 10, 18, 13, 21, 26, 29, 22,$$
$$17, 30, 6, 8, 15, 0, 24, 12, 20, 27, 28],$$
$$[7, 9, 2, 19, 0, 17, 26, 20, 16, 1, 10,$$
$$4, 8, 6, 13, 28, 3, 18, 25, 23, 27,$$
$$21, 30, 15, 11, 5, 14, 31, 12, 29, 22, 24],$$
$$[15, 1, 2, 19, 6, 23, 20, 26, 0, 17, 18,$$
$$28, 22, 24, 27, 10, 21, 4, 7, 9, 3,$$
$$13, 14, 31, 5, 11, 8, 25, 12, 29, 30, 16].$$

Also by using this method, we found that the total number of 5-bit bijective functions satisfying the maximum order SAC is 10,321,920.

3.5 Classification of S-boxes Satisfying the SAC

3.5.1 Boolean Functions

Let us first define that a Boolean function is balanced/unbalanced as:

Definition 3.15 (Balance, Unbalance) *For a function $f : Z_2^n \to Z_2$ is balanced if*

$$\sharp f^{-1}(0) = \sharp f^{-1}(1).$$

By computer search, we investigated the distribution of balanced and unbalanced Boolean functions satisfying any order SAC as shown in Table 3.9. This Table supports **Theorem 3.12** discussed in the previous section. Since the function satisfying the 1-st SAC satisfies the 0-th order SAC by the definition of SAC, we did not count twice a function satisfying a higher order SAC in Table 3.9. For example, when $n = 4$, there are 4128 Boolean functions satisfying the 0-th order SAC. Although there exists only a limited number of Boolean functions satisfying the SAC, we can construct many bijective functions satisfying the SAC from balanced Boolean functions by checking Eq. (3.19).

Table 3.9: Distribution of Boolean functions

n	2			3			4			5		
	B	U	S	B	U	S	B	U	S	B	U	S
No SAC	6	2	8	38	154	192	11502	49906	61408	?	?	?
0-th SAC	0	8	8	24	24	48	1152	2656	3808	?	?	?
1-st SAC	-	-	-	8	8	16	216	72	288	?	?	?
2-nd SAC	-	-	-	-	-	-	0	32	32	?	?	?
3-th SAC	-	-	-	-	-	-	-	-	-	32	32	64
Total	6	10	16	70	186	256	12870	52666	65536	?	?	*

B : Balanced
U : Unbalanced
S : Sum
* : 4.2×10^9

3.5.2 Bijective Functions

If we implement a bijective function satisfying the SAC for an S-box in a block cipher algorithm, it is sometimes required to decrypt the ciphertext by using the inverse of it. So when we make an inverse of a bijective function satisfying the SAC, it is interesting to note whether the inverse function satisfies the SAC or not. Let us define the unidirectional, bidirectional and self-bidirectional SAC function.

Definition 3.16 *For a given f satisfying the SAC, when f^{-1} also satisfies the SAC, we say f is bidirectional SAC. If not, f is unidirectional SAC. In particular, when $< f >=< f^{-1} >$, we say f is self-bidirectional SAC.*

That is, if f is self-bidirectional SAC, and in addition both f itself and its inverse function f^{-1} satisfy the SAC, the function table f and f^{-1} are the same. We give here by computer search the distribution of bidirectional or self-bidirectional bijective functions satisfying the SAC when the number of input bits is 3. We can see that about 10 % of all the 3-bit bijective functions satisfy the SAC and 192 bijective functions satisfy the 1-st order SAC.

Table 3.10: Distribution of *bijective* functions($n = 3$).

Class		Cases
No SAC(A)		35712
0-th SAC	bidirectional	2144
	self-bidirectional	104
	unidirectional	2168
Sum(B)		4416
1-st SAC	bidirectional	160
	self-bidirectional	32
	unidirectional	0
Sum(C)		192
(A)+(B)+(C)		40320

Note that all 3-bit bijective functions satisfying the 1-st order SAC are bidirectional SAC. In **Appendix C**, we listed all 3-bit bijective functions satisfying the 1-st order SAC. This table will aid us in enumerating [51] bijective functions satisfying the SAC.

We conclude this Chapter affirming that there are many simple methods to construct Boolean (or bijective) functions satisfying the (higher order) SAC. Every construction method has been strictly proved. All functions satisfying the SAC may also be useful to implement as a memoryless nonlinear combiner discussed in Chapter 2. Thus, this Chapter will be helpful in designing block cipher and stream cipher algorithms as well.

Chapter 4

Design and Evaluation of Substitution Boxes

We describe here a practical design methodology of S-boxes for symmetric cryptosystems and evaluate their cryptographic properties. Since a DES S-box is designed by 6-bit input and 4-bit output table lookup memory, we choose to design an S-box with the same number of input and output bits as DES S-boxes. Moreover, a DES S-box consists of four rows and each row is a permutation of 0 to 15, *i.e., bijective*. Thus, throughout this Chapter an S-box is usually referred to as a 6×4 lookup memory unless otherwise specified. We often call our designed S-boxes "DES-like S-boxes" to distinguish them from DES S-boxes.

4.1 Design Goal

4.1.1 Boolean Functions

In order to construct a 4-bit bijective function, we need to choose 4 Boolean functions as a first step. We limit ourselves by considering the following 4 cases for selecting a 4-bit input Boolean function in order to make the design criteria of S-boxes clear.

1. Boolean functions satisfying the 0-th order SAC
2. Boolean functions satisfying the 1-st order SAC
3. Boolean functions satisfying the 2-nd order SAC
4. Bent functions

Remember that the distribution of Boolean functions satisfying the SAC is given in Table 3.9.

We can consider that Boolean functions satisfying the maximum order SAC can be used as a basic building block for a 4-bit bijective function. However, since all 4-bit input Boolean functions satisfying the 2-nd SAC are 0/1 unbalanced, we cannot construct a 4-bit input bijective function. By the same reason, we can not construct a 4-bit bijective function by using 4-bit input bent functions. (Note that unbalanced Boolean functions satisfying the maximum order SAC are bent for an even number of input.)

Therefore, we choose to use 4-bit input balanced Boolean functions satisfying the 0-th order or the 1-st order SAC.

4.1.2 DES-like S-boxes

The following design criteria [10] of the DES S-boxes are well known.

D1 No S-box is a linear or affine function of its input.

D2 Each row is bijective.

D3 Changing one bit in the input of an S-box results in changing at least two output bits.

D4 The S-boxes are chosen to minimize the difference between the number of 1's and 0's when any single input bit is held constant.

D5 $S(\mathbf{x})$ and $S(\mathbf{x} \oplus (001100))$ differ in at least two bits.

D6 $S(\mathbf{x}) \neq S(\mathbf{x} \oplus (11ef00))$ for any choice of e and f.

In the open literature, the entropy drop of DES S-boxes and the linear structure of DES have been already pointed out in [81] and [68] [21] respectively. However, no one could claim that these properties have any direct linkage in breaking DES. So, we did not consider these criteria in designing DES-like S-boxes except **D1** and **D2.**

If we think of an S-box as consisting of 2^r permutations on the set of all n-bit vectors (Note that a DES S-box has a value of n=4 and r=2), then a further possible criterion seems natural. In DES, key bits are used to select which permutation is used, and so it

might be desirable to make permutations as different from one another as possible. This condition requires that two indexing entries in the same column of any two permutations disagree with each other. We call this condition "column constraint".

Thus, we try to construct DES-like S-boxes based on the following criteria:

1. Neither linear nor affine.
2. Each row is bijective.
3. Using Boolean functions satisfying the 0-th (or 1-st) order SAC.
4. Meets column constraint.
5. Resistance against differential attack described in the next Section.

4.2 Resistance against Differential Attack

Biham and Shamir [8] proposed the differential cryptanalysis for DES-like cryptosystems. This method analyses the effect of particular differences in plaintext pairs on the differences of the resulting ciphertext pairs. These differences can be used to assign probabilities to the possible keys and locate the most probable key. Their method usually works on many pairs of plaintexts with the same particular difference using only the resultant ciphertext pairs. For DES-like cryptosystems, the difference is chosen as a fixed XORed value of the two plaintexts.

The S-boxes in existing DES-like cryptosystems are known to be nonlinear. Knowledge of the XOR of the input pairs cannot guarantee knowledge of the XOR of the output pairs. Usually, several output XORs are possible. A special case arises when both inputs are equal, in which case both outputs must be equal too. However, a crucial observation is that for any particular input XOR not all the output XORs are possible (*i.e.*, the possible ones do not appear uniformly), and some XORed values appear much more frequently than others.

In DES any S-box has $64 \cdot 64$ possible input pairs, and each one of them has an input XOR and an output XOR. There are only $64 \cdot 16$ possible tuples of input and output XORs. Therefore, each tuple appears on average in 4 pairs. However, not all the tuples exist as a result of a pair, and the existing ones do not have a uniform distribution. So,

the important properties of S-boxes are derived from the analysis of the tables that show its particular distribution—called the pairs XOR distribution—defined as follows:

Definition 4.1 *A table that shows the distribution of the input XORs and output XORs of all possible pairs of an S-box is called the pairs XOR distribution table of the S-box. In this table, each row corresponds to a particular input XOR, each column corresponds to a particular output XOR and the entries in the table count the number of possible pairs with such an input XOR and output XOR.*

Each line in a pairs XOR distribution table contains 64 possible pairs in 16 different entries. Thus in each line of the table, the average of the entries is exactly 4.

Since differential cryptanalysis makes a direct use of the pairs XOR distribution in any S-box of DES-like cryptosystems, we must take the possibility of this attack of any block cipher algorithm into consideration.

We first consider how many entries appear in the pairs XOR distribution–% of entry.

$$\%\ \ of\ \ entry\ \ =\ \ \frac{nz}{16 \times 64} \times 100 \tag{4.1}$$

where nz denotes the number of non-zero entries in the pairs XOR distribution tables.

Second, we compute to compare the values of all entries with the ideal (uniform) value of entry which equals 4. We can check the standard deviation, σ of all entries by

$$\sigma\ \ =\ \ \sqrt{\sum_{i=0}^{63}\sum_{j=0}^{15}(e_{ij}-4)^2/64 \times 16.} \tag{4.2}$$

where e_{ij} is a measured number of entry in pairs XOR distribution table.

Third, we check the nontrivial[1] maximum value of entries in DES S-boxes since the relatively higher valued entries are directly utilized in the differential attack.

We simply call these three parameters "differential characteristics of an S-box". We investigated the differential characteristics of DES S-boxes as shown in Table 4.1.

4.3 Construction of DES-like S-boxes

Recently, Forré proposed statistical methods [24] [25] for designing S-boxes by checking her defined statistical independence criteria between Boolean functions forming an S-

[1]It is trivial that the entry always has a value of 64 when the input XOR and output XOR of any DES-like S-box are zero.

Table 4.1: Differential characteristics of DES S-boxes

Box	% of entry	σ	max. entry
S1	79.49	3.76	16
S2	78.61	3.83	16
S3	79.69	3.78	16
S4	68.55	4.18	16
S5	76.56	3.86	16
S6	80.47	3.69	16
S7	77.25	3.95	16
S8	77.15	3.82	16

box. But she did not consider the weight constraint of Boolean functions forming an S-box and the column constraint of DES-like S-boxes.

We attempt to design DES-like S-boxes as simple as possible by using Boolean functions satisfying the SAC and make the design criteria of them clear.

In order to consider the differential characteristics of DES-like S-boxes, we choose two threshold parameters μ and λ. μ corresponds to % of entry and λ corresponds to maximum value of entry in our construction method. Intuitively, as more entries appear uniformly in the pairs XOR distribution table of a DES-like S-box, the differential cryptanalysis becomes more and more difficult. Also, since a higher value of μ and a lower value of λ than those of DES are desirable, we set the threshold value $\mu_{DES} = 80$ and $\lambda_{DES} = 16$. Afterwards, Boolean functions are restricted to have 4-bit number of inputs. We here summarize our method to construct DES-like S-boxes as follows:

step 1 Set m=4.

step 2 Initialize i=1 and j=1.

step 3 Choose randomly a candidate Boolean function satisfying the 0-th (or 1-st) order SAC.

step 4 Check weight constraint (*i.e.*, Eq. (3.19)) of linear combinations of candidate Boolean functions. If satisfied then $i = i + 1$ else goto **step 3.**

step 5 If $i < m$ goto **step 3** else save a row of a DES-like S-box and $j = j + 1$.

step 6 If $j < m$, goto **step 2.**

step 7 Check column constraint. If not satisfied, goto **step 2.**

step 8 Compute μ and λ. If $\mu > \mu_{DES}$ and $\lambda \leq \lambda_{DES}$ then goto **step 2** else output a DES-like S-box.

When we choosed Boolean functions satisfying the 1-st order SAC in **step 3** of the above method, we found by experiment that $\mu_{1st-SAC}$ ranges from 73 to 80. Also, we experimentally checked μ by using bent functions in **step 3** too. But μ_{bent} ranges from 35 to 60 (Note that each row of an S-box is not bijective in these two cases.). These experiments indicate to us that any function satisfying one of the design criteria is not always good from other points of view. So we found again that it is better to use Boolean functions satisfying the 0-th order SAC.

In practice, after preparing 10,000 bijective functions by choosing 1152 balanced Boolean functions satisfying the 0-th order SAC randomly at **step 6** of this method, we constructed 32 DES-like S-boxes with an engineering workstation (SONY) in a few minutes.

4.4 Comparison with DES S-boxes

We compare the quantitative characteristics of constructed S-boxes with DES S-boxes according to various points of view and evaluate their goodness-of-fit. Since 8 S-boxes are used in DES, we choose 8 typical DES-like S-boxes constructed by our method for comparison. We listed 8 DES-like S-boxes in **Appendix D** and other constructed examples of DES-like S-boxes in **Appendix E**.

The nonlinearity of 4 Boolean functions: $Z_2^6 \rightarrow Z_2$ consisting of an S-box in DES S-boxes and our designed S-boxes are compared in Tables 4.2. In the output bit column of this Table, 4 denotes the most significant location of an output vector and 1 denotes the least significant location of an output vector. For the sake of simplicity, we call our designed S-boxes s^2DES S-boxes.

We can see that the nonlinearity of DES S-boxes ranges from 14 to 22, but the nonlinearity of s^2DES S-boxes ranges from 18 to 24. Thus, it can be said that s^2DES

Table 4.2: Comparison of nonlinearity of DES and s^2DES S-boxes

Box	DES				s^2DES			
	output bit				output bit			
	1	2	3	4	1	2	3	4
S1	18	20	22	18	22	20	20	22
S2	22	20	18	18	24	22	22	22
S3	18	22	20	18	20	24	22	22
S4	22	22	22	22	20	22	22	22
S5	22	20	18	20	22	24	22	24
S6	20	20	20	20	22	22	20	22
S7	18	22	14	20	22	20	22	18
S8	22	20	20	22	22	22	22	22

S-boxes are more nonlinear than DES S-boxes. In this case the maximum nonlinearity equals 26.

We also measured the differential characteristics of s^2DES S-boxes as shown in Table 4.3.

Table 4.3: Differential characteristics of s^2DES S-boxes

Box	% of entry	σ	max. entry
S1	84.38	3.44	14
S2	85.25	3.39	14
S3	84.38	3.34	14
S4	83.40	3.54	16
S5	82.91	3.57	16
S6	83.98	3.48	16
S7	81.93	3.62	16
S8	82.81	3.54	16

s^2DES S-boxes are found to exhibit better differential characteristics than those of DES S-boxes (Table 4.1). This suggests that s^2DES has stronger resistance against differential cryptanalysis than DES.

Moreover, we checked the dependence matrix of a DES-like S-box. The dependence matrix $\mathbf{P} = (p_{i,j})$ of the S-box is defined as follows: the element $p_{i,j}$ of $\mathbf{P}$ is the probability that the output variable $\mathbf{y}_j$ of the S-box changes when the input variable $\mathbf{x}_i$ is

complemented. The average values of $(p_{i,j})$ (*i.e.* $(p_{i,1}+p_{i,2}+p_{i,3}+p_{i,4})/4$) of DES S-boxes and s^2DES S-boxes are compared in Table 4.4.

Table 4.4: Comparison of average $(p_{i,j})$ of DES and s^2DES S-boxes

Box	DES	s^2DES
S1	0.620	0.495
S2	0.633	0.510
S3	0.661	0.505
S4	0.615	0.521
S5	0.633	0.516
S6	0.651	0.516
S7	0.656	0.516
S7	0.625	0.508

Whenever one bit of an input in s^2DES S-boxes is complemented, every output bit can be said to change with probability close to $\frac{1}{2}$. But, the probability values of the dependence matrix in DES S-boxes are found to be greater than 0.6. This shows indirectly that most Boolean functions of all DES S-boxes do not satisfy the SAC.

Finally, we compare the average values of cross correlations of avalanche variables, $\rho_{i,j}(k)$ of DES S-boxes with those of s^2DES S-boxes in Table 4.5. See the definition of $\rho_{i,j}(k)$ in Section 3.2.3. In average, the cross correlations between the output bits of s^2DES S-boxes can be said to be more independent than those of DES S-boxes.

Table 4.5: Comparison of average $\rho_{i,j}(k)$ of DES and s^2DES S-boxes

Box	DES	s^2DES
S1	-0.195	-0.051
S2	-0.188	-0.07
S3	-0.165	-0.053
S4	-0.232	-0.083
S5	-0.184	-0.074
S6	-0.183	-0.096
S7	-0.153	-0.105
S7	-0.176	-0.101

The detailed values of the dependence matrix and the cross correlations of avalanche

variables of s^2DES S-boxes are given in **Appendix D**.

Therefore, we can construct many DES-like S-boxes by using Boolean functions satisfying the 0-th order SAC. Compared with DES S-boxes, our designed DES-like S-boxes exhibit better cryptographic properties from various points of view within our experiments.

In the next Chapter, we shall check and compare the overall security of s^2DES with other block cipher algorithms by experiment.

Chapter 5

Analysis of Symmetric Cryptosystem Using Substitution Boxes

5.1 Statistical Properties

If a cryptosystem has any statistical weakness within a smaller number of pairs of plaintext and ciphertext, the cryptosystem can be said to be badly designed. Based on the statistical hypothesis testing, this Section presents the partial input-output dependence test and randomness test of DES-like cryptosystems including s^2DES and compares their statistical properties. We call s^2DES a DES-like cryptosystem in which only S-boxes in DES are replaced by our designed S-boxes listed in **Appendix D.**

5.1.1 Partial Input-output Dependence Test

Let a plaintext $\mathbf{x} = (x_0, x_1, \ldots, x_{r-1})$ be enciphered by any DES-like cryptosystem F with a key $\mathbf{k} = (k_0, k_1, \ldots, k_{s-1})$ into a ciphertext $\mathbf{y} = (y_0, y_1, \ldots, y_{r-1})$ as

$$\mathbf{y} = F(\mathbf{k}, \mathbf{x}).$$

In DES and s^2DES $r = 64$ and $s = 56$, but $r = s = 64$ for FEAL and Multi2. Each output bit y_i $(0 \leq i \leq r-1)$ is a function of the l input bits x_i $(0 \leq i \leq r-1)$ and m key bits k_i $(0 \leq i \leq s-1)$. We want to test if one of the following conditions holds.

- Some set of output bits are dependent on some set of input bits for a fixed key.
- Some set of output bits are dependent on some set of key bits for a fixed plaintext.

Table 5.1: Average of χ^2 statistics

	FEAL-4	FEAL-8	Multi2	DES	s^2DES
Fixed key	65.43	61.10	65.63	62.01	60.19
Fixed plaintext	62.47	57.97	61.57	69.40	64.99

$N_{in} = \{0, 1, 2\}$, $N_{out} = \{0, 1, 2\}$, $N_s = 500$, and 20 trials

Dependence might be used to estimate the key or plaintext. For example if k_i is dependent on some set of output or input bit positions, we could make this the basis of the recovery of k_i from the corresponding plaintext and ciphertext. There is dependence, of course, if a subset is the set $\{0, 1, \ldots, r-1\}$ of all input/output bit positions. What we intend to check is dependence for small subsets.

χ^2-test

The procedure of this test is as follows: At first, with one of two inputs (plaintext or key) fixed, we decide the subset (N_{in}) of input space and the subset (N_{out}) of output space. Next, we specify the null hypothesis $H0$: *there is no dependence between N_{in} and N_{out}*. After we compute the acceptance region at the given level of confidence, we count the number of pair occurrences, $Y_{i,j}$, between the two subsets with independent and identically distributed input at the predetermined number of sampling N_s. Then, we compute the χ^2-statistic with the degree of freedom, $m = 2^{(|N_{in}|+|N_{out}|)} - 1$ as

$$\chi^2 = \sum_{i=0}^{a} \sum_{j=0}^{b} (Y_{i,j} - N_s p)^2 / N_s p \tag{5.1}$$

where $p = 2^{(|N_{in}|+|N_{out}|)}$, $a = 2^{|N_{in}|} - 1$, and $b = 2^{|N_{out}|} - 1$. We check whether this value exists inside the acceptance region at 1 and 99 % confidence levels. If $m \geq 31$, the approximated χ^2 statistic is computed as

$$\chi_{m,q} \approx m\{1 - 2/9m + x_q\sqrt{2/9m}\}^{1/3} \tag{5.2}$$

where if the level of confidence is 0.01 or 0.99, x_q is -2.33 or 2.33 respectively.

We checked all DES-like cryptosystems: dependence between a subset {0,1,2} of the input block and a subset {0,1,2} of the output block with 500 samples. The average χ^2 statistics of 20 different trials are compared in Table 5.1.

Table 5.2: % of χ^2 statistics in acceptance region

Size of subset	N_s	Condition	FEAL-4	FEAL-8	Multi2	DES	s^2DES
4 bits	2000	A	98.55	98.92	98.24	97.77	98.31
		B	97.73	98.02	98.22	98.82	97.23
8 bits	327800	A	98.41	98.75	98.25	98.12	98.22
		B	98.38	98.71	96.09	98.02	98.74

A : Fixed key, B : Fixed plaintext

Moreover, we expand the size of the subset from 3 bit to both 4 bit and 1 byte and perform the χ^2-test on all pairs of input/output subsets exhaustively. We counted the number of times where χ^2 statistics fell into the acceptance region at the level of confidence 0.01 and 0.99. Table 5.2 shows the percentage of χ^2 statistics in the acceptance region. We may say that the null hypothesis can be partly acceptable for 5 DES-like cryptosystems.

Kolmogorov-Smirnov test

In practice, the acceptance or rejection of the null hypothesis is based on the results of several independent χ^2-test. The evaluation of multiple χ^2-tests is often made using the Kolmogorov-Smirnov test.

Let $X_0, X_1, \ldots, X_{n-1}$ be random variables with independent and identically distributed, and we assume that $H0$ behaves like the continuous distribution:

$$F(x) \;=\; \Pr\{X \leq x\} \tag{5.3}$$

The discrete distribution F_n of multiple χ^2 statistics $X_0, X_1, \ldots, X_{n-1}$ is computed as:

$$F_n(x|X) \;=\; F_n(x|X_0, X_1, \ldots, X_{n-1}) \tag{5.4}$$

By the law of the large numbers, we can confirm that Eq. (5.3) equals Eq. (5.4).

We begin this test by sorting the ascending order of sampled test statistics X_0, X_1, ..., X_{n-1}. Then we compute two Kolmogorov-Smirnov statistics $K_{n,+}$ and $K_{n,-}$ as

$$K_{n,+} \;=\; \sqrt{n} \max_{0 \leq i < n-1} ((i+1)/n - F(X_i|X_0, X_1, \ldots, X_{n-1})) \tag{5.5}$$

Table 5.3: Range of $K_{n,+}$ and $K_{n,-}$

Size of subset		FEAL-4	FEAL-8	Multi2	DES	s^2DES
4 bits	$K_{n,-}$	0.024-1.201	0.313-1.216	0.025-1.155	0.007-1.385	0.022-1.392
	$K_{n,+}$	0.017-1.272	0.142-1.354	0.022-1.417	0.017-1.415	0.047-1.444
8 bits	$K_{n,-}$	0.022-1.326	0.043-1.238	0.032-1.345	0.002-1.483	0.035-1.432
	$K_{n,+}$	0.018-1.489	0.029-1.372	0.029-1.315	0.025-1.446	0.012-1.462

$$K_{n,-} = \sqrt{n} \max_{0 \le i < n-1} (F(X_i | X_0, X_1, \ldots, X_{n-1}) - (i/n)) \tag{5.6}$$

Intuitively, $K_{n,+}$ and $K_{n,-}$ measure the positive and negative deviation of sample distribution function F_n from the actual distribution F at the points $X_0, X_1, \ldots, X_{n-1}$. We can compute F_n like

$$F_n = erf[(X_i - m)/\sqrt{2m}]. \tag{5.7}$$

where erf denotes the error function. When $n = 20$, the acceptance region of $K_{n,-}$ and $K_{n,+}$ is 0.038 and 1.468 respectively when the level of confidence is 0.01 and 0.99.

By using multiple χ^2-statistics of the previous Section, we performed the Kolmogorov-Smirnov test to verify the dependence between subsets of input and output block. The range of $K_{n,-}$ and $K_{n,+}$ are summarized in Table 5.3. Finally, we can accept the null hypothesis that there is no dependence between the checked subsets of input and output block.

5.1.2 Randomness Test

Statistical tests provide us the quantitative measure for randomness of sequence. We need to determine our own levels of confidence for the tests so that we can decide whether a sequence has passed or failed the test. From this section onwards, we shall use a coding scheme to indicate a test condition summarized in Table 5.4. In this Table, pt, ct, and key denote the block of plaintext, ciphertext, and key. pt $\oplus$ ct indicates the bit-by-bit exclusive-or of plaintext and ciphertext block. For example, if a code is 000, we check the randomness of ciphertext with a random fixed key and 4162 fixed-pattern plaintexts. These fixed-patterns are: all zero pattern, patterns with Hamming weight-1 and Hamming weight-2, and their bit-by-bit inverted patterns.

Table 5.4: Summary of a test condition

First Digit (plaintext)	0	Fixed-pattern(4,162 samples)
	1	Random (10,000 samples)
Middle digit (target to test)	0	ct
	1	pt $\oplus$ ct
	2	key $\oplus$ ct
Last digit (key)	0	Random
	1	All zeros
	2	0123456789abcdef

We investigate the randomness properties of FEAL-4, FEAL-8, Multi2, DES, and s^2DES in this Section and the quantitative properties of them in the next Section.

Frequency Test

This is perhaps the most obvious of the tests and is applied to ensure that there is roughly the same number of 0's and 1's. For this we merely compute

$$\chi^2 = \frac{(n_0 - n_1)^2}{n}$$

Clearly if $n_0 = n_1$ then $\chi^2 = 0$, and, in addition, the larger the value of χ^2 the larger the discrepancy between the observed and the expected frequencies. If the value of χ^2 is no greater than 3.841 or 6.635 at 95 % or 99 % level of confidence, we judge the sequence to be random. We counted the percentage of number of one's less than 22 or greater than 42 under a given test condition in Table 5.5.

Table 5.5: Result of frequency test (% of n_1, $n_1 < 22$ or $n_1 > 42$)

Code	DES	FEAL-4	FEAL-8	Multi2	s^2DES
000	0.6968	0.8169	0.7689	0.9130	0.6968
100	0.8800	0.8900	0.7900	0.8400	0.8500
102	0.8100	0.9000	0.9100	0.9600	0.7800
110	0.8000	0.8600	0.6800	0.6500	0.7500

Serial Test

The serial test is used to ensure that the transition probabilities are reasonable:*i.e.*, that the probability of consecutive entries being equal or different is about the same. This will then give us some level of confidence that each bit is independent of its predecessor. Let 00 occurs n_{00} times, 01 occurs n_{01} times, 10 occurs n_{10} times, and 11 occurs n_{11} times. Then, $n_{01} + n_{00} = n_0$ or $n_0 - 1$, $n_{10} + n_{11} = n_1$ or $n_1 - 1$, and $n_{00} + n_{01} + n_{10} + n_{11} = n - 1$.(Note that the -1 occurs because in a section of length n, there are only $n - 1$ transitions.) Ideally we want $n_{01} = n_{10} = n_{00} = n_{11} \approx \frac{n-1}{4}$.

The χ^2 statistic is used to test the above hypothesis where

$$\chi^2 = \frac{4}{n-1}\sum_{i=0}^{1}\sum_{j=0}^{1}(n_{ij})^2 - \frac{2}{n}\sum_{i=0}^{1}(n_i)^2 + 1$$

If this value is greater than 5.991 or 9.210 at the level of confidence of 95 % or 99 %, we judge that the sequence is not random at two degrees of freedom. Tables 5.6 and 5.7 show the results of the serial test at the level of confidence of 95 % and 99 % respectively.

Table 5.6: Result of serial test 1 (% of $\chi^2 > 5.991$)

Code	DES	FEAL-4	FEAL-8	Multi2	s^2DES
000	4.0125	4.8775	4.5171	5.2859	4.7573
100	4.7100	4.7500	4.6700	4.8500	4.9300
102	4.7000	4.7800	4.4800	5.2100	4.9400
110	4.7300	4.8100	4.8700	4.8700	4.7000

Table 5.7: Result of serial test 2 (% of $\chi^2 > 9.21$)

Code	DES	FEAL-4	FEAL-8	Multi2	s^2DES
000	0.9611	0.9851	0.7929	1.3936	1.0572
100	1.0500	1.1100	0.9500	1.1500	1.0400
102	0.9400	1.0800	1.0200	1.0800	1.0400
110	0.9500	1.0600	0.9700	0.9000	0.9500

Runs Test

We divide the binary sequence into blocks (runs of ones) and gaps (runs of zeros). The runs test examines the number of runs for random data. This test is only applied if the sequence has already passed the serial test in which case it is known that the number of blocks and gaps are in acceptable limits. The number of runs is normally distributed with

$$\begin{aligned} mean &= 1 + \frac{2n_0n_1}{n} \\ variance &= \frac{(mean-1)(mean-2)}{n-1} \\ z &= \frac{runs - mean}{\sqrt{variance}} \end{aligned}$$

At the 95 % and 99 % levels of confidence, z scores ± 1.96 and ± 2.58 respectively as shown in Tables 5.8 and 5.9 .

Table 5.8: Result of runs test 1 (% of $|z| > 1.96$)

Code	DES	FEAL-4	FEAL-8	Multi2	s^2DES
000	3.3157	3.5320	3.4358	4.0365	3.3157
100	3.4400	3.6300	3.5300	3.2900	3.7000
102	3.5000	3.5900	3.5300	3.4200	3.2200
110	3.5200	3.5400	3.4200	3.5200	3.6300

Table 5.9: Result of runs test 2 (% of $|z| > 2.58$)

Code	DES	FEAL-4	FEAL-8	Multi2	s^2DES
000	0.6968	0.7208	0.5526	0.9611	0.5526
100	0.6200	0.7200	0.7000	0.6700	0.6900
102	0.5000	0.6500	0.7100	0.5500	0.6800
110	0.6200	0.6100	0.7200	0.7300	0.5400

Since the number of runs is a discrete variable, a continuity correction ± 0.5 was included in the numerator of the standard normal variable z. Absolute values of z, $abs(z)$, were obtained where

$$abs(z) = \frac{abs(abs(runs - mean) - 0.5)}{\sqrt{variance}}$$

5.2 Quantitative Properties

5.2.1 Sequence Complexity Test

The idea of randomness also reflects the impossibility of predicting the next digit of a sequence from all previous ones. An interesting approach to a definition of randomness of finite sequences based on this concept of unpredictability was taken by Solomonov [75] and Kolmogorov [38]. They characterized the "patternlessness" of a finite sequence by the length of the shortest Turing machine program that could generate the sequence. This concept was further developed by Martin-Loef [48].

A different approach to evaluating the complexity of finite sequences was given by Lempel and Ziv [41]. They further employed their definition for the universal data compression [83].

A measure for the complexity of a finite sequence S is given in terms of the number of new patterns which appear as we move along the sequence [43]. This number is $C(S)$ called the sequence complexity of S.

Example

$$\begin{aligned} S &= 100111101100001110 \\ S &= 1/0/01/1110/1100/001110 \\ C(S) &= 6 \end{aligned}$$

Also, Lempel and Ziv have shown that almost all binary sequences of length n have complexity exceeding

$$\frac{n}{\log_2 n}.$$

This value will be used as a threshold of complexity for random sequences. In the case where $n = 64$ this threshold value is $10\frac{2}{3}$. Since the sequence complexity test counts new patterns, the poker test and autocorrelation test from [6] were not included in the statistical tests in the previous section. The poker test counts the number of similar patterns of a chosen length in the block and the autocorrelation test checks for periodicity in the block.

Table 5.10 shows the result of the sequence complexity test. Since this test checks for a new pattern of a 64-bit sequence from left to right, the final pattern happens to occur

in the same pattern which we processed before. In this case, we did not count this final pattern as a new pattern.

Table 5.10: Sequence complexity (% <= 10)

Code	DES	FEAL-4	FEAL-8	Multi2	s^2DES
000	0.7926	0.8650	0.8169	0.9851	0.6968
001	0.7208	0.6728	0.6247	0.9130	0.9370
010	0.8650	0.7689	0.7448	0.8409	0.8650
011	0.7926	0.9851	0.7989	1.0332	1.1293
020	0.8169	0.8890	0.8650	0.9130	0.7689
022	0.9611	0.7929	0.7989	0.9370	0.7500
100	0.6600	0.7300	0.8000	0.7000	0.9300
101	0.8700	0.8300	0.8700	0.9400	0.8500
110	0.8700	0.7500	0.7700	0.8100	0.7700
111	0.9200	0.8300	1.0100	0.8200	0.8900
120	0.8300	0.8200	0.8400	0.8500	0.7500
122	0.9300	0.8000	0.8600	0.7600	0.7500

5.2.2 Binary Derivative Test

Given a string of binary digits, the first derivative is taken by considering each overlapping pair of digits and recording a zero if they are the same and a one if they are different [12]. Every successive binary derivative drops one digit. A sequence of length 16 and its first four derivatives are given below:

1 0 0 0 1 0 0 0 1 1 0 1 0 1 0 1
1 0 0 1 1 0 0 1 0 1 1 1 1 1 1
1 0 1 0 1 0 1 1 1 0 0 0 0 0
1 1 1 1 1 1 0 0 1 0 0 0 0
0 0 0 0 0 1 0 1 1 0 0 0

Let $p(i)$ denote the fraction of ones in the i-th derivatives where $p(0)$ denotes the fraction ones in original sequence. Suppose that k derivatives are evaluated for a string of length n. Let r denote the range of the $p(i)$ and p denote the average of the $p(i)$ where

$$p_{max} = Max\ p(i)$$

Table 5.11: Range of variables for binary derivative test

Attribute	Patterned String	Random Strings
$p(0)$	variable	close to 0.5
p	low	close to 0.5
r	high	low

$$\begin{aligned} p_{min} &= Min\ p(i) \\ r &= p_{max} - p_{min} \\ p &= \sum_{i=0}^{k} \frac{p(i)}{k+1} \end{aligned}$$

By practical experimentation, Carroll and Robbins [12] suggested that we can use the values of $p(0), p$ and r to distinguish between patterned and random strings as shown in Table 5.11.

In order to know how many derivatives we have to evaluate to distinguish between patterned and random sequences of length $n = 64$, it is suggested [30] that seven derivatives is adequate to measure randomness. Moreover, to measure the significance level [30] for p and r, 10,000 random blocks of length 64, 95 % and 99 % confidence intervals for p were determined to be approximated by $0.45 \leq p \leq 0.54$ and $0.43 \leq p \leq 0.55$ respectively. Also, 95 % and 99 % had values of approximately r less than 0.28 and 0.33 respectively. Also, Tables 5.12 and 5.13 show the percentage of average values at the level of confidence 95 % and 99 % respectively.

Tables 5.14 and 5.15 show the percentage of range values at the level of confidence 95 % and 99 % respectively.

Table 5.12: Average of binary derivative test 1 (% of $p < 0.45$ or $p > 0.54$)

Code	DES	FEAL-4	FEAL-8	Multi2	s^2DES
000	4.3008	5.4061	4.4690	4.4930	4.3008
001	4.4930	4.4210	4.3969	4.5171	4.3729
010	4.7814	4.4930	4.2528	4.9015	4.8054
011	4.6132	4.5891	4.3729	4.8534	4.7333
020	4.7573	4.4450	4.4210	4.5651	4.2287
022	4.9736	5.0697	4.4210	4.7333	4.6372
100	4.6700	4.4800	4.7800	4.5400	4.9800
101	4.5100	4.7500	5.0500	4.3100	4.7000
110	4.6900	4.6200	4.7700	4.6500	4.8700
111	4.9000	4.8800	4.8600	4.6200	5.1000
120	4.5700	4.8800	4.7500	4.5400	4.9900
122	4.6900	4.8300	4.6000	4.6500	4.8700

Table 5.13: Average of binary derivative test 2 (% of $p < 0.43$ or $p > 0.55$)

Code	DES	FEAL-4	FEAL-8	Multi2	s^2DES
000	0.7929	1.1773	0.8890	1.0332	1.0332
001	1.1533	1.2013	0.7929	0.9370	1.0812
010	1.1293	0.8650	0.9611	1.1773	0.8409
011	0.8169	1.0572	0.9611	0.8650	0.8409
020	0.7448	0.7689	0.9851	0.7929	0.7448
022	1.1773	1.2254	0.7689	0.9611	0.9851
100	0.9800	1.0200	1.1100	1.0900	1.0700
101	0.9900	1.3000	1.0500	0.8600	0.9400
110	0.9900	0.8500	1.0100	0.8600	1.0600
111	1.0000	0.9500	1.1000	0.9100	1.0300
120	0.9700	1.1300	1.0300	1.1000	1.0000
122	1.0100	1.0300	1.0400	1.0700	1.0300

Table 5.14: Range of binary derivative test 1 (% of $r > 0.28$)

Code	DES	FEAL-4	FEAL-8	Multi2	s^2DES
000	5.0697	5.4541	5.1418	5.4781	5.0457
001	4.7093	4.5411	4.6852	4.8054	4.9015
010	5.3820	4.7093	5.4061	5.2138	4.8775
011	4.8294	4.9736	4.8775	4.6372	5.1418
020	4.9255	4.8534	4.8775	4.6852	4.6372
022	4.8534	4.7093	4.9015	4.6612	5.4301
100	5.0100	5.3100	4.9400	4.7600	5.2700
101	5.1100	4.9100	5.0100	4.9400	4.8500
110	5.0300	4.9400	4.8200	4.6500	4.3200
111	4.8500	5.0400	5.3700	5.0400	5.0300
120	4.9900	4.9600	4.9300	4.8800	4.5800
122	5.0800	4.8800	4.7500	4.9200	4.8500

Table 5.15: Range of binary derivative test 2 (% of $r > 0.33$)

Code	DES	FEAL-4	FEAL-8	Multi2	s^2DES
000	1.0572	1.3215	0.9611	1.4416	1.1293
001	1.1533	0.7448	0.9370	1.3215	1.0332
010	1.2254	1.2975	1.2254	1.3455	1.2494
011	0.9851	1.0812	1.4897	0.9611	1.3455
020	1.2494	1.2494	1.2013	0.9370	0.9370
022	0.8890	1.1052	0.8650	0.6007	0.9370
100	1.0900	1.2000	1.2500	1.3600	1.2400
101	1.1300	1.0700	1.1100	1.1100	0.9500
110	1.0100	1.1800	1.1500	1.1300	0.9900
111	1.3400	1.1100	1.3600	1.1100	1.1600
120	1.0600	1.1200	1.0000	0.9500	0.9900
122	1.1600	1.0600	1.1300	1.0400	1.0800

5.3 Cyclic Properties

In this Section we check the cyclic properties of DES-like cryptosystems. First, we review some previous works about the cyclic or algebraic [14] property of cryptosystems in the open literature. Davies and Parkin [15] and Juenman [33] analyzed the cyclic property of DES assuming that DES is a random permutation [70] or random function [61]. Moore and Simmons [53] [54] reported the cyclic property of special type, (*i.e.* weak, semiweak or palindromic or antipalindromic) keys of DES. On the other hand, Kaliski *et. al.* tried to find the algebraic structure[20] [34] of DES. But no one can find a better cryptanalysis method than the exhaustive search method at this time to the best of our knowledge.

If we can reduce the size of a cryptosystem into any tractable size without disturbing its internal structure, we can easily examine any property of a cryptosystem. We have some evidence of the reducibility of several cryptosystems. For example, the group property of RSA algorithm [62] is invariant even if we reduce the size of modulus which is the product of two large prime numbers greater than 10^{100}.

This brings about a new problem of *Reducibility of Cryptosystem*. We don't have, presently, a satisfactory solution to this problem but we will try an empirical approach such that, if the block size of a cryptosystem could be reduced into half-sized cryptosystem recursively, some checked properties of the cryptosystem would be invariant. There was a similar approach to cryptanalyze DES [13] by reducing the number of rounds. If this method could be proved to be one solution of this problem, it can be of great use to evaluate the security of a cryptosystem.

Putting aside this problem for the moment, we will check the cyclic properties (cycle length,number of disjoint cycles, variation of maximum cycles) of *reduced* cryptosystem when we operate the OFB mode and compare the three known cryptosystems, *i.e.*, FEAL-4, FEAL-8, and Multi2. We could not check the cyclic properties of DES and s^2DES due to the limited 3-year research period.

5.3.1 Background

Due to the pigeonhole principle[1] and the finiteness of any deterministic cryptosystem, any arbitrary value(IV) from finite plaintext space must be repeated after computation

[1] A huge number of pigeons laying eggs in two letter boxes $\Rightarrow$ at least one letter box contains a huge number of angry pigeons.

of a mapping function, g, like Eq. (5.8).

$$g(IV) : E(K, IV) \rightarrow E(K, E(K, IV)). \tag{5.8}$$

where K is a fixed key of a cryptosystem and E is any deterministic cryptosystem and where $E(K, P)$ means encrypt a plaintext P with a key K and make a ciphertext C.

For some integers l and c, we have $l+c$ distinct values of $g^0(IV), g^1(IV), \cdots, g^{l+c-1}(IV)$ but we have $g^{l+c}(IV) = g^l(IV)$. This implies, in turn, that $g^{i+c}(IV) = g^i(x)$ for all $i \geq l$. We call the integer l the leader length, and the integer c the cycle length, and we set $n = l + c$.

If any cryptosystem has a short cycle under OFB mode, the encrypted data are vulnerable to cryptanalysis by passive attackers. We call this attack method the *transmission-in-depth problem.*

5.3.2 Reduction Criteria

As an initial step in solving the *Reducibility of Cryptosystem* problem, we made up some reduction criteria as follows:

- Do not change the number of rounds of the original cryptosystem.
- Keep the internal nonlinearity of the original cryptosystem.
- Reduce the block size as much as possible until we can quickly verify any structure of the cryptosystem.

We reduced the original cryptosystems (FEAL-4,FEAL-8 and Multi2) into half of their original size (*i.e.*, 32 bits) according to the criteria at first, but it was untractable by an engineering workstation. In practice, it takes about one month to find one cycle of any half-reduced cryptosystem. So, we again reduced the latter from 32 bit-block into 16 bit-block cryptosystems and checked the cyclic properties of these cryptosystems.

5.3.3 Cycle Finding Algorithm

Knuth proposed the *hare and tortoise algorithm* [37] to find one cycle of a given function. But, his algorithm needs to check $O(3n)$ computations of a function to find one cycle. We need a more efficient cycle-finding algorithm of a cryptosystem so that we can find

one cycle less than $n + O(1)$ computations. Sedgewick *et al.* [67] proposed an efficient algorithm to find one cycle for small $O(1)$ complexity, but we modified their algorithm to find every cyclic parameter of a cryptosystem like **Algorithm A**.

Algorithm A

input: IV:initial value, $E(K, P)$:finite cryptosystem: $i = 1$;

step1: Make global table which stores all data on ciphertext space.

step2: Find i-th cycle leader and cycle length given IV and $E(K, P)$ based on Sedgewicks' algorithm.

step3: If found, delete ith cycling data in global table.

step4: $i = i + 1$;

step5: Generate new initial value on global table.

step6: Find i-th cycle leader and length with new initial value.

step7: Check if i-th cycling data collides with global table.

step8: If collides, output two data.(to check a bridge point)

step9: Searched all ciphertext space ? If yes, output all statistics or else update global table and goto **step4**.

5.3.4 Results

As previously mentioned, after reducing the block length of FEAL-4, FEAL-8 and Multi2 from 64 bits into 16 bits with the 16-bit OFB operation, we selected 256 random keys from a built-in random number generator routine in the C-programming language library of NEWS engineering workstation.

First we found that all cycles of 3 cryptosystems exist separately. So, we checked the distribution of disjoint cycles shown in Fig. (5.1). Moreover, 3 cryptosystems have even-numbered disjoint cycles and the peak number of disjoint cycles are 10.

When we see the simulation result, we can hardly distinguish the difference of cyclic property of FEAL-4, FEAL-8 and Multi2. The cyclic properties of FEAL family, therefore, do not depend on the number of rounds.

We also checked their maximum cycle lengths in 256 cases as shown in Fig. (5.2).

The maximum cycle length of FEAL-4 and FEAL-8 varies from 16152 to 65473 but that of Multi2 varies from 15760 to 65452. The average cycle of FEAL-4 and FEAL-8

Figure 5.1: Distribution of disjoint cycles

shown in Table 5.16 is 40799, but that of Multi2 is 39817. When we assume that a cryptosystem is a randomly selected permutation, the theoretical average cycle length is about $2^{15} = 32536$.

Table 5.16: Range of maximum cycle length

	FEAL-4	FEAL-8	Multi2
Maximum	65473	65473	65452
Average	40799	40799	39817
Minimum	16152	16152	15760

Therefore, we can say that the three cryptosystems may have the same algebraic structure.

5.3.5 Concluding Remarks

We made an efficient cycle-finding algorithm and checked some cyclic properties (cycle lengths, variation of maximum cycles, numbers of disjoint cycles) of published symmetric cryptosystems whose block length was reduced from 64 bits to 16 bits. We obtained some experimental results of common cyclic properties of these cryptosystems:

- All cycles exist separately,*i.e.*, there are no bridge points between different cycles.
- The checked cyclic properties of FEAL-4, FEAL-8 and Multi2 are almost the same.

(a) FEAL-4

(b) FEAL-8

(c) Multi2

Figure 5.2: Variation of maximum cycle length

- Three cryptosystems have even-numbered disjoint cycles.

This experiment gives us strong evidence that these cryptosystems work like random permutations mathematically. In addition, this experiment can be applicable to decide upon the key-change period of any deterministic cryptosystem.

Chapter 6

Conclusions

In the last two decades, symmetric cryptosystems have been playing a prominent role in information security. With the rapid advance of today's semiconductors and communications technology, it is expected that these cryptosystems undertake the indispensable position in providing information privacy and authentication in modern communications.

In this thesis, after reviewing the basic concepts of symmetric cryptosystems including block and stream ciphers, we described the standard operation modes of block ciphers (*e.g.*, ECB, CFB, CBC and OFB) for their practical applications. For the stream ciphers case, some operation modes like CFB or OFB are equally applicable too.

Since the S(ubstitution)-boxes play an important part in a block cipher, we are required to be able to prove their cryptographic properties. The general cryptographic features of S-boxes for block cipher are considered to have high nonlinearity, low cross correlation, strict avalanche criterion (SAC), bijection, *etc.*

We have shown that the S-boxes satisfying the SAC have the following properties:

1. Neither linear nor affine.

2. All 1-bit or 2-bit input bijective functions never satisfy the SAC.

3. Closed under linear or affine transformation.

These properties of S-boxes satisfying the SAC give us the facts that, firstly, a block cipher consisting of S-boxes satisfying the SAC can be said to be neither linear nor affine. Secondly, when we require bijective S-boxes satisfying the SAC, we must treat quadratic functions having more than 3-bit numbers of input. Thirdly, after finding one S-box satisfying the SAC, we can make different S-box(es) satisfying the SAC.

Until now bent functions have been given great attention in coding theory, logic synthesis and spread spectrum communications. We proved that there exists an interesting relationship between bent functions and Boolean functions satisfying the (maximum order) SAC. All Boolean functions satisfying the maximum order SAC are always bent and all bent functions satisfy at least the 0-th order SAC. By computer search, we have also validated their relationship. However, due to their 0/1 imbalance and their existence for only the case of an even number of input bits, bent functions have some restrictions when used as a building block for constructing bijective cryptographic functions.

Also, we have proposed and proved a variety of systematic construction methods for S-boxes satisfying the SAC. For Boolean S-boxes satisfying the SAC, we can construct and enlarge them by using concatenation, Kronecker (or direct) product, and dyadic shift. For bijective S-boxes satisfying the SAC, when an n-bit input Boolean function and an n-bit input bijective function satisfying the SAC are given, the combined function is proved to become an $(n+1)$-bit bijective function satisfying the SAC. Moreover, we have proposed one simple method to construct bijective functions satisfying the maximum order SAC. After classifying S-boxes satisfying the SAC into balanced/unbalanced and unidirectional, bidirectional, and self-bidirectional functions, we gave how many functions exist for each case.

Since the design criteria of DES S-boxes have not been fully published in the open literature, there are many criticisms regarding DES S-boxes. As a practical application of Boolean functions satisfying the SAC, we proposed a simple method to design DES-like S-boxes based on our new design criteria and, in particular, included the possibility of differential cryptanalysis. We compared the well-known cryptographic measures of DES S-boxes and our newly designed DES-like S-boxes in respect of their nonlinearity, dependence matrix, cross correlation, and pairs XOR distribution. Our newly designed DES-like S-boxes were found to exhibit high nonlinearity, good independence between avalanche variables, and relatively higher pairs XOR distribution than DES S-boxes.

We called s^2DES a DES-like cryptosystem in which only S-boxes in DES were replaced by our designed DES-like S-boxes. If a cryptosystem had any statistical weakness within a smaller number of pairs of plaintext and ciphertext, the cryptosystem was said to be badly designed.

In order to evaluate the overall security of s^2DES, we compared and analyzed the

cryptographic properties of published DES-like cryptosystems including FEAL-4, FEAL-8, Multi2, and DES. We investigated the statistical properties (partial input/output dependence, randomness) and the new quantitative properties (sequence complexity, binary derivative test) of the above mentioned DES-like cryptosystems. All DES-like cryptosystems were found to have cryptographically satisfactory properties within our experiments. These results indicate that s^2DES can work well as a new DES-like cryptosystem too.

Moreover, after reducing the block size of three published Japanese DES-like cryptosystems from 64 bits to 16 bits, we checked their cycling properties (cycle length, variation of maximum cycles, number of disjoint cycles). The three DES-like cryptosystems were found to have almost the same cyclic structures.

We left the following topics of this thesis as open problems:

1. Counting bent functions and Boolean functions satisfying the SAC.

2. Relationship between the security of DES-like cryptosystems and the pairs XOR distribution.

3. Theoretical analysis of *Reducibility of Cryptosystem* and other ways of finding algebraic structure of DES-like cryptosystems.

We conclude that this thesis introduces several techniques valuable directly in the design and analysis of block cipher algorithms and indirectly to stream cipher algorithms.

Published Papers

1. Refereed Papers

1. Kwangjo Kim, Tsutomu Matsumoto, and Hideki Imai, "On Generating Cryptographically Desirable Substitutions", Trans. IEICE, Vol.E73, No.7, pp.1031–1035, Jul., 1990.

2. Kwangjo Kim, Tsutomu Matsumoto, and Hideki Imai, "A Recursive Construction Method of S-boxes Satisfying Strict Avalanche Criterion", Proc. of CRYPTO'90, Santa Barbara, CA., U.S.A., Aug.11–15, pp.545–553, 1990.

3. Kwangjo Kim, Tsutomu Matsumoto, Hideki Imai, and Hankyu Park, "A Study on the Construction Methods of Cryptographic Functions", *To appear in Trans. Korean Institute of Communication Sciences*, 1991.

2. Symposium, Workshop, *etc.*

1. Kwangjo Kim, Tsutomu Matsumoto, and Hideki Imai, "On the Statistical Security on Cipher Algorithms (*in Japanese*)", The 1989 Symposium on Cryptography and Information Security, Godenba, SCIS89-1-4, Feb.2–4, 1989.

2. Kwangjo Kim, "A Note on the Cyclic Properties of Reduced Block Cipher Algorithms", The Fifth KITE Convention in Japanese Branch, J89-03, Jun.3, 1989.

3. Kwangjo Kim, "A Study on the Cyclic Properties of Reduced Symmetric Cryptosystems", Proc. of the First Korean Workshop on Information Security and Cryptography, Daejeon, KOREA, Jul.24–26, pp.208–218, 1989.

4. Kwangjo Kim, Tsutomu Matsumoto, and Hideki Imai, "Generation of S-boxes Satisfying Strict Avalanche Criterion", The 1990 Symposium on Cryptography and Information Security, Nihondaira, SCIS90-14B, Jan.31–Feb.2, 1990.

5. Kwangjo Kim, Tsutomu Matsumoto, and Hideki Imai, "A Heuristic Algorithm for Generating Strong S-boxes", The 1990 Spring National Convention, A-287, Mar.18–21, 1990.

6. Kwangjo Kim, Tsutomu Matsumoto, and Hideki Imai, "An Efficient Generation Method of S-boxes Satisfying Strict Avalanche Criterion", The Seventh KITE Convention in Japanese Branch, (*Appeared in Proc. of the 1990 Korean Summer National Convention*), J90-06, Jun.2, 1990.

7. Kwangjo Kim, "A Recursive Construction Method of S-boxes Satisfying the Strict Avalanche Criterion", Proc. of the Second Korean Workshop on Information Security and Cryptography, Daejeon, KOREA, Sep.5–7, pp.165–174, 1990.

8. Kwangjo Kim, Tsutomu Matsumoto, and Hideki Imai, "Methods to Generate Functions Satisfying the Strict Avalanche Criterion", Technical Report on Information Security, ISEC90-30, Nov.13, 1990.

9. Kwangjo Kim, Tsutomu Matsumoto, and Hideki Imai, "On the Cryptographic Significance of Bent Functions", Joint Seminar KITE and KIEE, Tokyo, KSEAJ Letters, Dec.8, 1990.

10. Kwangjo Kim, Tsutomu Matsumoto, and Hideki Imai, "Design and Evaluation of DES-like S-boxes", The 1991 Symposium on Cryptography and Information Security, Jan.31–Feb.2, SCIS91-15B, Fujiyoshida, 1991.

Appendix A

Characteristics of DES S-boxes

A.1 DES S1-box

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

$(p_{i,j})$ of DES S1 box (Mean = 0.620)

Bit	1	2	3	4
c_1	0.563	0.563	0.500	0.750
c_2	0.563	0.563	0.563	0.688
c_3	0.625	0.625	0.688	0.500
c_4	0.688	0.625	0.563	0.500
c_5	0.688	0.750	0.625	0.750
c_6	0.750	0.625	0.625	0.500

$\rho_{i,j}(k)$ of DES S1 box (Mean = -0.195)

k	1	2	3	4	5	6
ρ_{12}	-0.143	-0.016	-0.467	0.035	-0.078	-0.149
ρ_{13}	-0.126	-0.270	-0.104	-0.459	-0.104	-0.149
ρ_{14}	-0.218	-0.323	-0.129	-0.135	-0.234	0.0
ρ_{23}	-0.252	-0.397	-0.244	-0.293	0.0	-0.200
ρ_{24}	-0.364	-0.323	-0.129	-0.258	-0.333	0.0
ρ_{34}	-0.144	-0.051	-0.270	-0.126	-0.447	-0.129

A.2 DES S2-box

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

$(p_{i,j})$ of DES S2 box (Mean = 0.633)

Bit	1	2	3	4
c_1	0.563	0.563	0.500	0.750
c_2	0.500	0.625	0.813	0.750
c_3	0.625	0.563	0.688	0.688
c_4	0.563	0.688	0.438	0.750
c_5	0.625	0.750	0.563	0.625
c_6	0.500	0.563	0.938	0.563

$\rho_{i,j}(k)$ of DES S2 box (Mean = -0.188)

k	1	2	3	4	6	6
ρ_{12}	-0.270	-0.129	-0.033	-0.187	-0.447	0.0
ρ_{13}	-0.378	-0.160	0.174	0.016	-0.163	-0.258
ρ_{14}	-0.073	-0.144	-0.522	-0.509	0.067	-0.252
ρ_{23}	0.000	-0.372	-0.187	-0.085	-0.364	-0.228
ρ_{24}	-0.364	-0.149	-0.187	-0.078	-0.149	-0.016
ρ_{34}	-0.289	-0.092	-0.455	-0.364	-0.163	0.033

A.3 DES S3-box

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

$(p_{i,j})$ of DES S3 box (Mean = 0.661)

Bit	1	2	3	4
c_1	0.750	0.563	0.688	0.688
c_2	0.625	0.750	0.563	0.813
c_3	0.563	0.563	0.563	0.813
c_4	0.625	0.750	0.563	0.625
c_5	0.875	0.500	0.750	0.563
c_6	0.563	0.500	0.875	0.750

$\rho_{i,j}(k)$ of DES S3 box (Mean = -0.165)

k	1	2	3	4	5	6
ρ_{12}	0.073	-0.149	-0.397	0.0	-0.189	-0.252
ρ_{13}	-0.389	0.098	-0.270	-0.293	0.0	-0.143
ρ_{14}	-0.234	-0.372	-0.101	-0.333	0.048	-0.218
ρ_{23}	0.357	-0.218	-0.016	-0.073	-0.289	-0.189
ρ_{24}	-0.323	-0.092	-0.262	-0.149	-0.126	-0.289
ρ_{34}	-0.309	-0.262	-0.101	-0.033	-0.218	-0.218

A.4 DES S4-box

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

$(p_{i,j})$ of DES S4 box (Mean = 0.615)

Bit	1	2	3	4
c_1	0.500	0.500	0.500	0.500
c_2	0.625	0.625	0.625	0.625
c_3	0.625	0.625	0.625	0.625
c_4	0.625	0.625	0.625	0.625
c_5	0.625	0.625	0.625	0.625
c_6	0.688	0.688	0.688	0.688

$\rho_{i,j}(k)$ of DES S4 box (Mean = -0.232)

k	1	2	3	4	5	6
ρ_{12}	-1.0	-0.067	-0.067	-0.333	-0.600	0.127
ρ_{13}	0.0	-0.333	-0.6	-0.067	-0.067	-0.455
ρ_{14}	0.0	-0.333	0.2	-0.333	0.200	-0.455
ρ_{23}	0.0	-0.333	0.2	-0.333	0.200	-0.455
ρ_{24}	0.0	-0.333	-0.6	-0.067	-0.067	-0.455
ρ_{34}	-1.0	-0.067	-0.067	-0.333	-0.600	0.127

A.5 DES S5-box

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

$(p_{i,j})$ of DES S5 box (Mean = 0.633)

Bit	1	2	3	4
c_1	0.625	0.438	0.813	0.563
c_2	0.625	0.563	0.750	0.625
c_3	0.625	0.625	0.625	0.563
c_4	0.500	0.625	0.688	0.625
c_5	0.563	0.750	0.688	0.688
c_6	0.688	0.625	0.688	0.625

$\rho_{i,j}(k)$ of DES S5 box (Mean = -0.184)

k	1	2	3	4	5	6
ρ_{12}	-0.098	-0.423	-0.067	-0.258	-0.218	-0.104
ρ_{13}	-0.207	-0.149	-0.200	0.000	-0.051	-0.164
ρ_{14}	-0.163	-0.200	-0.033	-0.129	-0.323	-0.383
ρ_{23}	-0.222	-0.073	-0.200	-0.244	-0.234	-0.244
ρ_{24}	-0.365	-0.033	-0.423	-0.067	-0.078	-0.067
ρ_{34}	-0.101	-0.149	-0.163	-0.383	-0.455	0.035

A.6 DES S6-box

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

$(p_{i,j})$ of DES S6 box (Mean = 0.651)

Bit	1	2	3	4
c_1	0.563	0.563	0.750	0.750
c_2	0.625	0.625	0.750	0.563
c_3	0.563	0.813	0.500	0.750
c_4	0.500	0.688	0.750	0.563
c_5	0.750	0.625	0.688	0.563
c_6	0.563	0.688	0.688	0.750

$\rho_{i,j}(k)$ of DES S6 box (Mean = -0.183)

k	1	2	3	4	5	6
ρ_{12}	-0.397	-0.200	-0.262	-0.405	0.149	-0.323
ρ_{13}	-0.364	-0.149	-0.378	-0.144	-0.234	-0.051
ρ_{14}	-0.073	-0.293	0.073	0.126	-0.509	-0.073
ρ_{23}	-0.218	-0.298	0.000	-0.234	-0.244	-0.164
ρ_{24}	-0.364	-0.033	-0.277	-0.459	-0.163	0.078
ρ_{34}	0.167	-0.364	0.000	-0.073	-0.051	-0.389

A.7 DES S7-box

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

$(p_{i,j})$ of DES S7 box (Mean = 0.656)

Bit	1	2	3	4
c_1	0.750	0.625	0.625	0.625
c_2	0.750	0.438	0.875	0.750
c_3	0.750	0.688	0.438	0.625
c_4	0.625	0.750	0.563	0.625
c_5	0.563	0.688	0.625	0.688
c_6	0.875	0.688	0.563	0.563

$\rho_{i,j}(k)$ of DES S7 box (Mean = -0.153)

k	1	2	3	4	5	6
ρ_{12}	-0.298	-0.073	-0.234	0.000	-0.051	-0.051
ρ_{13}	-0.149	-0.218	-0.073	0.098	-0.423	-0.143
ρ_{14}	-0.149	-0.167	-0.149	-0.200	-0.051	0.048
ρ_{23}	0.200	-0.048	-0.221	-0.218	-0.104	-0.323
ρ_{24}	-0.333	0.218	-0.244	-0.447	-0.309	-0.051
ρ_{34}	-0.467	-0.218	-0.358	-0.163	-0.244	0.111

A.8 DES S8-box

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

$(p_{i,j})$ of DES S8 box (Mean = 0.625)

Bit	1	2	3	4
c_1	0.500	0.438	0.813	0.500
c_2	0.688	0.438	0.625	0.625
c_3	0.500	0.625	0.688	0.625
c_4	0.688	0.625	0.563	0.750
c_5	0.625	0.813	0.688	0.625
c_6	0.813	0.688	0.375	0.688

$\rho_{i,j}(k)$ of DES S8 box (Mean = -0.176)

k	1	2	3	4	5	6
ρ_{12}	-0.252	-0.221	-0.129	0.035	-0.207	-0.151
ρ_{13}	0.000	-0.383	-0.135	-0.187	0.035	-0.124
ρ_{14}	-0.375	-0.244	-0.387	-0.078	-0.067	-0.324
ρ_{23}	-0.545	-0.228	-0.104	-0.163	-0.324	0.104
ρ_{24}	-0.126	-0.228	-0.333	0.000	-0.372	-0.164
ρ_{34}	0.000	-0.067	-0.383	-0.218	-0.244	0.244

Appendix B

List of All 4-bit Input Bent Functions

No.	0–15	No.	0–15	No.	0–15	No.	0–15	No.	0–15
1	(6ac0)	61	(d148)	121	(7844)	181	(a90c)	241	(7822)
2	(9ac0)	62	(1d48)	122	(b444)	182	(590c)	242	(b422)
3	(a6c0)	63	(8b48)	123	(d244)	183	(650c)	243	(d222)
4	(56c0)	64	(4748)	124	(1e44)	184	(950c)	244	(1e22)
5	(a9c0)	65	(d828)	125	(e144)	185	(60ac)	245	(e122)
6	(59c0)	66	(e428)	126	(2d44)	186	(90ac)	246	(2d22)
7	(65c0)	67	(7228)	127	(4b44)	187	(06ac)	247	(4b22)
8	(95c0)	68	(4e28)	128	(8744)	188	(f6ac)	248	(8722)
9	(6ca0)	69	(b128)	129	(e824)	189	(09ac)	249	(48e2)
10	(9ca0)	70	(8d28)	130	(d424)	190	(f9ac)	250	(84e2)
11	(c6a0)	71	(1b28)	131	(b224)*	191	(6fac)	251	(12e2)
12	(36a0)	72	(2728)	132	(8e24)	192	(9fac)	252	(dee2)
13	(c9a0)	73	(18e8)	133	(7124)	193	(a06c)	253	(21e2)
14	(39a0)	74	(24e8)	134	(4d24)*	194	(506c)	254	(ede2)
15	(63a0)	75	(42e8)	135	(2b24)	195	(0a6c)	255	(7be2)
16	(93a0)	76	(7ee8)*	136	(1724)	196	(fa6c)	256	(b7e2)
17	(ac60)	77	(81e8)*	137	(28e4)	197	(056c)	257	(b812)
18	(5c60)	78	(bde8)	138	(14e4)	198	(f56c)	258	(7412)
19	(ca60)	79	(dbe8)	139	(82e4)	199	(af6c)	259	(e212)
20	(3a60)	80	(e7e8)	140	(bee4)	200	(5f6c)	260	(2e12)
21	(c560)	81	(e818)	141	(41e4)	201	(a09c)	261	(d112)
22	(3560)	82	(d418)	142	(7de4)	202	(509c)	262	(1d12)
23	(a360)	83	(b218)	143	(ebe4)	203	(0a9c)	263	(8b12)
24	(5360)	84	(8e18)*	144	(d7e4)	204	(fa9c)	264	(4712)
25	(ac90)	85	(7118)*	145	(d814)	205	(059c)	265	(88d2)
26	(5c90)	86	(4d18)	146	(e414)	206	(f59c)	266	(44d2)
27	(ca90)	87	(2b18)	147	(7214)	207	(af9c)	267	(22d2)
28	(3a90)	88	(1718)	148	(4e14)	208	(5f9c)	268	(eed2)
29	(c590)	89	(28d8)	149	(b114)	209	(605c)	269	(11d2)
30	(3590)	90	(14d8)	150	(8d14)	210	(905c)	270	(ddd2)
31	(a390)	91	(82d8)	151	(1b14)	211	(065c)	271	(bbd2)
32	(5390)	92	(bed8)	152	(2714)	212	(f65c)	272	(77d2)
33	(6c50)	93	(41d8)	153	(18d4)	213	(095c)	273	(18b2)
34	(9c50)	94	(7dd8)	154	(24d4)	214	(f95c)	274	(24b2)*
35	(c650)	95	(ebd8)	155	(42d4)*	215	(6f5c)	275	(42b2)
36	(3650)	96	(d7d8)	156	(7ed4)	216	(9f5c)	276	(7eb2)
37	(c950)	97	(48b8)	157	(81d4)	217	(6afc)	277	(81b2)
38	(3950)	98	(84b8)	158	(bdd4)*	218	(9afc)	278	(bdb2)
39	(6350)	99	(12b8)	159	(dbd4)	219	(a6fc)	279	(dbb2)*
40	(9350)	100	(deb8)	160	(e7d4)	220	(56fc)	280	(e7b2)
41	(6a30)	101	(21b8)	161	(88b4)	221	(a9fc)	281	(2872)
42	(9a30)	102	(edb8)	162	(44b4)	222	(59fc)	282	(1472)
43	(a630)	103	(7bb8)	163	(22b4)	223	(65fc)	283	(8272)
44	(5630)	104	(b7b8)	164	(eeb4)	224	(95fc)	284	(be72)
45	(a930)	105	(8878)	165	(11b4)	225	(d882)	285	(4172)
46	(5930)	106	(4478)	166	(ddb4)	226	(e482)	286	(7d72)
47	(6530)	107	(2278)	167	(bbb4)	227	(7282)	287	(eb72)
48	(9530)	108	(ee78)	168	(77b4)	228	(4e82)	288	(d772)
49	(7888)	109	(1178)	169	(4874)	229	(b182)	289	(6c0a)
50	(b488)	110	(dd78)	170	(8474)	230	(8d82)	290	(9c0a)
51	(d288)	111	(bb78)	171	(1274)	231	(1b82)	291	(c60a)
52	(1e88)	112	(7778)	172	(de74)	232	(2782)	292	(360a)
53	(e188)	113	(b884)	173	(2174)	233	(e842)	293	(c90a)
54	(2d88)	114	(7484)	174	(ed74)	234	(d442)*	294	(390a)
55	(4b88)	115	(e284)	175	(7b74)	235	(b242)	295	(630a)
56	(8788)	116	(2e84)	176	(b774)	236	(8e42)	296	(930a)
57	(b848)	117	(d184)	177	(6a0c)	237	(7142)	297	(60ca)
58	(7448)	118	(1d84)	178	(9a0c)	238	(4d42)	298	(90ca)
59	(e248)	119	(8b84)	179	(a60c)	239	(2b42)*	299	(06ca)
60	(2e48)	120	(4784)	180	(560c)	240	(1742)	300	(f6ca)

* : satisfying the 2-nd order SAC

No.	0–15	No.	0–15	No.	0–15	No.	0–15	No.	0–15
301	(09ca)	361	(c056)	421	(111e)	481	(7811)	541	(0359)
302	(f9ca)	362	(3056)	422	(dd1e)	482	(b411)	542	(f359)
303	(6fca)	363	(0c56)	423	(bb1e)	483	(d211)	543	(cf59)
304	(9fca)	364	(fc56)	424	(771e)	484	(1e11)	544	(3f59)
305	(c06a)	365	(0356)	425	(b8de)	485	(e111)	545	(a039)
306	(306a)	366	(f356)	426	(74de)	486	(2d11)	546	(5039)
307	(0c6a)	367	(cf56)	427	(e2de)	487	(4b11)	547	(0a39)
308	(fc6a)	368	(3f56)	428	(2ede)	488	(8711)	548	(fa39)
309	(036a)	369	(a036)	429	(d1de)	489	(48d1)	549	(0539)
310	(f36a)	370	(5036)	430	(1dde)	490	(84d1)	550	(f539)
311	(cf6a)	371	(0a36)	431	(8bde)	491	(12d1)	551	(af39)
312	(3f6a)	372	(fa36)	432	(47de)	492	(ded1)	552	(5f39)
313	(c09a)	373	(0536)	433	(d8be)	493	(21d1)	553	(acf9)
314	(309a)	374	(f536)	434	(e4be)	494	(edd1)	554	(5cf9)
315	(0c9a)	375	(af36)	435	(72be)	495	(7bd1)	555	(caf9)
316	(fc9a)	376	(5f36)	436	(4ebe)	496	(b7d1)	556	(3af9)
317	(039a)	377	(acf6)	437	(b1be)	497	(28b1)	557	(c5f9)
318	(f39a)	378	(5cf6)	438	(8dbe)	498	(14b1)	558	(35f9)
319	(cf9a)	379	(caf6)	439	(1bbe)	499	(82b1)	559	(a3f9)
320	(3f9a)	380	(3af6)	440	(27be)	500	(beb1)	560	(53f9)
321	(603a)	381	(c5f6)	441	(e87e)*	501	(41b1)	561	(6c05)
322	(903a)	382	(35f6)	442	(d47e)	502	(7db1)	562	(9c05)
323	(063a)	383	(a3f6)	443	(b27e)	503	(ebb1)	563	(c605)
324	(f63a)	384	(53f6)	444	(8e7e)	504	(d7b1)	564	(3605)
325	(093a)	385	(188e)*	445	(717e)	505	(1871)*	565	(c905)
326	(f93a)	386	(248e)	446	(4d7e)	506	(2471)	566	(3905)
327	(6f3a)	387	(428e)	447	(2b7e)	507	(4271)	567	(6305)
328	(9f3a)	388	(7e8e)	448	(177e)*	508	(7e71)	568	(9305)
329	(6cfa)	389	(818e)	449	(e881)*	509	(8171)	569	(60c5)
330	(9cfa)	390	(bd8e)	450	(d481)	510	(bd71)	570	(90c5)
331	(c6fa)	391	(db8e)	451	(b281)	511	(db71)	571	(06c5)
332	(36fa)	392	(e78e)*	452	(8e81)	512	(e771)*	572	(f6c5)
333	(c9fa)	393	(284e)	453	(7181)	513	(ac09)	573	(09c5)
334	(39fa)	394	(144e)	454	(4d81)	514	(5c09)	574	(f9c5)
335	(63fa)	395	(824e)	455	(2b81)	515	(ca09)	575	(6fc5)
336	(93fa)	396	(be4e)	456	(1781)*	516	(3a09)	576	(9fc5)
337	(ac06)	397	(414e)	457	(d841)	517	(c509)	577	(c065)
338	(5c06)	398	(7d4e)	458	(e441)	518	(3509)	578	(3065)
339	(ca06)	399	(eb4e)	459	(7241)	519	(a309)	579	(0c65)
340	(3a06)	400	(d74e)	460	(4e41)	520	(5309)	580	(fc65)
341	(c506)	401	(482e)	461	(b141)	521	(a0c9)	581	(0365)
342	(3506)	402	(842e)	462	(8d41)	522	(50c9)	582	(f365)
343	(a306)	403	(122e)	463	(1b41)	523	(0ac9)	583	(cf65)
344	(5306)	404	(de2e)	464	(2741)	524	(fac9)	584	(3f65)
345	(a0c6)	405	(212e)	465	(b821)	525	(05c9)	585	(c095)
346	(50c6)	406	(ed2e)	466	(7421)	526	(f5c9)	586	(3095)
347	(0ac6)	407	(7b2e)	467	(e221)	527	(afc9)	587	(0c95)
348	(fac6)	408	(b72e)	468	(2e21)	528	(5fc9)	588	(fc95)
349	(05c6)	409	(78ee)	469	(d121)	529	(c0a9)	589	(0395)
350	(f5c6)	410	(b4ee)	470	(1d21)	530	(30a9)	590	(f395)
351	(afc6)	411	(d2ee)	471	(8b21)	531	(0ca9)	591	(cf95)
352	(5fc6)	412	(1eee)	472	(4721)	532	(fca9)	592	(3f95)
353	(c0a6)	413	(e1ee)	473	(88e1)	533	(03a9)	593	(6035)
354	(30a6)	414	(2dee)	474	(44e1)	534	(f3a9)	594	(9035)
355	(0ca6)	415	(4bee)	475	(22e1)	535	(cfa9)	595	(0635)
356	(fca6)	416	(87ee)	476	(eee1)	536	(3fa9)	596	(f635)
357	(03a6)	417	(881e)	477	(11e1)	537	(c059)	597	(0935)
358	(f3a6)	418	(441e)	478	(dde1)	538	(3059)	598	(f935)
359	(cfa6)	419	(221e)	479	(bbe1)	539	(0c59)	599	(6f35)
360	(3fa6)	420	(ee1e)	480	(77e1)	540	(fc59)	600	(9f35)

* : satisfying the 2-nd order SAC

No.	0–15	No.	0–15	No.	0–15	No.	0–15	No.	0–15
601	(6cf5)	661	(71bd)	721	(488b)	781	(d17b)	841	(7877)
602	(9cf5)	662	(4dbd)	722	(848b)	782	(1d7b)	842	(b477)
603	(c6f5)	663	(2bbd)*	723	(128b)	783	(8b7b)	843	(d277)
604	(36f5)	664	(17bd)	724	(de8b)	784	(477b)	844	(1e77)
605	(c9f5)	665	(d87d)	725	(218b)	785	(8887)	845	(e177)
606	(39f5)	666	(e47d)	726	(ed8b)	786	(4487)	846	(2d77)
607	(63f5)	667	(727d)	727	(7b8b)	787	(2287)	847	(4b77)
608	(93f5)	668	(4e7d)	728	(b78b)	788	(ee87)	848	(8777)
609	(288d)	669	(b17d)	729	(884b)	789	(1187)	849	(6acf)
610	(148d)	670	(8d7d)	730	(444b)	790	(dd87)	850	(9acf)
611	(828d)	671	(1b7d)	731	(224b)	791	(bb87)	851	(a6cf)
612	(be8d)	672	(277d)	732	(ee4b)	792	(7787)	852	(56cf)
613	(418d)	673	(6a03)	733	(114b)	793	(4847)	853	(a9cf)
614	(7d8d)	674	(9a03)	734	(dd4b)	794	(8447)	854	(59cf)
615	(eb8d)	675	(a603)	735	(bb4b)	795	(1247)	855	(65cf)
616	(d78d)	676	(5603)	736	(774b)	796	(de47)	856	(95cf)
617	(184d)	677	(a903)	737	(182b)	797	(2147)	857	(6caf)
618	(244d)*	678	(5903)	738	(242b)	798	(ed47)	858	(9caf)
619	(424d)	679	(6503)	739	(422b)*	799	(7b47)	859	(c6af)
620	(7e4d)	680	(9503)	740	(7e2b)	800	(b747)	860	(36af)
621	(814d)	681	(60a3)	741	(812b)	801	(2827)	861	(c9af)
622	(bd4d)	682	(90a3)	742	(bd2b)*	802	(1427)	862	(39af)
623	(db4d)*	683	(06a3)	743	(db2b)	803	(8227)	863	(63af)
624	(e74d)	684	(f6a3)	744	(e72b)	804	(be27)	864	(93af)
625	(882d)	685	(09a3)	745	(d8eb)	805	(4127)	865	(ac6f)
626	(442d)	686	(f9a3)	746	(e4eb)	806	(7d27)	866	(5c6f)
627	(222d)	687	(6fa3)	747	(72eb)	807	(eb27)	867	(ca6f)
628	(ee2d)	688	(9fa3)	748	(4eeb)	808	(d727)	868	(3a6f)
629	(112d)	689	(a063)	749	(b1eb)	809	(e8e7)	869	(c56f)
630	(dd2d)	690	(5063)	750	(8deb)	810	(d4e7)	870	(356f)
631	(bb2d)	691	(0a63)	751	(1beb)	811	(b2e7)	871	(a36f)
632	(772d)	692	(fa63)	752	(27eb)	812	(8ee7)*	872	(536f)
633	(b8ed)	693	(0563)	753	(281b)	813	(71e7)*	873	(ac9f)
634	(74ed)	694	(f563)	754	(141b)	814	(4de7)	874	(5c9f)
635	(e2ed)	695	(af63)	755	(821b)	815	(2be7)	875	(ca9f)
636	(2eed)	696	(5f63)	756	(be1b)	816	(17e7)	876	(3a9f)
637	(d1ed)	697	(a093)	757	(411b)	817	(1817)	877	(c59f)
638	(1ded)	698	(5093)	758	(7d1b)	818	(2417)	878	(359f)
639	(8bed)	699	(0a93)	759	(eb1b)	819	(4217)	879	(a39f)
640	(47ed)	700	(fa93)	760	(d71b)	820	(7e17)*	880	(539f)
641	(481d)	701	(0593)	761	(e8db)	821	(8117)*	881	(6c5f)
642	(841d)	702	(f593)	762	(d4db)	822	(bd17)	882	(9c5f)
643	(121d)	703	(af93)	763	(b2db)*	823	(db17)	883	(c65f)
644	(de1d)	704	(5f93)	764	(8edb)	824	(e717)	884	(365f)
645	(211d)	705	(6053)	765	(71db)	825	(d8d7)	885	(c95f)
646	(ed1d)	706	(9053)	766	(4ddb)*	826	(e4d7)	886	(395f)
647	(7b1d)	707	(0653)	767	(2bdb)	827	(72d7)	887	(635f)
648	(b71d)	708	(f653)	768	(17db)	828	(4ed7)	888	(935f)
649	(78dd)	709	(0953)	769	(78bb)	829	(b1d7)	889	(6a3f)
650	(b4dd)	710	(f953)	770	(b4bb)	830	(8dd7)	890	(9a3f)
651	(d2dd)	711	(6f53)	771	(d2bb)	831	(1bd7)	891	(a63f)
652	(1edd)	712	(9f53)	772	(1ebb)	832	(27d7)	892	(563f)
653	(e1dd)	713	(6af3)	773	(e1bb)	833	(b8b7)	893	(a93f)
654	(2ddd)	714	(9af3)	774	(2dbb)	834	(74b7)	894	(593f)
655	(4bdd)	715	(a6f3)	775	(4bbb)	835	(e2b7)	895	(653f)
656	(87dd)	716	(56f3)	776	(87bb)	836	(2eb7)	896	(953f)
657	(e8bd)	717	(a9f3)	777	(b87b)	837	(d1b7)		
658	(d4bd)*	718	(59f3)	778	(747b)	838	(1db7)		
659	(b2bd)	719	(65f3)	779	(e27b)	839	(8bb7)		
660	(8ebd)	720	(95f3)	780	(2e7b)	840	(47b7)		

* : satisfying the 2-nd order SAC

Appendix C

List of All Bidirectional 3-bit Bijective Functions Satisfying the 1-st order SAC

No.	0 1 2 3 4 5 6 7		No.	0 1 2 3 4 5 6 7	
1	0 1 2 4 3 5 6 7	S	33	1 3 0 5 2 7 4 6	
2	0 1 3 5 2 4 6 7		34	1 3 2 7 0 5 4 6	
3	0 1 4 2 5 3 6 7		35	1 3 5 0 7 2 4 6	
4	0 1 5 3 4 2 6 7	S	36	1 3 7 2 5 0 4 6	
5	0 2 1 4 3 6 5 7	S	37	1 4 0 2 5 7 3 6	
6	0 2 3 6 1 4 5 7		38	1 4 2 0 7 5 3 6	
7	0 2 4 1 6 3 5 7		39	1 4 5 7 0 2 3 6	
8	0 2 6 3 4 1 5 7		40	1 4 7 5 2 0 3 6	
9	0 3 1 5 2 6 4 7		41	1 5 0 3 4 7 2 6	
10	0 3 2 6 1 5 4 7		42	1 5 3 0 7 4 2 6	
11	0 3 5 1 6 2 4 7	S	43	1 5 4 7 0 3 2 6	
12	0 3 6 2 5 1 4 7		44	1 5 7 4 3 0 2 6	
13	0 4 1 2 5 6 3 7		45	1 7 2 3 4 5 0 6	
14	0 4 2 1 6 5 3 7		46	1 7 3 2 5 4 0 6	
15	0 4 5 6 1 2 3 7	S	47	1 7 4 5 2 3 0 6	
16	0 4 6 5 2 1 3 7		48	1 7 5 4 3 2 0 6	
17	0 5 1 3 4 6 2 7		49	2 0 1 4 3 6 7 5	
18	0 5 3 1 6 4 2 7		50	2 0 3 6 1 4 7 5	
19	0 5 4 6 1 3 2 7		51	2 0 4 1 6 3 7 5	
20	0 5 6 4 3 1 2 7	S	52	2 0 6 3 4 1 7 5	
21	0 6 2 3 4 5 1 7	S	53	2 1 0 4 3 7 6 5	S
22	0 6 3 2 5 4 1 7	S	54	2 1 3 7 0 4 6 5	
23	0 6 4 5 2 3 1 7	S	55	2 1 4 0 7 3 6 5	
24	0 6 5 4 3 2 1 7	S	56	2 1 7 3 4 0 6 5	
25	1 0 2 4 3 5 7 6	S	57	2 3 0 6 1 7 4 5	
26	1 0 3 5 2 4 7 6		58	2 3 1 7 0 6 4 5	
27	1 0 4 2 5 3 7 6		59	2 3 6 0 7 1 4 5	
28	1 0 5 3 4 2 7 6	S	60	2 3 7 1 6 0 4 5	
29	1 2 0 4 3 7 5 6		61	2 4 0 1 6 7 3 5	
30	1 2 3 7 0 4 5 6		62	2 4 1 0 7 6 3 5	
31	1 2 4 0 7 3 5 6		63	2 4 6 7 0 1 3 5	
32	1 2 7 3 4 0 5 6		64	2 4 7 6 1 0 3 5	

S : Self-bidirectional SAC

No.	0 1 2 3 4 5 6 7		No.	0 1 2 3 4 5 6 7	
65	2 6 0 3 4 7 1 5	S	97	4 0 1 2 5 6 7 3	
66	2 6 3 0 7 4 1 5		98	4 0 2 1 6 5 7 3	
67	2 6 4 7 0 3 1 5		99	4 0 5 6 1 2 7 3	
68	2 6 7 4 3 0 1 5		100	4 0 6 5 2 1 7 3	
69	2 7 1 3 4 6 0 5		101	4 1 0 2 5 7 6 3	
70	2 7 3 1 6 4 0 5		102	4 1 2 0 7 5 6 3	
71	2 7 4 6 1 3 0 5		103	4 1 5 7 0 2 6 3	S
72	2 7 6 4 3 1 0 5		104	4 1 7 5 2 0 6 3	
73	3 0 1 5 2 6 7 4		105	4 2 0 1 6 7 5 3	
74	3 0 2 6 1 5 7 4		106	4 2 1 0 7 6 5 3	
75	3 0 5 1 6 2 7 4		107	4 2 6 7 0 1 5 3	
76	3 0 6 2 5 1 7 4		108	4 2 7 6 1 0 5 3	
77	3 1 0 5 2 7 6 4		109	4 5 0 6 1 7 2 3	
78	3 1 2 7 0 5 6 4		110	4 5 1 7 0 6 2 3	
79	3 1 5 0 7 2 6 4	S	111	4 5 6 0 7 1 2 3	
80	3 1 7 2 5 0 6 4		112	4 5 7 1 6 0 2 3	
81	3 2 0 6 1 7 5 4		113	4 6 0 5 2 7 1 3	
82	3 2 1 7 0 6 5 4		114	4 6 2 7 0 5 1 3	S
83	3 2 6 0 7 1 5 4		115	4 6 5 0 7 2 1 3	
84	3 2 7 1 6 0 5 4		116	4 6 7 2 5 0 1 3	
85	3 5 0 1 6 7 2 4		117	4 7 1 5 2 6 0 3	
86	3 5 1 0 7 6 2 4		118	4 7 2 6 1 5 0 3	
87	3 5 6 7 0 1 2 4		119	4 7 5 1 6 2 0 3	
88	3 5 7 6 1 0 2 4		120	4 7 6 2 5 1 0 3	
89	3 6 0 2 5 7 1 4		121	5 0 1 3 4 6 7 2	
90	3 6 2 0 7 5 1 4	S	122	5 0 3 1 6 4 7 2	
91	3 6 5 7 0 2 1 4		123	5 0 4 6 1 3 7 2	
92	3 6 7 5 2 0 1 4		124	5 0 6 4 3 1 7 2	
93	3 7 1 2 5 6 0 4		125	5 1 0 3 4 7 6 2	
94	3 7 2 1 6 5 0 4		126	5 1 3 0 7 4 6 2	
95	3 7 5 6 1 2 0 4		127	5 1 4 7 0 3 6 2	
96	3 7 6 5 2 1 0 4		128	5 1 7 4 3 0 6 2	S

S : Self-bidirectional SAC

No.	0 1 2 3 4 5 6 7		No.	0 1 2 3 4 5 6 7	
129	5 3 0 1 6 7 4 2		161	6 5 0 4 3 7 2 1	
130	5 3 1 0 7 6 4 2		162	6 5 3 7 0 4 2 1	
131	5 3 6 7 0 1 4 2		163	6 5 4 0 7 3 2 1	
132	5 3 7 6 1 0 4 2		164	6 5 7 3 4 0 2 1	
133	5 4 0 6 1 7 3 2		165	6 7 2 4 3 5 0 1	S
134	5 4 1 7 0 6 3 2		166	6 7 3 5 2 4 0 1	
135	5 4 6 0 7 1 3 2		167	6 7 4 2 5 3 0 1	
136	5 4 7 1 6 0 3 2		168	6 7 5 3 4 2 0 1	S
137	5 6 0 4 3 7 1 2		169	7 1 2 3 4 5 6 0	S
138	5 6 3 7 0 4 1 2		170	7 1 3 2 5 4 6 0	S
139	5 6 4 0 7 3 1 2		171	7 1 4 5 2 3 6 0	S
140	5 6 7 3 4 0 1 2	S	172	7 1 5 4 3 2 6 0	S
141	5 7 1 4 3 6 0 2		173	7 2 1 3 4 6 5 0	S
142	5 7 3 6 1 4 0 2		174	7 2 3 1 6 4 5 0	
143	5 7 4 1 6 3 0 2		175	7 2 4 6 1 3 5 0	
144	5 7 6 3 4 1 0 2		176	7 2 6 4 3 1 5 0	
145	6 0 2 3 4 5 7 1		177	7 3 1 2 5 6 4 0	
146	6 0 3 2 5 4 7 1		178	7 3 2 1 6 5 4 0	S
147	6 0 4 5 2 3 7 1		179	7 3 5 6 1 2 4 0	
148	6 0 5 4 3 2 7 1		180	7 3 6 5 2 1 4 0	
149	6 2 0 3 4 7 5 1		181	7 4 1 5 2 6 3 0	
150	6 2 3 0 7 4 5 1		182	7 4 2 6 1 5 3 0	S
151	6 2 4 7 0 3 5 1		183	7 4 5 1 6 2 3 0	
152	6 2 7 4 3 0 5 1		184	7 4 6 2 5 1 3 0	
153	6 3 0 2 5 7 4 1		185	7 5 1 4 3 6 2 0	
154	6 3 2 0 7 5 4 1		186	7 5 3 6 1 4 2 0	
155	6 3 5 7 0 2 4 1		187	7 5 4 1 6 3 2 0	
156	6 3 7 5 2 0 4 1		188	7 5 6 3 4 1 2 0	S
157	6 4 0 5 2 7 3 1		189	7 6 2 4 3 5 1 0	S
158	6 4 2 7 0 5 3 1		190	7 6 3 5 2 4 1 0	
159	6 4 5 0 7 2 3 1		191	7 6 4 2 5 3 1 0	
160	6 4 7 2 5 0 3 1		192	7 6 5 3 4 2 1 0	S

S : Self-bidirectional SAC

Appendix D

Characteristics of s^2DES S-boxes

D.1 s^2DES S1-box

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

$(p_{i,j})$ of s^2DES S1 box (Mean = 0.495)

Bit	1	2	3	4
c_1	0.500	0.563	0.500	0.438
c_2	0.500	0.500	0.500	0.500
c_3	0.500	0.500	0.500	0.500
c_4	0.500	0.500	0.500	0.500
c_5	0.500	0.500	0.500	0.500
c_6	0.563	0.375	0.563	0.375

$\rho_{i,j}(k)$ of s^2DES S1 box (Mean = -0.051)

k	1	2	3	4	5	6
ρ_{12}	0.000	0.000	-0.125	0.250	0.125	-0.098
ρ_{13}	-0.125	0.125	-0.250	-0.125	0.250	-0.270
ρ_{14}	-0.126	0.000	0.125	0.125	-0.250	0.033
ρ_{23}	-0.252	0.000	0.125	-0.250	-0.125	0.163
ρ_{24}	-0.238	-0.125	-0.375	0.375	0.250	0.067
ρ_{34}	0.252	-0.125	0.125	-0.750	-0.250	-0.358

D.2 s^2DES S2-box

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

$(p_{i,j})$ of s^2DES S2 box (Mean = 0.510)

Bit	1	2	3	4
c_1	0.625	0.438	0.625	0.563
c_2	0.500	0.500	0.500	0.500
c_3	0.500	0.500	0.500	0.500
c_4	0.500	0.500	0.500	0.500
c_5	0.500	0.500	0.500	0.500
c_6	0.438	0.625	0.500	0.438

$\rho_{i,j}(k)$ of s^2DES S2 box (Mean = -0.07)

k	1	2	3	4	5	6
ρ_{12}	-0.358	-0.125	0.250	-0.250	-0.375	-0.098
ρ_{13}	0.067	-0.250	-0.250	-0.125	0.125	0.000
ρ_{14}	-0.033	0.125	-0.125	-0.250	-0.125	-0.016
ρ_{23}	-0.098	0.000	-0.250	-0.125	0.125	-0.258
ρ_{24}	0.143	-0.125	0.125	0.000	-0.125	-0.098
ρ_{34}	0.098	-0.125	-0.250	0.125	0.125	0.000

D.3 s^2DES S3-box

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

$(p_{i,j})$ of s^2DES S3 box (Mean =0.505)

Bit	1	2	3	4
c_1	0.563	0.500	0.438	0.438
c_2	0.500	0.500	0.500	0.500
c_3	0.500	0.500	0.500	0.500
c_4	0.500	0.500	0.500	0.500
c_5	0.500	0.500	0.500	0.500
c_6	0.500	0.563	0.500	0.625

$\rho_{i,j}(k)$ of s^2DES S3box (Mean = -0.053)

k	1	2	3	4	5	6
ρ_{12}	-0.126	0.000	-0.125	0.000	0.000	0.126
ρ_{13}	0.270	0.000	-0.125	0.250	-0.375	0.250
ρ_{14}	-0.238	0.125	0.250	-0.125	-0.375	-0.258
ρ_{23}	0.000	-0.125	-0.375	0.250	0.000	0.000
ρ_{24}	-0.126	0.125	-0.375	-0.250	-0.125	-0.163
ρ_{34}	-0.143	0.000	0.250	-0.125	-0.125	-0.129

D.4 s^2DES S4-box

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

$(p_{i,j})$ of s^2DES S4 box (Mean = 0.521)

Bit	1	2	3	4
c_1	0.563	0.500	0.563	0.625
c_2	0.500	0.500	0.500	0.500
c_3	0.500	0.500	0.500	0.500
c_4	0.500	0.500	0.500	0.500
c_5	0.500	0.500	0.500	0.500
c_6	0.500	0.750	0.500	0.500

$\rho_{i,j}(k)$ of s^2DES S4 box (Mean =-0.083)

k	1	2	3	4	5	6
ρ_{12}	-0.126	0.250	0.000	0.000	-0.250	0.144
ρ_{13}	0.111	-0.250	-0.250	-0.500	-0.500	-0.125
ρ_{14}	-0.033	-0.125	-0.250	0.125	-0.250	0.250
ρ_{23}	0.252	0.000	0.250	-0.125	0.000	-0.289
ρ_{24}	-0.387	-0.375	-0.250	-0.125	0.375	0.144
ρ_{34}	-0.293	0.000	-0.375	0.000	0.000	0.000

D.5 s^2DES S5-box

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

$(p_{i,j})$ of s^2DES S5 box (Mean =0.516)

Bit	1	2	3	4
c_1	0.688	0.500	0.438	0.500
c_2	0.500	0.500	0.500	0.500
c_3	0.500	0.500	0.500	0.500
c_4	0.500	0.500	0.500	0.500
c_5	0.500	0.500	0.500	0.500
c_6	0.563	0.500	0.563	0.625

$\rho_{i,j}(k)$ of s^2DES S5 box (Mean =-0.074)

k	1	2	3	4	5	6
ρ_{12}	0.405	0.250	-0.250	0.125	-0.125	-0.126
ρ_{13}	-0.221	-0.125	-0.375	-0.125	-0.125	-0.143
ρ_{14}	0.000	-0.375	0.250	0.250	0.125	-0.033
ρ_{23}	0.126	0.000	0.000	-0.125	0.125	-0.252
ρ_{24}	-0.250	-0.125	-0.250	0.125	-0.250	0.000
ρ_{34}	-0.252	-0.125	-0.125	-0.250	-0.375	-0.033

D.6 s^2DES S6-box

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

$(p_{i,j})$ of s^2DES S6 box (Mean =0.516)

Bit	1	2	3	4
c_1	0.625	0.500	0.625	0.438
c_2	0.500	0.500	0.500	0.500
c_3	0.500	0.500	0.500	0.500
c_4	0.500	0.500	0.500	0.500
c_5	0.500	0.500	0.500	0.500
c_6	0.500	0.563	0.625	0.500

$\rho_{i,j}(k)$ of s^2DES S6 box (Mean =-0.096)

k	1	2	3	4	5	6
ρ_{12}	0.000	-0.125	-0.250	-0.250	-0.250	-0.126
ρ_{13}	-0.200	-0.125	0.000	0.250	-0.125	-0.387
ρ_{14}	-0.228	-0.125	0.125	0.125	-0.250	0.250
ρ_{23}	-0.129	0.125	-0.125	-0.625	0.125	0.098
ρ_{24}	0.000	-0.250	0.000	-0.500	-0.125	0.000
ρ_{34}	0.163	0.000	-0.125	0.250	-0.375	-0.258

D.7 s^2DES S7-box

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

$(p_{i,j})$ of s^2DES S7 box (Mean =0.516)

Bit	1	2	3	4
c_1	0.563	0.688	0.438	0.688
c_2	0.500	0.500	0.500	0.500
c_3	0.500	0.500	0.500	0.500
c_4	0.500	0.500	0.500	0.500
c_5	0.500	0.500	0.500	0.500
c_6	0.500	0.563	0.438	0.500

$\rho_{i,j}(k)$ of s^2DES S7 box (Mean =-0.105)

k	1	2	3	4	5	6
ρ_{12}	0.085	-0.375	-0.250	0.125	0.375	-0.126
ρ_{13}	-0.111	0.125	0.000	0.125	0.000	-0.126
ρ_{14}	-0.051	-0.625	-0.500	-0.125	-0.375	-0.250
ρ_{23}	0.051	-0.125	-0.125	0.125	0.250	-0.111
ρ_{24}	-0.164	0.125	0.000	-0.375	0.000	0.000
ρ_{34}	-0.085	-0.250	-0.125	-0.250	-0.375	-0.252

D.8 s^2DES S8-box

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

$(p_{i,j})$ of s^2DES S8 box (Mean = 0.508)

Bit	1	2	3	4
c_1	0.688	0.500	0.438	0.500
c_2	0.500	0.500	0.500	0.500
c_3	0.500	0.500	0.500	0.500
c_4	0.500	0.500	0.500	0.500
c_5	0.500	0.500	0.500	0.500
c_6	0.500	0.500	0.688	0.375

$\rho_{i,j}(k)$ of s^2DES S8 box (Mean =-0.101)

k	1	2	3	4	5	6
ρ_{12}	0.000	-0.500	-0.375	-0.250	0.000	-0.375
ρ_{13}	0.187	0.125	0.375	0.375	-0.375	-0.135
ρ_{14}	-0.405	0.125	-0.250	0.000	0.250	0.000
ρ_{23}	-0.252	-0.125	-0.375	-0.625	-0.125	0.135
ρ_{24}	-0.125	0.000	-0.250	-0.250	-0.125	-0.129
ρ_{34}	-0.126	0.125	-0.375	0.250	0.000	-0.035

Appendix E

Examples of s^2DES S-boxes

s^2DES S9-box

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

s^2DES S10-box

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

s^2DES S11-box

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

s^2DES S12-box

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

s^2DES S13-box

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

s^2DES S14-box

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

s^2DES S15-box

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

s^2DES S16-box

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

s^2DES S17-box

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

s^2DES S18-box

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

s^2DES S19-box

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

s^2DES S20-box

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

s^2DES S21-box

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

s^2DES S22-box

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

s^2DES S23-box

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

s^2DES S24-box

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

s^2DES S25-box

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

s^2DES S26-box

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

s^2DES S27-box

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

s^2DES S28-box

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

s^2DES S29-box

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

s^2DES S30-box

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

s^2DES S31-box

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

s^2DES S32-box

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

Bibliography

[1] C. Adams and S. Tavares, "The Structured Design of Cryptographically Good S-boxes", *To appear in J. of Cryptology*, 1990.

[2] C. Adams and S. Tavares, "The Use of Bent Sequences to Achieve Higher-Order Strict Avalanche Criterion in S-box Design", *To appear in IEE*, 1990.

[3] F. Ayoub, "Probabilistic Completeness of Substitution–Permutation Encryption Networks", IEE, Vol.129, E, 5, pp195–199, Sep., 1982.

[4] S. Babbage, "On the Relevance of the Strict Avalanche Criterion", Electronics Letters, Vol.26, No.7, pp.461-462, 29th Mar., 1990.

[5] K.G. Beauchamp, *Applications of Walsh Functions with an Introduction to Sequency Theory*, Academic Press, 1984.

[6] H. Beker and F. Piper, *Cipher Systems : The Protection of Communications*, John Wiley and Sons, 1982.

[7] T. Beth and F. Piper, "The Stop-and-go Generator", Proc. of EUROCRYPT'85, 1985.

[8] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", Abstracts of CRYPTO'90, 1990.

[9] B.den Boer, "Cryptanalysis of FEAL", Advances in Cryptology, Proc. of EUROCRYPT'87, Springer-Verlag, pp.167–173, 1989.

[10] E.F. Brickell, J.H. Moore, and M.R. Purtill, "Structure in the S-boxes of the DES", Advances in Cryptology, Proc. of CRYPTO'86, Springer-Verlag, pp.3–8, 1986.

[11] L. Brown, J. Pieprzyk, and J. Seberry, "LOKI – a Cryptographic Primitive for Authentication and Secrecy", Proc. of AUSCRYPT'90, 1990.

[12] J.M. Carroll and L.E. Robbins, "Using Binary Derivatives to Test an Enhancement of DES", Cryptologia, Vol.12, pp.193–208, 1988.

[13] D. Chaum and J.H. Evertse, "Cryptanalysis of DES with a Reduced Number of Rounds", Advances in Cryptology, Proc. of CRYPTO'85, Springer-Verlag, pp.192–211, 1986.

[14] D. Coppersmith and E. Grossman, "Generators for Certain Alternating Groups with Applications to Cryptography", SIAM J. Appl. Math., Vol.19, No.4, pp.624–627, Dec., 1975.

[15] D.W. Davies and G.I.P. Parkin, "The Average Size of the Key Stream in Output Feedback Mode", Proc. of CRYPTO'82, pp97–98, 1982.

[16] D.W. Davies and W.L. Price, *Security for Computer Networks*, John Wiley and Sons, 1984.

[17] "Data Encryption Standard", National Bureau of Standards, Federal Information Processing Standard, Vol. 46, U.S.A., Jan., 1977.

[18] "DES Modes of Operation", National Bureau of Standards, Federal Information Processing Standard, Vol. 81, U.S.A., Jan., 1980.

[19] D. Denning, *Cryptography and Data Security*, Addison-Wesley Pub. , 1983.

[20] S. Even and O. Goldreich, "DES-like Functions can Generate the Alternating Group", IEEE Trans. on IT, Vol.29, pp863–865, 1983.

[21] J.H. Evertse, "Linear Structures in Block Ciphers", Advances in Cryptology, Proc. of EUROCRYPT'87, Springer-Verlag, pp.249–266, 1988.

[22] H. Feistel, "Cryptography and Computer Privacy", Scientific American, Vol.228, No.5, pp 15–23, 1973.

[23] R. Forré, "The Strict Avalanche Criterion : Spectral Properties of Boolean Functions and an Extended Definition", Proc. of CRYPTO'88, Springer-Verlag, 1988.

[24] R. Forré, "Problems and Methods Related to Cryptographic Applications of Smart Cards", Diss. ETH No. 9137, Swiss Federal Institute of Technology, 1990.

[25] R. Forré, "Methods and Instruments for Designing S-boxes", J. of Cryptology, Vol.2, No.3, pp.115–130, 1990.

[26] W. Fumy, "On the F-function of FEAL", Advances in Cryptology, Proc. of CRYPTO'87, Springer-Verlag, pp.434–437, 1988.

[27] P.R. Geffe, "How to Protect Data with Ciphers That Are Really Hard to Break", Electronics, pp.99–101, 1973.1.

[28] D. Gollman, "Pseudo Random Properties of Cascade Connection of Clock Controlled Shift Registers", Proc. of EUROCRYPT'84, Lecture Notes in Computer Sciences, Vol.209, Springer–Verlag, 1984.

[29] S.W. Golomb, *Shift Register Sequences*, Aegean Park Press, 1982.

[30] H. Gustafson, E. Dawson, and B. Caelli, "Comparison of Block Ciphers", Proc. of AUSCRYPT'90, pp. 153–165, 1990.

[31] M. Hellman, R. Merkle, R. Schroeppel, L. Washington, W. Diffie, S. Pohlig and P. Schweitzer, "Results of an Initial Attempt to Analyze the NBS Data Encryption Standard", Information Systems Laboratory Report, Stanford University, 1976.

[32] S.M. Jennings, "Multiplexed Sequences: Some Properties of the Minimum Polynomial", Lecture Notes in Computer Sciences, Vol.149, Cryptography, Springer–Verlag, 1982.

[33] R.R. Juenman, "Analysis of Certain Aspects of Output Feedback Mode", Proc. of CRYPTO'82, pp99–127, 1982.

[34] B.S. Kaliski Jr, R.L. Rivest, and A.T. Sherman, "Is the Data Encryption Standard a Group ?", J.Cryptology, Vol.1, pp3–36, 1988.

[35] J.B. Kam and G.I. Davida, "Structured Design of Substitution–Permutation Encryption Network", IEEE Trans. on Comput., Vol.C-28, No.10, pp.747–753, Oct., 1979.

[36] E.L. Key, "An Analysis of the Structure and Complexity of Nonlinear Binary Sequence Generators", IEEE Trans. on IT, Vol.22, No.6, pp.732–736, Nov., 1976.

[37] D.E. Knuth, The Art of Computer Programming : Vol 2. Seminumerical Algorithms, 2nd Edition, Addison Wesley Pub., 1981.

[38] A.N. Kolmogorov, "Three Approaches to the Quantitative Definition of Information", Prob. Inform. Transmission, Vol.1, 1965.

[39] A.G. Konheim, *Cryptography A Primer*, John Wiley and Sons Inc. 1981.

[40] P.V. Kumar, R.A. Scholtz, and L.R. Welch, "Generalized Bent Functions and their Properties", J. of Combinatorial Theory(A),Vol.40, pp.90–107,1985.

[41] A. Lempel and J. Ziv, "On the Complexity of Finite Sequences", IEEE Trans. on IT, Vol.22, No.1, pp.75–81, Jan., 1976.

[42] A. Lempel and M. Cohn, "Maximal Families of Bent Sequences", IEEE Trans. on IT., Vol.28, No.6, pp.865–868, Nov., 1982.

[43] A.K. Leung and S. Tavares, "Sequence Complexity as a Test for Cryptographic Systems", Proc. of CRYPTO'84, Springer-Verlag, 1984.

[44] S. Lloyd, "Counting Functions Satisfying a Higher Order Strict Avalanche Criterion", Proc. of EUROCYRPT'89, Springer-Verlag, 1989.

[45] S. Lloyd, "Properties of Binary Functions", Proc. of EUROCRYPT'90, 1990.

[46] G. Longo, *Secure Digital Communications*, CISM Courses and Lectures No.279, Springer-Verlag, 1983

[47] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error Correcting Codes*, North-Holland, 1983.

[48] P. Martin-Loef, "The Definition of Random Sequences", Information and Control, Vol.9, pp.602–619, 1966.

[49] J.L. Massey, "Shift-Register Synthesis and BCH Decoding", IEEE Trans. on IT, Vol.15, No.1, pp.122–127, Jan., 1969.

[50] W. Meier and O. Staffelbach, "Nonlinearity Criteria for Cryptographic Functions", Proc. of EUROCRYPT'89, 1989.

[51] C. Mitchell, "Enumerating Boolean Functions of Cryptographic Sigificance", J. of Cryptology, Vol.2, No.3, pp.155–170, 1990.

[52] S. Miyaguchi, A. Shiraishi, and A. Shimizu, "Fast Data Encryption Algorithm FEAL–8 (*in Japanese*)", Electr. Comm. Lab. Tech. J., NTT, Vol.37, No.4/5, pp.321–327, 1988.

[53] J.H. Moore and G.J. Simmons, "Cycle Structure of the DES with Weak and Semi-weak Keys", Proc. of CRYPTO'86, pp3–32, 1986

[54] J.H. Moore and G.J. Simmons, "Cycle Structure of the DES for Keys Having Palindromic (or Antipalindromic) Sequences of Round Keys", IEEE Trans.on Software Eng., Vol.13, No.2, pp262–273, Feb., 1987,

[55] J.D. Olsen, R.A. Scholtz, and L.R. Welch, "Bent-function Sequences", IEEE Trans. on IT., Vol.28, No.6, pp.858–864, Nov., 1982.

[56] J. Pieprzyk, "Nonlinear Functions and their Application to Cryptography", Archiwum Automatyki i Telemechaniki, 3-4, pp.311–323, 1985.

[57] J.P. Pieprzyk, "Non-linearity of Exponent Permutations", Proc. of EUROCRYPT'89, Springer-Verlag, 1989.

[58] J. Pieprzyk and G. Finkelstein, "Towards Effective Nonlinear Cryptosystem Design", IEE, Pt.E, Vol.135, pp.325–335, 1988.

[59] V.S. Pless, "Encryption Schemes for Computer Confidentiality", IEEE Trans. on Comp., Vol.26, pp1133–1136, Nov., 1977.

[60] B. Preneel, W.V. Leekwijck, L.V. Linden, R. Govaerts, and J. Vandewalle, "Propagation Characteristics of Boolean Functions", Proc. of EUROCRYPT'90, 1990.

[61] P.W. Purdon and J.H. Williams, "Cycle Length in a Random Function", Trans. of AMS, Vol.133, pp547–551, 1968.

[62] R. Rivest, A. Shamir, and A. Adleman, "On Digital Signatures and Public Key Cryptosystems", CACM, Vol.21, pp.210–126, 1978.

[63] O.S. Rothaus, "On "Bent"Functions", J. of Combinatorial Theory(A), Vol.20, pp.300–305, 1976.

[64] F. Rubin, "Decrypting a Stream Cipher Based on J–K Filp–Flops", IEEE Trans. on Comp, Vol.28, pp483–487, Jul., 1979.

[65] R.A. Rueppel, "Linear Complexity and Random Sequences", Proc. of EUROCRYPT'85, 1985.

[66] R.A. Rueppel, *Analysis and Design of Stream Ciphers*, Springer-Verlag, Berlin, 1986.

[67] R. Sedgewick, T.G. Szymanski, and A.C. Yao, "The Complexity of Finding Cycles in Periodic Functions", SIAM J. Computing, Vol.11, No.2, pp376–390, May, 1982.

[68] A. Shamir, "On the Security of DES", Advances in Cryptology, Proc. of CRYPTO'85, Springer–Verlag, pp.280–281, 1986.

[69] C.E. Shannon, "Communication Theory of Secrecy Systems", BSTJ, Vol. 28, pp.656–715, Oct., 1949.

[70] L.A. Shepp and S.P. Lloyd, "Ordered Cycle Lengths in a Random Permutation", Trans. of AMS, Vol.121, pp340–357, 1966.

[71] T. Siegenthaler, "Correlation-immunity of Nonlinear Combining Functions for Cryptographic Applications", IEEE Trans. on IT, Vol.30, No.5, pp.776–780, 1984.

[72] T. Siegenthaler, "Decrypting a Class of Stream Ciphers Using Ciphertext Only", IEEE Trans. on Computers, Vol.34, No.1, pp.81–85, Jan., 1985.

[73] M.K. Simon, J.K. Omura, R.A. Scholtz, and B.K. Levitt, *Spread Spectrum Communications*, Vol.1, Computer Science Press, 1985.

[74] B. Smeets, "A Note on Sequences Generated by Clock Controlled Shift Registers", Proc. of EUROCRYPT'85, 1985.

[75] R.J. Solomonov, "A Formal Theory of Inductive Inference", Part I, Information and Control, Vol.7, 1964.

[76] K. Takaraki, K. Sasaki, and F. Nakagawa, "Multi–Media Encryption Algorithm (*in Japanese*)", 89-MDP-40-5, 1989.1.19.

[77] A.F. Webster, "Plaintext/Ciphertext Bit Dependencies in Cryptographic Systems", Master's Thesis, Department of Electrical Engineering, Queen's University, CANADA, 1985.

[78] A.F. Webster and S.E. Tavares, "On the Design of S-boxes", Proc. of CRYPTO'85, Springer-Verlag, 1985.

[79] Guo-Zhen Xias and J.L. Massey, "A Spectral Characterization of Correlation-Immune Combing Functions", IEEE Trans. on IT, Vol.34, No.3, pp.569–571, May, 1988.

[80] R. Yarlaghadda and J.E. Hershey, "Analysis and Synthesis of Bent Sequences", IEE, Vol.136, Pt.E, No.2, pp.112–123, Mar., 1989.

[81] K.C. Zeng, J.H. Yang, and Z.T. Dai, "Patterns of Entrophy Drop of the KEY in an S-box of the DES", Advances in Cryptology, Proc. of CRYPTO'87, Springer-Verlag, pp.438–444, 1988.

[82] N. Zierler, "Linear Recurring Sequences", J.Soc. Indust. Appl. Math., Vol.7, 1959.

[83] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression", IEEE Trans. on IT, Vol.23, No.3, pp.337–343, May, 1977.