論 文

암호 함수의 구성 방법에 관한 연구

正會員 金 光 兆* 正會員 松本 勉* 正會員 今井 秀樹* 正會員 朴 漢 奎**

A Study on the Construction Methods of Cryptographic Functions

Kwang Jo KIM,* Tsutomu Matsumoto,* Hideki Imai,* Han Kyu PARK** *Regular Members*

要 約 DES like 암호계에서 s(ubstitution)-box는 암호계의 비선형성과 안전성을 결정하는 가장 중요한 요소이다.
본 논문에서는 입력의 임의의 한 비트의 변화에 대하여 모든 출력 비트가 1/2의 확률로 변화하는 SAC(Strict Avalanche Criterion) 조건을 만족하는 비선형 S-box의 구성 방법과 아울러, 최대차 SAC 조건을 만족하는 bijective S-box의 구성 방법에 대하여도 제안하였다. 또한, 제안한 구성 방법에 의한 S-box의 실례를 제시하였다.

ABSTRACT In a DES-like cryptosystem, the S(ubstitution)-boxes determine its cryptographic strength as well as its nonlinearity. When in an S-box a part of the output depends on a part of the input, it can be broken by the chosen plaintext attack. To prevent this attack, every output bit should change with a probability of one half whenever a single input bit is complemented. We call this criterion Strict Avalanche Criterion(SAC), which was proposed by Webster and Tavares. In this paper, we propose simple construction methods of Boolean functions satisfying the SAC and bijective functions satisfying the maximum order SAC in order to design cryptographically desirable S-boxes. Also, practical construction examples of S-boxes are provided.

I. Introduction

We are now living in a so-called era of information society. A variety of electronic communications such as FAX, teletex, electronic mail, mobile telephone, *etc.* are replacing paper media in a rapid way. This not only makes a great impact in our ordinary lives but also increases the amount of the information to available to an eavesdropper, and to make the act of eavesdropping easier.

When valuable or secret information is stored or transmitted, it is often protected physically through the use of safes, armed couriers, shielded cables, and the like. As electronic

*橫浜國立大學 工學部 電子情報工學科
Division of Electrical and Computer Engineering
Faculty of Engineering, Yokohama National University
**延世大學校 電子工學科
Dept. of Electronics, Yonsei University.
論文番號 : 91-9 (接受1990. 12. 8)

forms of communication and storage take over from their predecessors, however, such meas ures often become inapplicable, insufficient or uneconomical, and other techniques should be employed.

One of the practical techniques is cryptog raphy : the use of ransformation of data int ended to make the data useless to one's opp onents. Such transformations can solve two major problems of information security, the privacy problem : preventing an opponent from extracting information from a communication channel, and the authentication problem : preventing an opponent from injecting false data into the channel or altering messages so that their meaning is changed.

Let us briefly review the historical backgr ound of symmetric cryptosystems (which means that encryption and decryption keys are the same). In 1949, Shannon[1] proposed the out standing notion of mixing transform ations, which randomly distributes the mean ingful messages uniformly over the set of all possible ciphertext messages. Mixing transfo rmations could be implemented by alternatively applying permutation and substitution boxes.

In 1977, NBS(National Bureau of Standards) adopted DES(Data Encryption Standard)[5] which was proposed by IBM as a federal standard of commercial cryptosystem in the United States. NTT in Japan proposed in 19 88 a new cryptosystem called FEAL(Fast Encryption ALgorithm)[10] whose purpose is to make encryption and decryption operation faster than DES in the software or hardware implementation. FEAL has two types : FEAL 4 and FEAL-8 depending on the number of rounds. In 1990, Brown *et al*[17] proposed one symmetric cryptosystems called LOKI for banking securities in Australia.

It could be considered that the above men tioned symmetric cryptosystems are the good design practices of the mixing transformations. We call these cryptosystems "DES like cryp tosystme" altogether.

In DES like cryptosystems, the S(ubstitut ion) boxes behave like nonlinear table tran slation mappings from some numbers of input bits to some numbers of output bits and play an important role in the algorithm. This is because the remainder of the algorithm is linear, severe weakness in the S-boxes can therefore lead to a cryptosystem which can be easily broken.

For the good S box design, Webster and Tavares[8] proposed the concept of Strict Avalanche Criterion (abbreviated as "SAC") - defined in Section 2 - in order to combine the notions of the *avalanche effect*[2] and the *completeness*[6]. Forre'[11] extended this con cept of SAC into a higher order SAC and discussed the Walsh spectral properties of S boxes satisfying the SAC.

On the other hand, Hellman *et al*[3] have pointed out that any cryptosystem which implements linear or affine functions can be easily broken. By employing exponentiation over $GF(2^n)$, Pieprzyk[12] proposed one con struction method of S boxes satisfying the maximum nonlinearity dfined by Rueppel[9]. Also, Meier and Staffelbach[14] discussed the cryptographic application of bent functions[4][7] for satisfying the nonlinearity criterion.

However, there are rare publications propo sing constructions of cryptographic functions satisfying the SAC as well as the nonlinearity criterion. We will propose simple construction methods of cryptographic functions satisfying these two important criteria.

The organization of this paper is as follows

: After defining the necessary notation and definitions in Section 2, we state our previous results [19][20] on the proved properties of Boolean functions satisfying the SAC in Section 3.

In Section 4, we propose two methods to enlarge the given Boolean functions(s) satisfying the SAC into any input size. One method is to enlarge a 2-bit input function into a $2l$-bit($l \geq 2$) input function satisfying the SAC by using the Kronecker(or direct) product. The other one is to enlarge two distinct n-bit input Boolean functions satisfying the SAC into an unique $(n+1)$-bit input Boolean function satisfying the SAC as many as possible.

In Section 5, Lloyd[13] suggested that the counting function of the Boolean function satisfying the maximum order SAC is 2^{n+1}. We will extend Lloyd's result and suggest the counting function of a balanced and unbalanced Boolean function satisfying the maximum order SAC. Also we propose a new construction method of all bijective functions satisfying the maximum order SAC.

In Section 6, we make some concluding remarks and suggest some left problems.

II. Notation and Definition

We define here some necessary notation and definitions. Let Z denote the set of integers and Z_2 denote the n dimensional vector space over the finite field $Z_2 = GF(2)$. Also let $\oplus$ denote the addition over Z_2, or, thebit-wise exclusive-or.

Notation 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 specific 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

$$< f_m >, < f_{m-1} >, \ldots, < f_2 >, \\ < f_1 > \text{ as } < f >= \textstyle\sum_{j=1}^{m} < f_j > \cdot 2^{j-1}.$$

Throughout this paper, $wt(\ )$ denotes the Hamming weight function and $c_i^{(n)}$ denotes and n dimensional vector with Hamming weight 1 at the i th position. $f_i \| f_j$ denotes the concatenation of $\langle f_i \rangle$ and $\langle f_j \rangle$.

Let us define one of the most important criteria to design cryptographic functions.

Definition 1 (SAC) *We say that a function $f: Z_2^n \rightarrow Z_2^m$ satisfies the SAC, if for all i $(1 \leq i \leq n)$ there hold the following equations :*

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

When $f: Z_2^n \rightarrow Z_2$ satisfies the SAC, f is sometimes referred to as a Boolean function satisfying the SAC.

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.

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 bo able to detect these relations and use this inform ation to aid the search for the key. And because any lower dimensional space appro ximation of a mapping yields a wrong result in 25%[18] of the cases, cryptographic func tions satisfying the SAC play significant roles in cryptography.

Forre'[11] extended this definition of the SAC into a higher order SAC.

Definition 2 (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 $i \in \{1, 2, \cdots n\}$, and for $c=0$ and $c=1$.*

Naturally, the SAC defined in **Definition 1** can be said of 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 to the 0 *th* order SAC and $n\cdot(n-1)$ checks correspond to 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 $f(X)$ are kept constant.

Definition 3 (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 posi tions and of the values of k constant bits.)*

Therefore, verifying whether an n bit input Boolean function satisfies the *m-th* order SAC or notr equires as most $n+n\cdot(n-1)+\binom{n}{2}(n-2)+\cdots+\binom{n}{k}(n-k)$ checks.

Afterward, when we say that a function satisfies the SAC without specifying the order, the function as 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 fun ction satisfying the maximum order SAC in the sense that the $(n-2)$ *th* order SAC is maximally achievable in Boolean functions.

III. Properties of Boolean Function Satisfying the SAC

In this section, we outline our previous results on the properties of Boolean function satisfying the SAC.

1. Previous Results

We[19] [20] proved that any function satisfying the SAC holds the folowing prope rties.

1. It is neither linear nor a ffine.
2. For $n=1$, or 2, any bijective function f form Z_2^n into Z_2^n never satisfies the SAC.
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 comp osite function $gofoe: Z_2^n \rightarrow Z_2^m$ satisfies the SAC.

Due to Property 1, if we use any function satisfying the SAC for the S boxes of a DES like cryptosystem, the cryptosystem can be guaranteed to be neither linear nor affine. Property 2 ensures that we must treat at least quadratic function of at least three variables in order to obtain bijective functions satisfying the SAC. And by using Property 3, when we find a function satisfying the SAC, we can make many different functions satisfying the SAC with the same number of input bits.

2. Basic Functions Satisfying the SAC

When the number of input bits is 2 in a Boolean function, we searched all Boolean functions and obtained those satisfying the SAC. Table 1 shows all 2 bit input Boolean functions satisfying the SAC. The inside of (·) in the table is a hexadecimal representation of a function. We refer to these functions as "basic functions satisfying the SAC".

Table 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)

By employing these basic functions, we will construct a function satisfying the SAC in Section 4 and Section 5.

Ⅳ. Construction of Boolean function satisfying the SAC

Experimental results gave us that as the number of input bits(≥5) increases in a Boolean function, it becomes more and more difficult to find even one Boolean function satisfying the SAC by random search on an engineering workstation. So, it is necessary to construct Boolean functions satisfying the SAC from a relatively smaller number of inputs to any arbitrary number of inputs in a systematic way.

Here we propose two construction methods of a Boolean function satisfying the SAC from given basic functions satisfying the SAC.

1. 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 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 $f(x)$ is a 0/1 valued function.

Theorem 1 *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 g extended by*

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

satisfies the SAC.

Proof : See the **Appendix.**

Example 1 *An 1/−1 valued basic function satisfying the SAC is given below.*

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

Then,

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

$\langle \hat{g}^2\rangle$ is an 1/−1 valued 4-bit input Boolean function satisfying the SAC.

2. Construction by Concatenation

We can enlarge two given distinct n bit input Boolean functions satisfying the SAC into one $(n+1)$ bit input Boolean function satisfying the SAC.

Theorem 2 *When two distinct Boolean functions f_i and f_j : $Z_2^n \rightarrow Z_2$ satisfying the SAC are given, the concatenated Boolean function g : $Z_2^{n+1} \rightarrow Z_2$ ie.,*

$$g = f_i || f_j \qquad (2)$$

satisfies the SAC if and only if

$$wt(< f_i > \oplus < f_j >) = 2^{n-1}. \qquad (3)$$

Proof : We will show that $\sum_{x\in Z_2^{n+1}} g(x)\oplus g(x\oplus c_k^{(n+1)})=2^n$ for any $k\in\{1, 2,\cdots, n+1\}$. Let us denote the $(n+1)$ bit input vector of g as $x=(x_{n+1}, x')$ where $x'=(x_n, x_{n-1},\cdots, x_1)$. Also let x_0 dentote $(0, x')$and let x_1 denote $(1, x')$.

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 $k=1, 2,\cdots, n$,

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

When $k=n+1$, let $\langle g_l\rangle$ denote the left half of $\langle g\rangle$ and let $\langle g_r\rangle$ denote the right half of $\langle g\rangle$. In this case, the condition of SAC ness can be alternatively checked like,

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

This is equivalent to Equation (3).

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

Example 2 *When* $\langle f_1\rangle=[0,0,0,1]$ *and* $\langle f_2\rangle=[0,0,1,0]$,

$\langle g\rangle=\langle f_1\rangle||\langle f_2\rangle$

$=[0,0,0,1,0,0,1,0]$

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 thismethod are summarized in Table 2. In Table 2, A_n denotes the number of all n-bit input Boolean functions, B_n denote 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 2. 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 function 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 as the function to be concatenated in this method.

V. Construction of Bijective Functions Satisfying the Maximum Order SAC

An n bit bijective function ensures that all possible 2^n n-bit input vector will map to distinct output (*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 resultant input and output vectors of S boxes with some number of input bits fixed to constant, a cryptanalyst will use this relationship to break a cryptosystem.

Therefore, to prevent this possible way of cryptanalysis, we propose th 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 [11] proposed on recursive construction method which can generate Boolean functions satisfying the maximum order SAC. Her method can generate, for example, 3 bit input Boolean functions $g^{(3)}(x)$ satisfying the maximum SAC from the basic functions $f_i^{(2)}(x)$, $f_j^{(2)}(x)$ satisfying the SAC by applying Equation (4) recursively.

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

$$g^{(n+1)}(x) = x_{n+1} \cdot f_i^{(n)}(x) + \overline{x_{n+1}} \cdot f_j^{(n)}(x) \quad (4)$$

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

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[16] proposed one structured method that an enlarged Boolean function always satisfies the maximum order SAC. They utilized the idea of the construction method[15] of bent functions.

Let $f^{(n)} : Z_2^n \to Z_2$ 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 $\langle f^{(n)}\rangle = [b_0, b_1, \cdots, b_{2^n-1}]$, $\langle f_r^{(n)}\rangle = [b_{2^n-1}, b_{2^n-2}, \cdots, b_0]$) and let $\overline{f_r^{(n)}}$ be the bitwise complement of $f_r^{(n)}$.

If n is even,

$$g_1^{(n+1)} = f^{(n)} \| f_r^{(n)} \qquad (5)$$

$$g_2^{(n+1)} = f^{(n)} \| \overline{f_r^{(n)}} \qquad (6)$$

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

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

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 in [16] 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 se 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

Table 3. 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 0011(DB)
14	1101 0100(D4)
15	1110 0111(E7)
16	1110 1000(E8)

Table 4. All 4 bit input Boolean functions satisfying the 2 nd order SAC.

No.	01～15	No.	01～15
1	(188E)	17	(7118)
2	(1871)	18	(71E7)
3	(1781)	19	(7E17)
4	(177E)	20	(7EE8)
5	(244D)	21	(B221)
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)

functions satisfying the maximum order SAC in Table 3 (or Table 4).

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

Theorem 3 *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 ones are unbalanced. Therefore, when n is odd, the total number of the balanced Boolean functions safisfying the maximum order SAC is 2^n. When n is even, all functions satisfying the maximum order SAC are unbalanced.*

Proof : Let F_n^{max}, $F_{n:odd}^{max}$, and $Ff_{n:even}^{max}$ 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[4][7] that each bent function has Hamming weight $2^{n-1} \pm 2^{n/2-1}$ (*i.e.*, unbalanced). So any bitwise complement of abent 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 Equation (5) 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 Equation (6) 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 [13] 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 Equation (7) has Hamming weight

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

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 of obtain a bijective function satisfying the maximum order SAC, the number of input bit should be odd.

[Method K]

By using **Theorem 3**, we can generate all bijective functions satisfying the maximum order SAC. In order that n Boolean functions $f_1, f_2, \cdots f_n$ satisfying the maximum order SAC become an n-bit bijective function, the following equation gives as a necessary and sufficient condition that n-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.

$$wt(\bigoplus_{i=1}^{n} a_i f_i) = 2^{n-1} \qquad (8)$$

where $a_i \in \{0, 1\}$ and $(a_1, a_2, \cdots, a_n) \neq (0, 0 \cdots, 0)$. In other words, the Hamming weight of any nonzero linear combinations of f_i should be 2^{n-1}. To verift Equation(8) requires $\binom{n}{2} + \binom{n}{3} + \cdots + \binom{n}{n}$ 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 Equation(8). **Method K** can be illustrated as Fig.(1).

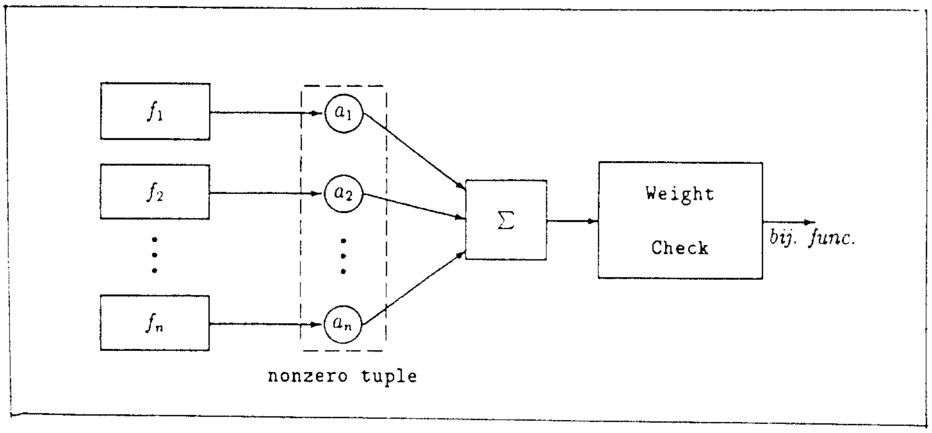


Figure 1. Illustrative description of **Method K**

Table 5. 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

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

We can consider the combinatorial search of **Method K.** We searched $(_3)\times 56$ combinatorial cases and obtained 32 3 bit bijective functions satisfying the 1 *st* order SAC. Table 5 shows examples of generated 3 bit bijective function by this method.

Also 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 to that of filtering from all permutation 8!. 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.

Ⅵ. Concluding Remarks

We propose two simple methods to enlarge the given Boolean function(s) satisfying the SAC into any input size. One method is to enlarge a 2-bit input function into a $2l$-bit ($l \geq 2$) input function satisfying the SAC by using the Kronecker product. The other one is to enlarge two distince n-bit input Boolean functions satisfying the SAC into an unique $(n+1)$-bit input Boolean satisfying the SAC as many as possible. The proposed construction method can be useful to count the lower bound or upper bound of the number of functions satisfying the SAC.

Among the total 2^{n-1} Boolean functions satisfying the maximum order SAC, when n is even,all Boolean functions satisfying the maximum order SAC are unbalanced and bent. However, when n is odd, the number of balanced Boolean functions satisfying the maximum order SAC is 2^n. The other half one sare unbalanced. Therefore, in the class of functions which are bijective and satis fying the max mum order SAC, there exist only bijective functions whose number of input bits is odd.

Also we proposed a new construction method of all bijective functions satisfying the maximum order SAC and suggested the practical generated examples of bijective S-boxes by employing our proposed method.

We think that the following topics require further research.

1. To generate all Boolean functions satisfying the SAC.
2. How to combine functions satisfying the SAC $f_i : Z_2^n \rightarrow Z_2^n$ into $g : Z_2^m \rightarrow Z_2^n$, $m > n$.
3. Measuring nonlinearity or any other criteria of a function satisfying the SAC.

This paper will help to design cryptographic functions for symmetric cryptosystems, pseudorandom number generators, nonlinear combiner for stream ciphers, hash functions, *etc*.

Appendix : Proof of Theorem 1

We can prove this theorem by mathematical induction.

(Basis) $l=2$. $\langle \hat{g}^2 \rangle = \langle \hat{f} \rangle \otimes \langle \hat{f} \rangle$. Because $\hat{f}(x)$ satisfies the SAC, it holds that $\sum_{x \in Z_2} f(x) \oplus f(x \oplus c_j^{(2)} = 2$ or $\sum_{x \in Z_2} \hat{f}(x)\hat{f}(x \oplus c_j^{(2)} = 0$ for $j = 1, 2$. Let $\langle \hat{f} \rangle = [x_0, x_1, x_2, x_3]$ 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,

$$\begin{aligned} &< \hat{g}^2 > \\ &\quad = < \hat{f} > \otimes < \hat{f} > \end{aligned}$$

$$= [x_0 < \hat{f} >, x_1 < \hat{f} >, x_2 < \hat{f} >, x_3 < \hat{f} >]$$

$$= [x_0^2, x_0x_1, x_0x_2, x_0x_3, x_0x_1, x_1^2, x_1x_2, x_1x_3,$$

$$x_0x_2, x_1x_2, x_2^2, x_2x_3, x_0x_3, x_1x_3, x_2x_3, x_3^2]$$

We must prove the $\sum_{x \in Z_2^4} \hat{g}^2(x)\hat{g}^2(x \oplus c_k)=0$ for any $k \in \{1, 2, 3, 4\}$.

(Case 1) When $k=1$,

$$\sum_{x \in Z_2^4} \hat{g}^2(x)\hat{g}^2(x \oplus c_1^{(4)}) =$$

$$x_0^2 \cdot x_0x_1 + x_0x_2 \cdot x_0x_3 + x_0x_1 \cdot x_1^2 + x_1x_2 \cdot x_1x_3$$

$$+x_0x_2 \cdot x_1x_2 + x_2^2 \cdot x_2x_3 + x_0x_3 \cdot x_1x_3 + x_2x_3 \cdot x_3^2$$

$$= x_0x_1 + x_2x_3 + x_0x_1 + x_2x_3$$

$$+x_0x_1 + x_2x_3 + x_0x_1 + x_2x_3$$

$$= 0.$$

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

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

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.*,

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

for all $j=\{1, 2, \cdots, 2n\}$. Let $\langle \hat{g}^n \rangle = [x_0, x_1, \cdots, x_{4^n-1}]$, then there hold the following equations.

$$x_0'x_1' + x_2'x_3' + \cdots + x_{4^n-2}'x_{4^n-1}' = 0 \ (\diamondsuit)$$

$$x_0'x_2' + x_1'x_3' + \cdots + x_{4^n-3}'x_{4^n-1}' = 0$$

$$\vdots$$

$$x_0'x_{2^n}' + x_1'x_{2^n+1}' + \cdots + x_{2^n-1}'x_{4^n-1}' = 0$$

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

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

On the other hand,

$$< \hat{g}^{n+1} > = < \hat{f} > \oplus < \hat{g}^n >$$

$$= [x_0 < \hat{g}^n >, x_1 < \hat{g}^n >, x_2 < \hat{g}^n >, x_3 < \hat{g}^n >].$$

When $k=1$, we can get that

$$\sum_{x \in Z_2^{2n+2}} \hat{g}^{n+1}(x)\hat{g}^{n+1}(x \oplus c_1^{(2n+2)})$$

$$= x_0^2 \cdot (\diamondsuit) + x_1^2 \cdot (\diamondsuit) + x_2^2 \cdot (\diamondsuit) + x_3^2 \cdot (\diamondsuit)$$

$$= 0.$$

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

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

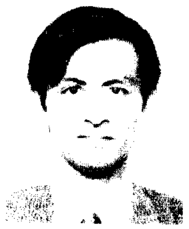
Thus, we complete the proof.

References

1. C.E. Shannon, "Communication Theory of Secrecy Systems", BSTJ, Vol. 28, pp. 656~715, Oct. 1949.
2. H. Feistel, "Cryptography and Computr Privacy", Scientific American, Vol. 228, No. 5, pp. 15~23, 1973.

3. 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.
4. O.S. Rothaus, "On "Bent" Functions", J. of Combinatorial Theory(A), Vol. 20, pp. 300~305, 1976.
5. "Data Encryption Standard", National Bureau of Standards, Federal Information Processing Standard, Vol. 46, U.S.A., Jan., 1977.
6. J.B. Kam and G.I. Davida, "Structured Design of Substitution Permutation Networks", IEEE Trans. on Comp., Vol. C-28, No. 10, pp. 747~753, Oct., 1979.
7. 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.
8. A.F. Webster and S.E. Tavares, "On the Design of S-boxes", Proc. of CRYPTO'85, Springer-Verlag, 1985.
9. R.A. Rueppel, *Analysis and Design of Stream Ciphers*, Springer Verlag, Berlin, 1986.
10. 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.
11. R. Forre', "The Strict Avalanche Criterion : Spectral Properties of Boolean Functions and an Extended Definition", Proc. of CRYPTO'88, Springer Verlag, 1988.
12. J. Pieprzyk, "Nonlinearity of Exponent Permutations", Proc. of EUROCRYPT'89, Springer Verlag, 1989.
13. S. Lloyd, "Counting Functions satisfying a Higher Order Strict Avalanche Criterion", Proc. of EUROCRYPT'89, Springer Verlag, 1989.
14. W. Meier and O. Staffelbach, "Nonlinearity Criteria for Cryptographic Functions", Proc. of EUROCRYPT'89, Springer Verlag, 1989.
15. R. Yarlaghadda and J.E. Hershey, "Analysis and Synthesis of Bent Sequences", IEEE, Vol. 136, Pt.E, No. 2, pp. 112~123, Mar., 1989.
16. C. Adams and S. Tavares, "The Use of Bent Sequences to Achieve Highex-Order Strict Avalanche Criterion in S-box Design", (Private Communication), 1990.
17. L. Brown, J. Pieprzyk, and J. Seberry, "LOKI a Cryptographic Primitive for Authentication and Secrecy", Proc. of AUSCRYPT'90, 1990.
18. S. Babbage, "On the Relevance of the Strict Avalanche Criterion", Electronics Letters, Vol. 26, No. 7, pp. 461~462, 29th Mar., 1990.
19. K. Kim, T. Matsumoto, and H. Imai, "On Generating Crptographically Desirable Substitutions", Trans. IEICE, Vol. E73, No. 7, Jul., 1990.
20. K. Kim, T. Matsumoto, and H. Imai, "A Recursive Construction Method of S-boxes Satisfying Strict Avalanche Criterion", Proc. of CRYPTO'90, 1990.

Kwangjo Kim was born in Kwangwon, Korea on April 10, 1954. He received the B. Eng. and M. Eng. degrees in electronic engineering from Yonsei University, Seoul, Korea in 1980 and 1983 respectively. Since 1979, he has been with Electronics and Telecommunications Research Institute, Daejeon, Korea. By ETRI's program, he is currently a candidate for the Ph. D. under the supervision of Professor Hideki Imai. His research interests include cryptography, communication security, microwave and their applications. He is a member of IEICE of Japan and KITE of Korea.



Tsutomu Matsumoto was born in Maebashi, Japan, on October 20, 1958. He received the B. Eng. and M. Eng. degrees in computer eng. both from Yokohama National University, Yokohama, Japan, in 1981 and 1983, respectively, and Ph. D. degree in electronic eng. from the University of Tokyo, Tokyo, Japan, in 1986. From 1986 to 1989, he was a Lecturer for Electrical and Computer Engineering at Yokohama National University. Since 1989, he has been an Associate Professor and is currently working in cryptography, complexity theory, computational mathematics, and their applications to information security. Dr. Matsumoto is a member of ACM, IACR, IEEE, IPSJ.

Hideki Imai was born in Shimane, Japan on May 31, 1943. He received the B.E., M.E., and Ph. D. degrees in electrical engineering from University of Tokyo in 1966, 1968, and 1971, respectively. He is currently a Professor and a Chairman in the Division of Electrical and Computer Engineering, Yokohama National University. His current research interests include information theory, coding theory, crytography and their applications. He is the author of three books and coauthor of several books. Dr. Imai is a member of IEEE, IEICE of Japan, IEEE of Japan, IPS of Japan, and ITE of Japan.

朴 漢 奎(Han Kyu PARK) 正會員
1941年 6月 21日生
1961年 2月 : 延世大學校 電氣工學科 卒業
1968年 2月 : 延世大學校 大學院 電氣工學科 卒業 工學碩士
1973年 : 불란서 파리大學校 (소르본느大學校) 博士課程修了 (D.E.A.)
1975年 : 불란서 파리 6 대학 (ph.D)
1976年 ~ 現在 : 延世大學校 電子工學科 敎授