J. Cryptology (1990) 3: 27–41

Journal of Cryptology

The Structured Design of Cryptographically Good S-Boxes[1]

Carlisle Adams and Stafford Tavares
Department of Electrical Engineering, Queen's University at Kingston, Kingston, Ontario, Canada K7L 3N6

Abstract. We describe a design procedure for the s-boxes of private key cryptosystems constructed as substitution–permutation networks (DES-like cryptosystems). Our procedure is proven to construct s-boxes which are bijective, are highly nonlinear, possess the strict avalanche criterion, and have output bits which act (vitually) independently when any single input bit is complemented. Furthermore, our procedure is very efficient: we have generated approximately 60 such 4×4 s-boxes in a few seconds of CPU time on a SUN workstation.

Key words. Cryptography, s-boxes, Boolean functions, Nonlinearity, Strict avalanche criterion.

1. Introduction

In the development of symmetric, or private-key, cryptosystems which are constructed as substitution–permutation (S–P) networks (i.e., "DES-like" systems), a significant portion of the time spent on design or on analysis is centered around the substitution boxes (s-boxes) of the algorithm. This is because the remainder of the algorithm is linear; severe weaknesses in the s-boxes can therefore lead to a cryptosystem which is easily broken.

In this work we select four properties which we feel any cryptographically good s-box should possess and describe a design procedure which is guaranteed to produce s-boxes possessing all four of these properties. This gives us new insight into the design of good s-boxes, a topic of considerable interest in the cryptographic community, especially since the middle to late 1970s when DES first gained widespread attention. Furthermore, it also allows us to generate quickly and easily s-boxes which can be used in the development of new private-key cryptosystems, an area of renewed importance since the increasing power and speed of supercomputers, mainframes, and workstations make early fears about the relatively small keysize of DES increasingly relevant.

[1] Date received: September 12, 1989. Date revised: February 28, 1990. This work was partially supported by a grant from the Natural Sciences and Engineering Research Council of Canada.

2. Background

S-boxes, first introduced by Shannon in [23] and described more explicitly in [8] and [9], provide the cryptograhic strength of S–P networks: weak s-boxes can lead to weak cryptosystems (see [11] and [6], for example). This has made the s-box an important object of study in the design and analysis of certain types of cryptosystems. Of particular interest are the s-boxes of the Data Encryption Standard (DES) [16] since some of the design and evaluation criteria regarding these s-boxes have never been published. As a result, the cryptographic community has long speculated on how the DES s-boxes were chosen and has spent considerable effort in analyzing their properties [12], [22], [3], [26]. In this paper we describe a structured, methodical design procedure which will *provably* construct s-boxes with cryptographically desirable properties.

Note that, in general, s-boxes are a substitution (mapping) from m-bit vectors to n-bit vectors; m and n need not be equal. The s-boxes in DES, for example, are 6×4 bit mappings. In our work we deal with the case $m = n$ (i.e., we construct $n \times n$ bit s-boxes). This is not the restriction it initially appears since it is well known that two of the six "external" inputs in a DES s-box serve only to select one of the four "internal" 4×4 bit s-boxes (in other words, it may be important to know how to design $n \times n$ bit s-boxes in order to design $m \times n$ bit s-boxes). The general problem of combining 2^{m-n} good $n \times n$ bit s-boxes into a single $m \times n$ bit s-box with similar properties is nontrivial but can be solved using an extension of the work reported in this paper. The details of this aspect of s-box design will be given in an upcoming paper.

3. S-Box Design and Evaluation

The design and evaluation of an s-box are conceptually linked because the ways in which an s-box will eventually be evaluated influence or constrain the ways it can be designed. For example, one obvious design constraint is that the output bits of an s-box not be simply a permutation of its input bits; this follows from the evaluation criterion that an s-box be nonlinear. In this work we follow this process of finding design procedures which lead to s-boxes which satisfy certain evaluation criteria. Our interest, however, is primarily in design rules which can be proven to yield such s-boxes. The aim of this approach is clear: by integrating as many evaluation criteria and their corresponding design rules as possible into one overall procedure, we may be able to design quickly and easily cryptographically good s-boxes for private-key algorithms.

4. Evaluation Criteria

There are several properties which we feel to be cryptographically desirable in an s-box. We can regard these as our evaluation criteria, then, and use them to guide our s-box design. They are:

(1) bijection;
(2) nonlinearity;

(3) strict avalanche;
(4) independence of output bits.

Property (1) simply ensures that all possible 2^n n-bit input vectors will map to distinct output vectors (i.e., that the s-box is a permutation of the integers from 0 to $2^n - 1$); this is a necessary condition for invertibility of the s-box (which may or may not be important, depending on the structure of the surrounding network) and ensures that all output vectors appear once (which guarantees that all possible input vectors are available to the next stage in the network). Property (2) ensures that the s-box is not a linear mapping from input vectors to output vectors (since this would render the entire cryptosystem easily breakable). Property (3), the strict avalanche criterion (SAC) [25], requires that for every input bit i, inverting bit i causes output bit j to be inverted 50% of the time (over all possible input vectors), for all j. Such s-boxes exhibit what is generally referred to as 'good avalanche," where inverting any input bit i causes approximately half of the output bits to be inverted; this is equivalent to good "diffusion" [8] and extends the property of "completeness" defined in [13]. To avoid any confusion on the part of the reader, perhaps we should recall that the SAC, as originally defined in [24] and [25], applies to s-boxes or to full cryptosystems. It is a quantitative measure of "good avalanche effect" and it is this property that we desire to design into our s-boxes. Recently, Forre [10] has applied the definition of SAC to binary Boolean functions, which are then referred to as SAC-fulfilling functions. We make use of these functions to construct SAC-fulfilling $n \times n$ bit s-boxes. Property (4), independence of output bits (see [25]), ensures that any two output bits i and j act "independently" of each other; that is, bit i and j are not equal to each other significantly more, or significantly less, than half the time (over all possible input vectors). This is necessary since [24] has shown how the search space can be reduced in certain attacks if the correlation between two output bits is significantly other than zero.

Note that the property of s-box strict avalanche satisfies most of the "s-box design principles" released by IBM and NSA regarding DES [2], [3] most of the time (since strict avalanche deals with *expected* results over all inputs). One "design principle" not dealt with by SAC is that if the two middle input bits to an s-box are flipped simultaneously, two or more output bits will change (SAC deals specifically with flipping single input bits). The other design principle not dealt with by property (3) has to do with the combining of four 4×4 bit s-boxes into one 6×4 bit s-box. As we have stated, this is not of concern to us at this point; we address this issue in an upcoming paper.

We therefore restrict our attention to designing s-boxes which will satisfy the four properties listed above.

5. S-Box Design

Most of the published work on s-box design has attempted to find good s-boxes by generating them randomly, checking them against chosen evaluation criteria, and throwing away those which fail to meet these criteria [25], [4] (see also [21] for an interesting approach to the random generation process). In [3], s-boxes were

generated one entry at a time, with each new entry randomly chosen among those possible entries which could satisfy the criteria being studied. Another notable approach to s-box design is described in recent work by Pieprzyk and Finkelstein [18]. Here the authors build $n \times n$ bit s-boxes by constructing a Boolean function of the input bits and using this as one of the output bits. This Boolean function must have certain special properties (a given "weight" and "nonlinearity"), but the authors give an equation for constructing a function which is guaranteed to satisfy these conditions. The remaining $n - 1$ output bits are also formed from Boolean functions, but these functions are simple variations of the initial function.

Pieprzyk and Finkelstein then go on to define the nonlinearity of an s-box S as the sum of the nonlinearities of its n Boolean functions plus the sum of the nonlinearities of the n Boolean functions which make up its inverse S^{-1}. Alternatively (for a cryptographic context), they define it as the minimum nonlinearity over all $2n$ Boolean functions. Given their method of s-box construction, either definition seems to be an intuitive and convenient way of measuring s-box nonlinearity (see also [20]). However, s-boxes created by their procedure suffer from several limitations:

(a) by their own measures of nonlinearity these s-boxes are weak since an inverse s-box S^{-1} has only one nonlinear component; other $n - 1$ components are linear;
(b) there is no proof that property (3) holds when s-boxes are constructed with this method;
(c) these s-boxes fail property (4); in fact, it is easy to show that *every* pair of output bits has a correlation of either $+1$ or -1.

In their work, Pieprzyk and Finkelstein concentrated only on the nonlinearity property of s-boxes. Here we are interested in simultaneously achieving all four properties outlined above with a single design process. We have been successful in meeting this goal. The main conceptual difference between our method and that of Pieprzyk and Finkelstein is that the n Boolean functions which make up the s-box output bits are independently chosen, rather than simple variations of each other. If these functions are chosen carefully (i.e., chosen to fulfill certain requirements), then the resulting s-box can be proven to possess our four evaluation criteria.

Briefly, some of the relevant concepts and definitions are as follows. A Boolean function of n inputs, $f(x_1, x_2, \ldots, x_n)$, can be regarded as a binary vector $\mathbf{f}$ of length 2^n, where $\mathbf{f}$ is the rightmost column of the truth table describing this function. The Hamming weight of this vector and the Hamming distance between two such vectors are defined in the usual sense. Linear vectors are defined to be $(a_1\mathbf{x}_1 + a_2\mathbf{x}_2 + \cdots + a_n\mathbf{x}_n + a_{n+1}\mathbf{1})$ where the sum is taken modulo 2, a_i is in $\{0, 1\}$, $\mathbf{1}$ is the all-one vector, and the $\mathbf{x}_i$ are the truth table vectors of length 2^n for the n inputs of f (i.e., $\mathbf{x}_1 = 00\cdots011\cdots1$, $\mathbf{x}_n = 0101\cdots01$). Viewed in this way, for any given n, there are 2^{n+1} linear vectors of length 2^n; we note that this set of linear vectors comprise exactly the set of codewords for a binary first-order Reed-Muller code with parameters $(2^n, n + 1, 2^{n-1})$—see [14], for example. The nonlinearity of a vector is defined as the minimum Hamming distance between that vector and the 2^{n+1} linear vectors. S-box design then consists of choosing n nonlinear Boolean

functions which possess certain properties (for example, the spectral property described in [10]) and setting these as the output bits. The next section gives details regarding the necessary properties of the Boolean functions and proofs mapping these properties to our overall design criteria.

6. Selection of Boolean Functions

In this section we describe the selection process for the Boolean functions which will serve as the n output bits of the s-box. We find that there are no conflicting requirements on these functions for our four evaluation criteria; consequently, these can all be met simultaneously.

The arrangement of Boolean functions into output bits is as follows. Let the n functions be $f_1, f_2, \ldots, f_n$, where each function f_i corresponds to a binary vector $\mathbf{f}_i$ of length 2^n. Then the s-box $S = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n]$ is a $2^n \times n$ bit matrix with the $\mathbf{f}_i$ as column vectors. Any given input vector $x = x_1, x_2, \ldots, x_n$ maps to an output vector $y = y_1, y_2, \ldots, y_n$ by the assignment $y_i = f_i(x_1, x_2, \ldots, x_n)$. As a simple example, let $n = 3$ and $\mathbf{f}_1 = (00001111)$, $\mathbf{f}_2 = (00110011)$, $\mathbf{f}_3 = (01010101)$. Then

$$S = S^{-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}^t.$$

In this case, the s-box is a nonpermuted mapping of the integers 0–7 onto themselves. Our purpose is to find a set of functions $f_1, \ldots, f_n$ such that the corresponding s-box is bijective and nonlinear and fulfills the SAC and the (output) bit independence criterion (BIC). Note that it may be desirable in some applications to have the inverse s-box S^{-1} possess these four properties as well.

6.1. *Bijection*

We require that an $n \times n$ bit s-box be a bijection; that is, that every possible input vector x maps to a unique output vector y. The most obvious example of such an s-box is the nonpermuting s-box (we could also think of it as the "identity" s-box) S_I, where $y = x$:

$$S_I = \begin{bmatrix} \mathbf{f}_1 & \mathbf{f}_2 & \cdots & \mathbf{f}_{n-1} & \mathbf{f}_n \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & 1 & \cdots & 1 & 0 \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix}.$$

Pieprzyk and Finkelstein give two properties of such an s-box: the Hamming weight of any column of S is 2^{n-1} (i.e., $\mathrm{wt}(\mathbf{f}_i) = 2^{n-1}$) and the Hamming distance between any two columns of S is also 2^{n-1} (i.e., $\mathrm{d}(\mathbf{f}_i, \mathbf{f}_j) = 2^{n-1}$, for $i \neq j$). These two

properties are not sufficient to prove bijection, however; a counterexample is

$$S = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}^t,$$

$$\mathrm{wt}(\mathbf{f}_i) = 8, \qquad i = 1, \ldots, 4,$$

$$\mathrm{d}(\mathbf{f}_i, \mathbf{f}_j) = 8, \qquad i \neq j.$$

Here we prove that a necessary and sufficient condition for S to be bijective is that any linear combination of the columns of S has Hamming weight 2^{n-1} (i.e., $\mathrm{wt}(a_1\mathbf{f}_1 \oplus a_2\mathbf{f}_2 \oplus \cdots \oplus a_n\mathbf{f}_n) = 2^{n-1}$, where the $a_i \in \{0, 1\}$ and the a_i are not all simultaneously zero). We begin by showing that S_I has this property (the proof is by induction).

The notation is as follows. Let $(a_1\mathbf{f}_1 \oplus a_2\mathbf{f}_2 \oplus \cdots \oplus a_n\mathbf{f}_n) = \sum_{i=1}^{n} a_i\mathbf{f}_i$, (i.e., we take $\sum$ to be the sum modulo 2 for convenience). Furthermore, let $\mathrm{top}(V)$ be the top half of a vector of any (even) length and let $\mathrm{bot}(V)$ be the bottom half of the same vector V.

Proof

(I) For $n = 2$,

$$S_I = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}^t$$

and $\mathrm{wt}(\mathbf{f}_1) = \mathrm{wt}(\mathbf{f}_2) = \mathrm{wt}(\mathbf{f}_1 \oplus \mathbf{f}_2) = 2^{n-1} = 2$.

(II) Assume that for $n = k$, $\mathrm{wt}(\sum_{i=1}^{k} a_i\mathbf{f}_i) = 2^{k-1} = 2^{n-1}$.

(III) Let $n = k + 1$. Relabel the columns of S_I such that the leftmost column is labeled $\mathbf{f}_{k+1}$ and the remaining columns are labeled $\mathbf{f}_1 \cdots \mathbf{f}_k$ (left to right). Now, $\mathrm{wt}(\sum_{i=1}^{k} a_i \times \mathrm{top}(\mathbf{f}_i)) = \mathrm{wt}(\sum_{i=1}^{k} a_i \times \mathrm{bot}(\mathbf{f}_i)) = 2^{k-1}$ (from Step II). This implies that $\mathrm{wt}(\sum_{i=1}^{k+1} a_i \times \mathrm{top}(\mathbf{f}_i)) = 2^{k-1}$ since $\mathrm{top}(\mathbf{f}_{k+1})$ consists of all zeros and $\mathrm{wt}(\sum_{i=1}^{k+1} a_i \times \mathrm{bot}(\mathbf{f}_i)) = 2^{k-1}$ since $\mathrm{bot}(\mathbf{f}_{k+1})$ consists of all ones. Therefore $\mathrm{wt}(\sum_{i=1}^{k+1} a_i\mathbf{f}_i) = 2^{k-1} + 2^{k-1} = 2^k = 2^{n-1}$

We now prove that any $2^n \times n$ matrix S with the property that $\mathrm{wt}(\sum_{i=1}^{n} a_i\mathbf{f}_i) = 2^{n-1}$ is a bijective mapping. The idea of the proof is to permute the rows of S to S_I. The proof is given as a recursive procedure which accepts as parameters a counter m (indicating the depth of recursion) and column vectors $V_m, V_{m+1}, \ldots, V_n$ (the length of V_i will vary according to the depth of recursion).

If S_i is the ith column of S, then we note that $\mathrm{wt}(S_1) = 2^{n-1}$; we can therefore rearrange the rows of S so that S_1 has 2^{n-1} zeros followed by 2^{n-1} ones. We then call procedure **Permute** $(1, S_1, S_2, \ldots, S_n)$ to reorder the rows of S into S_I. Note that it is always assumed that the binary coefficients a_i are not all simultaneously zero because this would imply that $\mathrm{wt}(\mathbf{0}) = 2^{n-1}$.

Logic used in procedure **Permute**. Note that

$$\mathrm{wt}\left(\sum_{i=m}^{n} a_i V_i\right) = 2^{n-m} = \mathrm{wt}\left(\sum_{i=m}^{n} a_i \cdot \mathrm{top}(V_i)\right) + \mathrm{wt}\left(\sum_{i=m}^{n} a_i \cdot \mathrm{bot}(V_i)\right),$$

$$a_m = 0 \quad \Rightarrow \quad \mathrm{wt}\left(\sum_{i=m+1}^{n} a_i V_i\right) = \mathrm{wt}\left(\sum_{i=m+1}^{n} a_i \cdot \mathrm{top}(V_i)\right) + \mathrm{wt}\left(\sum_{i=m+1}^{n} a_i \cdot \mathrm{bot}(V_i)\right),$$

i.e.,

$$2^{n-m} = \alpha + \beta \qquad \text{(for convenience)}, \tag{1}$$

$$a_m = 1 \quad \Rightarrow \quad \mathrm{wt}\left(V_m \oplus \sum_{i=m+1}^{n} a_i V_i\right) = \mathrm{wt}\left(\mathrm{top}(V_m) \oplus \sum_{i=m+1}^{n} a_i \cdot \mathrm{top}(V_i)\right) + \mathrm{wt}\left(\mathrm{bot}(V_m) \oplus \sum_{i=m+1}^{n} a_i \cdot \mathrm{bot}(V_i)\right),$$

i.e.,

$$2^{n-m} = \alpha + [2^{n-m} - \beta] \tag{2}$$

(since $\mathrm{top}(V_m) = \mathbf{0}$ and $\mathrm{bot}(V_m) = \mathbf{1}$).

$$(1) \text{ and } (2) \quad \Rightarrow \quad \beta = 2^{n-m} - \beta \quad \Rightarrow \quad \beta = 2^{n-(m+1)} = \alpha,$$

i.e.,

$$\mathrm{wt}\left(\sum_{i=m+1}^{n} a_i \cdot \mathrm{top}(V_i)\right) = \mathrm{wt}\left(\sum_{i=m+1}^{n} a_i \cdot \mathrm{bot}(V_i)\right) = 2^{n-(m+1)}.$$

Therefore, in particular,

$$\mathrm{wt}(\mathrm{top}(V_{m+1})) = \mathrm{wt}(\mathrm{bot}(V_{m+1})) = 2^{n-(m+1)}.$$

Permute (m, V_m, $V_{m+1}, \ldots, V_n$)
Begin
Rearrange the rows of S so that V_{m+1} has $2^{n-(m+1)}$ zeros followed by $2^{n-(m+1)}$ ones (this is $\mathrm{top}(V_{m+1})$), followed by $2^{n-(m+1)}$ zeros followed by $2^{n-(m+1)}$ ones (this is $\mathrm{bot}(V_{m+1})$); this can be done because of the pevious logic.
If $m + 1 = n$ **Return**. Otherwise,
Permute ($m + 1$, **top**(V_{m+1}), **top**(V_{m+2}), …, **top**(V_n)).
Permute ($m + 1$, **bot**(V_{m+1}), **bot**(V_{m+2}), …, **bot**(V_n)).
End. □

We conclude then that if we choose our Boolean functions $f_1, \ldots, f_n$ such that $\mathrm{wt}(\sum_{i=1}^{n} a_i \mathbf{f}_i) = 2^{n-1}$, where $a_i \in \{0, 1\}$ and the a_i are not all simultaneously zero, the corresponding s-box is guaranteed to be bijective.

6.2. *Nonlinearity*

There are two ways in which the output of an s-box can be considered to be a linear function of the input (note that here we are including affinity in the term linearity): at the bit level (i.e., modulo 2) and at the integer level (i.e., modulo 2^n). Let x be the n-bit input and let y be the n-bit output of an $n \times n$ bit s-box. At the bit level,

$$y = Ax + b \quad \Rightarrow \quad \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$\Rightarrow \quad y_i = \left(\sum_{j=1}^{n} a_{ij} x_j + b_i \right) \mod 2.$$

In terms of the s-box with columns $\mathbf{f}_1, \ldots, \mathbf{f}_n$, the equation above is equivalent to $\mathbf{f}_i = (\sum_{j=1}^{n} \alpha_j \mathbf{x}_j) + \alpha_{n+1} \mathbf{1}$, where $\alpha_j \in \{0, 1\}$ and the $\mathbf{x}_j$ are the standard truth table column vectors (i.e., $\mathbf{f}_i$ is linear). Therefore, for the output to be a linear function of the input at the bit level, it follows that every component Boolean function of the s-box must be linear. We can guarantee nonlinearity, then, simply by forcing at least one of these functions to be nonlinear.

At the integer level, x, $y \in \{0, 2^n - 1\}$ and linearity requires that $y = (ax + b) \bmod 2^n$. As an example, let $y = (3x + 7) \bmod 16$ for a 4×4 s-box. Then

$$S = [7 \quad 10 \quad 13 \quad 0 \quad 3 \quad 6 \quad 9 \quad 12 \quad 15 \quad 2 \quad 5 \quad 8 \quad 11 \quad 14 \quad 1 \quad 4]^t$$

$$= \begin{bmatrix} \mathbf{f}_1: & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ \mathbf{f}_2: & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ \mathbf{f}_3: & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ \mathbf{f}_4: & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}^t$$

so that

$$\mathbf{f}_4 = x_4 \oplus 1,$$

$$\mathbf{f}_3 = x_3 \oplus 1,$$

$$\mathbf{f}_2 = x_2 \oplus x_4 \oplus 1,$$

$$\mathbf{f}_1 = [\bar{x}_2(x_3 \oplus x_4) + x_2 x_3] \oplus x_1.$$

Note that while $\mathbf{f}_2$, $\mathbf{f}_3$, and $\mathbf{f}_4$ are linear, $\mathbf{f}_1$ is a nonlinear Boolean function. This means that to avoid linearity at the bit level it is not sufficient to make any *arbitrary* column of S nonlinear, because this may still allow linearity at the integer level. To see a necessary condition for integer linearity we can look at the least significant bit of y ($\mathrm{lsb}(y)$):

$$\mathrm{lsb}(y) = \mathrm{lsb}(a) \cdot \mathrm{lsb}(x) + \mathrm{lsb}(b)$$

$$= \begin{cases} 0, & \text{if } \mathrm{lsb}(a) = 0 \text{ and } \mathrm{lsb}(b) = 0, \\ 1, & \text{if } \mathrm{lsb}(a) = 0 \text{ and } \mathrm{lsb}(b) = 1, \\ \mathrm{lsb}(x), & \text{if } \mathrm{lsb}(a) = 1 \text{ and } \mathrm{lsb}(b) = 0, \\ \mathrm{lsb}(x) \oplus 1, & \text{if } \mathrm{lsb}(a) = 1 \text{ and } \mathrm{lsb}(b) = 1. \end{cases}$$

Therefore, in terms of the s-box S, if y is an (integer) linear function of x, then $\mathbf{f}_n$ is a linear function. Forcing $\mathbf{f}_n$ to be nonlinear, then, guarantees that S is nonlinear at both the integer level and at the bit level.

Of course, we may extend the idea of nonlinearity at the bit level by requiring that *every* output bit be a (highly) nonlinear function of the n input bits (i.e., that S not even be *close* to linear [20]). This is the idea behind Pieprzyk and Finkelstein's definition of s-box nonlinearity: it is the minimum nonlinearity over all $2n$ Boolean functions in S and S^{-1}. Clearly, for this it is necessary (but, it turns out, not sufficient) that we choose $\mathbf{f}_1, \ldots, \mathbf{f}_n$ to be nonlinear.

An issue that arises at this point is that of *amount* of nonlinearity (at the bit level): how much nonlinearity is "enough" and are there limits to what can be achieved? Recall that the nonlinearity of a function is defined as the minimum Hamming distance between that function and every linear function. Pieprzyk and Finkelstein give an equation in [18] to calculate the maximum achievable nonlinearity for a function of n input bits (see [17] for details):

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

It can be shown that N_f above is equal to $2^{n-1} - 2^{n/2-1}$ when n is even; functions of this nonlinearity are known in combinatorial theory as *bent* functions (see, for example, [19], [15], and [1]). Bent functions have ideal nonlinearity, but are not 0–1 balanced. If $\mathbf{f}$ is constrained to have Hamming weight equal to 2^{n-1} (see Section 6.1) Pieprzyk and Finkelstein [18] claim that N_f is modified to

$$N_f = \begin{cases} \displaystyle\sum_{i=(1/2)(n-3)}^{n-3} 2^{i+1} & \text{for } n = 3, 5, 7, \ldots, \\ \displaystyle\sum_{i=(1/2)(n-4)}^{n-4} 2^{i+2} & \text{for } n = 4, 6, 8, \ldots. \end{cases}$$

We refer to this modified N_f as "maximum" nonlinearity since it is the highest nonlinearity achievable with 0–1 balanced functions. It turns out that there are many functions of this maximum nonlinearity (many more than n), so that we can easily construct s-boxes for which all n functions are maximally nonlinear.

We can therefore state that if the n Boolean functions of an s-box S are nonlinear, then S is guaranteed to be nonlinear at the bit level and at the integer level. Furthermore, if these functions are maximally nonlinear, then Rueppel's "closest linear approximation" to the s-box [20] will be a bad approximation. We have also noticed that if these n functions are not simple variations of each other, as in [18], the inverse S^{-1} is also nonlinear at the bit and integer level and is composed of highly nonlinear functions, although at present we have no proof that this is always true.

6.3. *Strict Avalanche Criterion*

Intuitively, it seems clear that choosing functions of Hamming weight 2^{n-1} for all n output bits would lead to s-boxes with good avalanche. This is because within

every vector $\mathbf{f}_i$, inverting input bit x_b corresponds to a change in location from position f_{i_j} to f_{i_k}, for some j and k. If $\mathbf{f}_i$ contains an equal number of zeros and ones, then over all possible inputs x, inverting bit x_b should cause y_i to be inverted 50% of the time. If this characteristic is shared by all vectors $\mathbf{f}_1, \ldots, \mathbf{f}_n$, then inverting x_b should cause half of the $y_1, \ldots, y_n$ to be inverted, on average.

To *prove* that an s-box will fulfill the SAC, however, we must go a step further. We do this by incorporating the work of Forré [10] into this context and use SAC-fulfilling Boolean functions. Let $\hat{f}(x) = (-1)^{f(x)}$ and let $\hat{F}(w)$ denote the Walsh–Hadamard transform of $\hat{f}(x)$. It has been proven that a necessary and sufficient condition on the Walsh-Hadamard transform of a Boolean function $f(x)$ for this function to fulfill the SAC is that

$$\sum_{w \in Z_2^n} (-1)^{w_i} \cdot \hat{F}^2(w) = 0 \qquad \text{for} \quad i = 1, \ldots, n,$$

where Z_2^n is the n-dimensional vector space over GF(2). Furthermore,

$$\hat{F}^2(w) = \hat{F}^2(\bar{w}) \qquad \text{for all} \quad w \in Z_2^n, \qquad \text{where} \quad \bar{w} = w \oplus (111 \ldots 1)$$

is shown to be a sufficient condition for $f(x)$ to be an SAC-fulfilling function (see [10] for further details). The terminology "50% dependence" introduced by Forré is very suggestive:

A function $f: Z_2^n \to \{0, 1\}$ is said to be 50% dependent on its ith input bit x_i if and only if any two vectors u_1 and u_2 that differ only in bit i are mapped

- onto two different values with probability 1/2,
- onto the same value with probability 1/2,

and

a function $f: Z_2^n \to \{0, 1\}$ fulfills the SAC if and only if f is 50% dependent on each of its n input bits.

In our context (in particular, if the s-box S is bijective and has independent output bits) it is clear that if f_i fulfills the SAC for $i = 1, \ldots, n$, then S is guaranteed to possess the SAC.

Note that the Walsh–Hadamard transform, like several other transform algorithms (the well-known Fourier transform, for example), has a "fast" version which reduces the running time from $O(m^2)$ to $O(m \log_2 m)$, where m is the size of the input vector (2^n bits, in our case). For modest m, checking the Walsh transform for a given function f is computationally intensive, but is not prohibitive on a reasonably powerful machine (a SUN workstation, for example). Since we are dealing with binary vectors, however, this computation can be reduced significantly simply by examining the definition of the SAC. The SAC for a Boolean function of n variables states that inverting a single input variable must cause the output to change one half the time, for all input variables. Consider the truth table for a Boolean function $f(x_1, \ldots, x_n)$. From the construction of the truth table it is clear that inverting input x_1 will cause a change in output from location i in the top half of $\mathbf{f}$ to location i in the bottom half of $\mathbf{f}$ (or vice versa). The modulo 2 sum of these two locations, then, will be one if and only if the output has changed (it will be zero otherwise). Therefore,

if the weight of the bitwise sum of these halves of $\mathbf{f}$ is 2^{n-2}, then f is 50% dependent on x_1. A similar check can be made on the other input variables. For example, let f be a function of four variables, so that $\mathbf{f}$ is 16 bits long. In a programming language such as "C," we can treat $\mathbf{f}$ as an unsigned integer and perform the *rightshift* operation ($\gg$), bitwise *exclusive-or* (^) and bitwise *and* (&) on it. If

(1) $\{\text{wt}((\mathbf{f} \,\&\, 00FF) \,\hat{}\, ((\mathbf{f} \gg 8) \,\&\, 00FF)) = 4\}$ and
(2) $\{\text{wt}((\mathbf{f} \,\&\, 0F0F) \,\hat{}\, ((\mathbf{f} \gg 4) \,\&\, 0F0F)) = 4\}$ and
(3) $\{\text{wt}((\mathbf{f} \,\&\, 3333) \,\hat{}\, ((\mathbf{f} \gg 2) \,\&\, 3333)) = 4\}$ and
(4) $\{\text{wt}((\mathbf{f} \,\&\, 5555) \,\hat{}\, ((\mathbf{f} \gg 1) \,\&\, 5555)) = 4\}$,

then f is an SAC-fulfilling function. Verifying that a function of length 2^n satisfies the SAC thus requires only $4n$ logical operations and n computations of Hamming weight and can be done extremely efficiently. Our notation is as follows: let $\mathbf{f}^{(i)}$ be the result of the ith set of logical operations on $\mathbf{f}$ (as defined above) for $i = 1, \ldots, n$. Then $\text{wt}(\mathbf{f}^{(i)}) = 2^{n-2}$ for all i if and only if f is an SAC-fulfilling function.

We therefore require that the n Boolean functions of an s-box each be SAC-fulfilling functions.

6.4. (*Output*) *Bit Independence Criterion*

Here we show that if the (mod 2) sum of any two Boolean functions in an s-box is a linear function (i.e., $\mathbf{f}_i \oplus \mathbf{f}_j = (\sum_{d=1}^{n} a_d \mathbf{x}_d) \oplus a_{n+1}\mathbf{1}$), then for any input bit x_k, flipping x_k will show a correlation of ± 1 between output bits y_i and y_j.

Proof. A general Boolean function f can be written as having two components N and L; we write $f = N(x_1, \ldots, x_n) \oplus L(x_1, \ldots, x_n)$, where N is a nonlinear function of the inputs (N may or may not be present) and L is a linear function of the inputs (L may or may not be the zero vector). Therefore $f_i = N_i \oplus L_i$ and $f_j = N_j \oplus L_j$.

But

$$\mathbf{f}_i \oplus \mathbf{f}_j = \left(\sum_{d=1}^{n} a_d \mathbf{x}_d\right) \oplus a_{n+1}\mathbf{1} \quad \Rightarrow \quad \begin{cases} N_i = N_j = N & \text{and} \\ L_i \oplus L_j = \left(\sum_{d=1}^{n} a_d \mathbf{x}_d\right) \oplus a_{n+1}\mathbf{1} \end{cases}$$

so that $f_i = N \oplus L_i$ and $f_j = N \oplus L_j$.

Now (without loss of generality),

$$a_k = 1 \quad \Rightarrow \quad \begin{cases} f_i = N \oplus L_i' \oplus 1\mathbf{x}_k, \\ f_j = N \oplus L_j, \end{cases}$$

where $L_i' \oplus L_j = (\sum_{d=1, d\neq k}^{n} a_d \mathbf{x}_d) \oplus a_{n+1}\mathbf{1}$. Therefore flipping x_k shows a correlation of -1 between f_i and f_j (i.e., between output bits y_i and y_j). Furthermore,

$$a_k = 0 \quad \Rightarrow \quad \begin{cases} f_i = N \oplus L_i' \oplus 0\mathbf{x}_k = N \oplus L_i', \\ f_j = N \oplus L_j, \end{cases}$$

where $L_i' \oplus L_j = (\sum_{d=1, d\neq k}^{n} a_d \mathbf{x}_d) \oplus a_{n+1}\mathbf{1}$. Therefore flipping x_k shows a correlation of $+1$ between $\mathbf{f}_i$ and $\mathbf{f}_j$ (i.e., between output bits y_i and y_j). □

Therefore, if $\mathbf{f}_i \oplus \mathbf{f}_j$ is a linear function, then flipping input x_k either causes y_i and y_j always to flip together or never to flip together. Put another way, inverting input bit x_k either causes the sum $(y_i \oplus y_j)$ always to remain the same (i.e., over all possible inputs) or always to be inverted. This undesirable situation can be avoided simply by ensuring that $\mathbf{f}_i \oplus \mathbf{f}_j = \mathbf{f}$, where $\mathbf{f}$ is a (maximally or near-maximally) nonlinear function of all n input bits. As with the strict avalanche section above, however, we can go one step further and guarantee that output bits y_i and y_j have a correlation as low as possible by requiring that $\mathbf{f}$ be close to an SAC-fulfilling function (i.e., that $\text{wt}(\mathbf{f}^{(i)}) \approx 2^{n-2}$ for all i). This will ensure that $\mathbf{f}_i \oplus \mathbf{f}_j$ is $\sim 50\%$ dependent on input x_k for all k (i.e., that y_i and y_j will flip independently). It turns out that forcing $\mathbf{f}$ to *be* an SAC-fulfilling function overconstrains $\mathbf{f}$ and conflicts with some of our other criteria. Therefore, for example, with 16-bit functions we can require that $\text{wt}(\mathbf{f}^{(i)}) \in \{3, 4, 5\}$ rather than that $\text{wt}(\mathbf{f}^{(i)}) = 4$ for all i.

We say that if $\mathbf{f}$ is maximally (or near-maximally) nonlinear and $\mathbf{f}$ is close to an SAC-fulfilling function, then $\mathbf{f}$ is a BIC-fulfilling function.

7. Results

We now have a procedure for generating what appear to be cryptographically good s-boxes. It consists of searching through the Boolean functions until we find n functions with the properties outlined in the previous section and setting these as the output bits.

For n input bits there are 2^{2^n} Boolean functions. Of these, all but 2^{n+1} are nonlinear. However, fewer of these are maximally (or near-maximally) nonlinear and only $\binom{2^n}{2^{n-1}}$ are of weight 2^{n-1}. Thus the search space is considerably reduced by these relatively simple checks before it becomes necessary to do the more computationally intensive checks such as testing for SAC and BIC.

The algorithm for generating s-boxes is as follows:

begin
  set $j = 1$
  set $cand = 1$
  while $j \leq n$ **do**
    for each candidate Boolean function f_{cand}
      begin
        set $f_j = f_{\text{cand}}$
        set $cand = cand + 1$

{Biject} **if** $\text{wt}\left(\sum_{i=1}^{j} a_i f_i\right) = 2^{n-1}$ **and**

{Nonlin} **if** $N_{f_j} = \left\{ \begin{array}{ll} \sum_{i=(1/2)(n-3)}^{n-3} 2^{i+1} & \text{for } n = 3, 5, 7, \ldots, \\ \sum_{i=(1/2)(n-4)}^{n-4} 2^{i+2} & \text{for } n = 4, 6, 8, \ldots \end{array} \right\}$ **and**

{SAC} **if** $\text{wt}(f_j^{(k)}) = 2^{n-2}$ for $k = 1, \ldots, n$ **and**
{BIC} **if** (for $i = 1, \ldots, j-1$) $N_{f_j \oplus f_i}$ is as above, and $\text{wt}((f_j \oplus f_i)^{(k)}) \approx 2^{n-2}$ for $k = 1, \ldots, n$
then set $y_j = f_j$ and **set** $j = j + 1$
end
end.

It has been proven that following this algorithm will construct an s-box which possesses our four cryptographically desirable criteria. Furthermore, a small modification can be made which will additionally guarantee that the s-boxes constructed will perform well with respect to multiple input bit complementation and will be free of linear structures [5], [7]. However, an interesting area for further research is to investigate the s-boxes generated by this algorithm for other features not included in the design procedure, such as cross-correlations of the avalanche variables or statistical dependencies between three or more output bits.

We have used the procedure above to generate approximately 60 4 × 4 s-boxes (this is not an exhaustive list) in a few seconds of CPU time on a SUN workstation and have found no deficiencies in any of the resulting s-boxes. Furthermore, most of the s-boxes generated have inverses which also satisfy our four properties, making them potential candidates for general S–P network cryptosystem design. We have also found that if some criteria are relaxed slightly, it is easy to generate hundreds or even thousands of "fairly good' s-boxes (note that all of the DES s-boxes fall into this category). Some example s-boxes generated by this procedure are

$$S_1 = [0\ f\ 3\ 1\ c\ d\ 4\ 8\ 9\ b\ 2\ 6\ a\ 5\ e\ 7]^t,$$
$$S_2 = [1\ e\ 7\ 9\ b\ a\ 3\ 0\ 8\ c\ d\ 4\ f\ 5\ 2\ 6]^t,$$
$$S_3 = [3\ c\ d\ e\ 1\ 5\ 9\ 0\ a\ 8\ 7\ b\ 2\ f\ 4\ 6]^t,$$
$$S_4 = [c\ d\ e\ 0\ 5\ 8\ 1\ 2\ 3\ 9\ 7\ 6\ f\ b\ 4\ a]^t,$$
$$S_5 = [e\ f\ 0\ 6\ a\ 9\ b\ 3\ 1\ 7\ 4\ d\ 8\ 5\ c\ 2]^t,$$
$$S_6 = [f\ 3\ c\ d\ 1\ 0\ 6\ 9\ a\ 8\ e\ 4\ b\ 5\ 2\ 7]^t.$$

8. Conclusion

We have succeeded in setting out a methodical procedure for s-box design which allows us to prove that any s-box so designed will possess our complete set of desirable properties. The method consists of performing a constrained search through a subset of the nonlinear Boolean functions and assigning n chosen functions as the s-box outputs. Our results indicate that even for relatively small dimensions (4 × 4 bits), many different s-boxes can be designed using this procedure.

This work aids our understanding of the s-box design process and allows us to create what we feel to be cryptographically good s-boxes quickly and easily; this, in turn, can lead to much reduced time and effort in the design of new private-key cryptosystems.

Acknowledgments

The authors are grateful to the anonymous referee(s) for insightful comments and suggestions.

References

[1] C. M. Adams and S. E. Tavares, A Note on the Generation and Counting of Bent Sequences, Tech. Rep. TR 89-07, Department of Electrical Engineering, Queen's University, July 1989. Also in *IEEE Transactions on Information Theory*, 36 (1990), 1170–1173.

[2] D. K. Branstad, J. Gait, and S. Katzke, Report of the Workshop on Cryptography in Support of Computer Security, Tech. Rep. NBSIR 77-1291, National Bureau of Standards, Sept. 1976.

[3] E. F. Brickell, J. H. Moore, and M. R. Purtill, Structure in the s-boxes of the DES (extended abstract), in *Advances in Cryptology: Proc. of CRYPTO '86*, Springer-Verlag, New York, 1987, pp. 3–8.

[4] J. M. Carroll and L. E. Robbins, Using binary derivatives to test an enhancement of DES, *Cryptologia*, 12 (1988), 193–208.

[5] D. Chaum and J.-H. Evertse, Cryptanalysis of DES with a reduced number of rounds, in *Advances in Cryptology: Proc. of CRYPTO '85*, Springer-Verlag, New York, 1986, pp. 192–211.

[6] B. den Boer, Cryptanalysis of F.E.A.L., in *Advances in Cryptology: Proc of EUROCRYPT '88*, Springer-Verlag, Berlin, 1989, pp. 167–173.

[7] J.-H. Evertse, Linear structures in block ciphers, in *Advances in Cryptology: Proc. of EUROCRYPT '87*, Springer-Verlag, Berlin, 1988, pp. 249–266.

[8] H. Feistel, Cryptography and computer privacy, *Scientific American*, 228 (1973), 15–23.

[9] H. Feistel, W. Notz, and J. L. Smith, Some cryptographic techniques for machine-to-machine data communications, *Proceedings of the IEEE*, 63 (1975), 1545–1554.

[10] R. Forré, The strict avalanche criterion: spectral properties of boolean functions and an extended definition, in *Advances in Cryptology: Proc. of CRYPTO '88*, Springer-Verlag, New York, 1989, pp. 450–468.

[11] W. Fumy, On the *F*-function of FEAL, in *Advances in Cryptology: Proc. of CRYPTO '87*, Springer-Verlag, New York, 1988, pp. 434–437.

[12] M. E. Hellman, R. Merkle, R. Schroeppel, L. Washington, W. Diffie, S. Pohlig, and P. Schweitzer, Results of an Initial Attempt to Cryptanalyze the NBS Data Encryption Standard, Tech. Rep. SEL 76-042, Information Systems Laboratory, Stanford University, Nov. 1976.

[13] J. B. Kam and G. I. Davida, Structured design of substitution–permutation encryption networks, *IEEE Transactions on Computers*, 28 (1979), 747–753.

[14] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam, 1977.

[15] W. Meier and O. Staffelbach, *Nonlinearity criteria for cryptographic functions*, in *Advances in Cryptology: Proc. of EUROCRYPT '89*, to appear.

[16] National Bureau of Standards (U.S.), *Data Encryption Standard (DES)*, Federal Information Processing Standards Publication 46, 1977.

[17] J. P. Pieprzyk, Nonlinear functions and their application to cryptography, *Archiwum Automatyki i Telemechaniki*, 3–4 (1985), 311–323.

[18] J. Pieprzyk and G. Finkelstein, Towards effective nonlinear cryptosystem design, *IEE Proceedings, Part E: Computers and Digital Techniques*, 135 (1988), 325–335.

[19] O. S. Rothaus, On "Bent" functions, *Journal of Combinatorial Theory*, 20(A) (1976), 300–305.

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

[21] R. Scott, Wide-open encryption design offers flexible implementations, *Cryptologia*, 9 (1985), 75–90.

[22] A. Shamir, On the security of DES, in *Advances in Cryptology: Proc. of CRYPTO '85*, Springer-Verlag, New York, 1986, pp. 280–281.

[23] C. E. Shannon, Communication theory of secrecy systems, *Bell Systems Technical Journal*, 28 (1949), 656–715.

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

[25] A. F. Webster and S. E. Tavares, On the design of s-boxes, in *Advances in Cryptology: Proc. of CRYPTO '85*, Springer-Verlag, New York, 1986, pp. 523–534.

[26] K. C. Zeng, J. H. Yang, and Z. T. Dai, Patterns of entropy drop of the key in an s-box of the DES, in *Advances in Cryptology: Proc. of CRYPTO '87*, Springer-Verlag, New York, 1988, pp. 438–444.