Practical S-Box Design

Serge Mister, Carlisle Adams

Nortel, P.O. Box 3511, Station C, Ottawa, Ontario, Canada, K1Y 4H7
cf744@freenet.carleton.ca, cadams@nortel.ca

1. Introduction

Much of the security of a block cipher based on the Feistel network [8, 9] depends on the properties of the substitution boxes (s-boxes) used in the round function. Although many desirable properties have been studied, relatively little work has been done to determine to what degree these properties are achievable in practice. This paper presents one effort to construct large, cryptographically secure s-boxes, contrasting theoretical and practical limitations, and highlighting areas for future research.

2. Background

An $n \times m$ s-box S is a mapping $S: \{0,1\}^n \rightarrow \{0,1\}^m$. S can be represented as 2^n m-bit numbers, denoted $r_0, \ldots, r_{2^n-1}$, in which case $S(x) = r_x$, $0 \le x < 2^n$ and the r_i are the rows of the s-box. Alternatively, $S(x) = \left[c_{m-1}(x)\ c_{m-2}\ (x) \ldots c_0(x)\right]$ where the c_i are fixed Boolean functions $c_i: \{0,1\}^n \rightarrow \{0,1\}\ \forall i$; these are the columns of the s-box. Finally, S can be represented by a $2^n \times m$ binary matrix M with the i, j entry being bit j of row i. All three representations will be used in this paper.

The linear combination of two functions $f, g: \{0,1\}^n \rightarrow \{0,1\}$ is defined to be

$$(f \oplus g)(x) = f(x) \oplus g(x)$$

where $\oplus$ denotes modulo 2 addition. Let V_n denote the set of functions mapping $\{0,1\}^n \rightarrow \{0,1\}$. Let L_n denote the set of linear functions mapping $\{0,1\}^n \rightarrow \{0,1\}$. Let A_n denote the set of affine functions mapping $\{0,1\}^n \rightarrow \{0,1\}$.

The following definitions will be useful in the discussion.

2.1 Walsh Transform

The Walsh transform of a function $f: \{0,1\}^n \rightarrow \{0,1\}$ is defined by

$$W(f)(w) = \sum_{x \in \{0,1\}^n} (-1)^{f(x)+w \cdot x}$$

where $w \cdot x = w_{n-1}x_{n-1} \oplus \cdots \oplus w_0 x_0$ (note that the transform is usually normalized by multiplying by $2^{-n/2}$).

The set of bent functions, denoted B_n with n even, is the set of functions $f : \{0,1\}^n \to \{0,1\}$ such that [14]

$$W(f)(w) = \pm 2^{n/2} \quad \forall w \in \{0,1\}^n .$$

2.2 Inverse Walsh Transform

The inverse Walsh transform of a function $F : \{0,1\}^n \to \mathbf{Z}$ is

$$W^{-1}(F)(x) = \frac{1}{2^n} \sum_{w \in \{0,1\}^n} F(w)(-1)^{w \cdot x} .$$

2.3 Nonlinearity

The nonlinearity of a function $f : \{0,1\}^n \to \{0,1\}$ is

$$\mathrm{nl}(f) = \min_{l \in A_n} \mathrm{wt}(f \oplus l)$$

where wt() denotes the Hamming weight of the function.

The nonlinearity of an s-box S is

$$\mathrm{nl}(S) = \min_{f \in C} \mathrm{nl}(f)$$

where C is the set of all nontrivial linear combinations of the columns of S.

2.4 XOR Table

Let $\alpha \in \{0,1\}^n \setminus \{\mathbf{0}\}$, $\beta \in \{0,1\}^m$. The XOR table entry of an s-box S corresponding to (α, β) is

$$\mathrm{XOR}(\alpha, \beta) = \#\{x \in \{0,1\}^n : S(x) \oplus S(x \oplus \alpha) = \beta\}$$

where # denotes the cardinality of the set. The XOR value of an s-box is the highest XOR table entry:

$$\mathrm{XOR}(S) = \max_{\alpha, \beta} \mathrm{XOR}(\alpha, \beta) .$$

2.5 Dynamic Distance

We define the dynamic distance of order j of a function $f : \{0,1\}^n \to \{0,1\}$ as follows:

$$\mathrm{DD}_j(f) = \max_{\substack{d \in \{0,1\}^n \\ 1 \le wt(d) \le j}} \frac{1}{2} \left| 2^{n-1} - \sum_{x=0}^{2^n - 1} f(x) \oplus f(x \oplus d) \right| .$$

It provides a measure from which can be defined other dynamic properties such as the strict avalanche criterion, bit independence criterion, and a precise "distance" from these properties.

2.6 Strict Avalanche Criterion (SAC)

A Boolean function $f:\{0,1\}^n \to \{0,1\}$ satisfies the SAC if $\mathrm{DD}_1(f) = 0$. The distance to SAC is defined by $\mathrm{DSAC}(f) = \mathrm{DD}_1(f)$. Similarly, $f:\{0,1\}^n \to \{0,1\}$ satisfies higher-order SAC (HOSAC) of order j if $\mathrm{DD}_j(f) = 0$ and the distance to higher-order SAC of order j is defined by $\mathrm{DHOSAC}_j(f) = \mathrm{DD}_j(f)$. Maximum order SAC (MOSAC) and the distance to maximum order SAC (DMOSAC) correspond to the case $j = n$. The concepts of SAC, higher-order SAC, and maximum order SAC coincide with those of [4, 15] (see also [13]). An s-box satisfies the (HO, MO)SAC if all of its columns satisfy (HO,MO)SAC.

2.7 Bit Independence Criterion (BIC)

For an s-box S, the distance to higher order BIC is defined by

$$\mathrm{DHOBIC}_{i,j}(S) = \max_{\substack{c\in\{0,1\}^m \\ 1<wt(c)\le i}} \mathrm{DD}_j(Mc)$$

where M is the binary matrix corresponding to S and the matrix multiplication is done using modulo 2 addition. S satisfies BIC[1] [15] if $\mathrm{DHOBIC}_{2,1}(S) = 0$ and satisfies $\mathrm{HOBIC}_{i,j}$ if $\mathrm{DHOBIC}_{i,j}(S) = 0$ (note that HOBIC can be defined in terms of varying i, or varying j, or both). Distances to BIC and HOBIC are given by $\mathrm{DHOBIC}_{2,1}(S)$ and $\mathrm{DHOBIC}_{i,j}(S)$ respectively. Maximum order BIC (MOBIC) and the distance to MOBIC (DMOBIC) correspond to HOBIC and DHOBIC with $i = m,\ j = n$.

2.8 Ideal S-Box Properties

The following are properties which we feel that an ideal sbox would possess:

I1. All linear combinations of s-box columns are bent.
I2. All entries in the s-box XOR table are 0 or 2.
I3. The s-box satisfies MOSAC.
I4. The s-box satisfies MOBIC.
I5. The set of weights of rows has a binomial distribution with mean $m/2$.
I6. The set of weights of all pairs of rows has a binomial distribution with mean $m/2$.

[1] The (output) Bit Independence Criterion (BIC) states that s-box output bits j and k should change independently when any single input bit i is inverted, for all i, j, and k (note that for a given i, j, and k the independence is computed over the set of all pairs of input vectors which differ only in bit i).

I7. The columns each have Hamming weight 2^{n-1}.

Property I1 will aid in protection against linear cryptanalysis (see [5]), and I2 against differential cryptanalysis (see [6]). Properties I1, I5, and I7 help to ensure a good static characteristic, and properties I2, I3, I4, and I6 help to ensure a good dynamic characteristic. Not all of these properties can be achieved simultaneously.

3. General S-Box Construction Methods

We observe the following properties of s-boxes:

3.1 Property S1

The nonlinearity distribution of an s-box is not affected when affine functions are added to columns of the s-box.

Proof:

This follows directly from the definition of nonlinearity.

3.2 Property S2

$\max_{\substack{\alpha\in\{0,1\}^n\setminus\{\mathbf{0}\}\\ \beta\in\{0,1\}^m}} \mathrm{XOR}(\alpha,\beta)$ for an s-box S is not affected when affine functions are added to the columns of S.

Proof:

Let $b(x)$ be a column of S, $l(x)$ be a linear function, $\delta \in \{0,1\}$, and $\gamma(x) = l(x) + \delta$ be an affine function. Let $h(x) = b(x) \oplus \gamma(x)$ be the s-box column modified by adding an affine function.

For a given input XOR α, the output XOR at the bit position corresponding to column $b(x)$ for the modified s-box is:

$$\begin{aligned} h(x) \oplus h(x \oplus \alpha) &= b(x) \oplus \gamma(x) \oplus b(x \oplus \alpha) \oplus \gamma(x \oplus \alpha) \\ &= b(x) \oplus b(x \oplus \alpha) \oplus \gamma(x) \oplus \gamma(x) \oplus l(\alpha) \quad (\text{since } \gamma(x) \text{ is affine}) \\ &= b(x) \oplus b(x \oplus \alpha) \oplus l(\alpha) \ . \end{aligned}$$

Thus for a given input XOR, the output XORs with the modified column are the output XORs of the original s-box, except that each is XORed with the constant $l(\alpha)$ at the bit position corresponding to $b(x)$. Therefore, the number of output XORs β which correspond to a given input XOR α is unchanged.

3.3 Property S3

Dynamic distance is unaffected by the addition of affine functions.

Proof:

Let $f:\{0,1\}^n \to \{0,1\}$ be a Boolean function. Let $\gamma(x) = l(x) + \delta$ be an affine function. Let $h(x) = f(x) \oplus \gamma(x)$.

For a fixed input change c, let $B = \frac{1}{2}\left|2^{n-1} - \sum_{x=0}^{2^n-1} f(x) \oplus f(x \oplus c)\right|$ be the dynamic distance of f corresponding to that input change and T be the summation term. The dynamic distance of the modified function is given by

$$\begin{aligned}
\frac{1}{2}\left|2^{n-1} - \sum_{x=0}^{2^n-1} h(x) \oplus h(x \oplus c)\right| &= \frac{1}{2}\left|2^{n-1} - \sum_{x=0}^{2^n-1} f(x) \oplus \gamma(x) \oplus f(x \oplus c) \oplus \gamma(x \oplus c)\right| \\
&= \frac{1}{2}\left|2^{n-1} - \sum_{x=0}^{2^n-1} f(x) \oplus \gamma(x) \oplus f(x \oplus c) \oplus \gamma(x) \oplus l(c)\right| \\
&= \frac{1}{2}\left|2^{n-1} - \sum_{x=0}^{2^n-1} f(x) \oplus f(x \oplus c) \oplus l(c)\right| \\
&= \frac{1}{2}\left|2^{n-1} - T\right| \\
&= B
\end{aligned}$$

where the second to last equality follows because $l(c)$ is a constant with respect to the summation. If $l(c) = 0$, the result is immediate. If $l(c) = 1$,

$$\begin{aligned}
\left|2^{n-1} - \sum_{x=0}^{2^n-1} f(x) \oplus f(x \oplus c) \oplus l(c)\right| &= \left|2^{n-1} - \sum_{x=0}^{2^n-1} f(x) \oplus f(x \oplus c) \oplus 1\right| \\
&= \left|2^{n-1} - \left(2^n - \sum_{x=0}^{2^n-1} f(x) \oplus f(x \oplus c)\right)\right| \\
&= \left|-2^{n-1} + \sum_{x=0}^{2^n-1} f(x) \oplus f(x \oplus c)\right| \\
&= \left|2^{n-1} - T\right|.
\end{aligned}$$

3.4 Property S4

The distances to (HO, MO)SAC and to (HO, MO)BIC for an s-box are not affected when affine functions are added to its columns.

Proof:

This follows directly from Property S3.

Properties S1-S4 show that an s-box with high nonlinearity, low distance to BIC, and a good XOR table can be modified by adding affine functions to the columns without disturbing these properties.

We will now focus on the case $n = 8$, $m = 32$, the dimensions typically used in CAST algorithms [1, 2]. Property I1 cannot be achieved for $2m > n$ [12]. I1 is then replaced with I1': the s-box has the highest possible nonlinearity. Property I2 can be achieved without much difficulty because m is sufficiently greater than n; ideal XOR tables for the s-box dimensions given have $2^{n-1} \cdot (2^n - 1) = 32\,640$ entries containing 2 and all other entries 0. Property I3 is guaranteed if bent functions are chosen for the c_i. Property I4 cannot be achieved for $2m > n$ [12]. We have reduced I4 to I4': the s-box minimizes $\mathrm{DHOBIC}_{32,1}$.

This leads us to construct s-boxes using bent functions as columns (Property I3), and emphasizing only nonlinearity and $\mathrm{HOBIC}_{32,1}$ (properties I1' and I4'). Once complete, the s-box can be inspected to see that it satisfies property I2, and can be modified by adding affine functions to columns to approximately obtain properties I5 and I6. Note that with $n = 8$, the bent functions have Hamming weight 120 or 136, which is close to the weight 128 required by property I7. These constructed s-boxes, therefore, represent what appears to be a good approximation to our definition of ideal s-boxes of these dimensions.

3.5 S-Box Construction Algorithm

The following construction algorithm (Alg. 1) was used:

1. Set $ncols = 0$.
2. Load a bent function into column *ncols* of the s-box.
3. Test the nonlinearity and DD_1 of all combinations of columns 0-*ncols* that involve column *ncols*.

4. If the minimum nonlinearity observed is greater than or equal to the minimum desired nonlinearity and the highest dynamic distance observed is at most the maximum allowable $\mathrm{DHOBIC}_{32,1}$, increment *ncols*.

5. If $ncols < 32$ go back to step 2.

The following algorithm (the lazy counting algorithm) can be used to generate combinations of columns. It generates all 2^{ncols} combinations of $ncols+1$ columns involving c_{ncols} as needed in step 3 of the algorithm above. In addition, only one XOR operation is required to generate the next combination whose nonlinearity and $\mathrm{DHOBIC}_{32,1}$ are to be tested.

1. Set $f = c_{ncols}$.
2. Test the nonlinearity of f.
3. For $i = 1$ to $2^{ncols} - 1$
   a) Find the least significant bit of the binary representation of i which is 1. Let b be the associated bit position number.
   b) Set $f = f \oplus c_b$. ($f_{new} = f_{previous} \oplus c_b$)
   c) Test the nonlinearity and $\mathrm{DHOBIC}_{32,1}$ of f.

4. Constructions of Bent Functions

The method of s-box generation proposed in the previous section requires a set of bent functions. Several construction techniques are known [7] (see also [3, 10, 13]). The ones which we used are described below.

4.1 Generation from 6 input bent functions:

Given a set of bent functions in B_6, bent functions in B_8 can be constructed using either of the following two methods:

4.1.1 Method 1:

Let $a, b \in B_6$. Then the function $f:\{0,1\}^8 \to \{0,1\}$ defined by

$$f(x_7 \ldots x_0) = \begin{cases} a(x_5 \ldots x_0), & x_6 = 0, x_7 = 0 \\ a(x_5 \ldots x_0), & x_6 = 0, x_7 = 1 \\ b(x_5 \ldots x_0), & x_6 = 1, x_7 = 0 \\ b(x_5 \ldots x_0) \oplus \mathbf{1}, & x_6 = 1, x_7 = 1 \end{cases}$$

is bent [3]. Rearrangements of the 64 bit blocks in the expression above also result in bent functions.

4.1.2 Method 2:

Let $a,b,c \in B_6$ and let A, B, C be their respective Walsh Transforms. If

$$D(w_7 \ldots w_0) = \begin{cases} A(w_5 \ldots w_0), & w_6 = 0, w_7 = 0 \\ B(w_5 \ldots w_0), & w_6 = 0, w_7 = 1 \\ C(w_5 \ldots w_0), & w_6 = 1, w_7 = 0 \\ -2^{2n}\left[A(w_5 \ldots w_0)B(w_5 \ldots w_0)C(w_5 \ldots w_0)\right]^{-1}, & w_6 = 1, w_7 = 1 \end{cases}$$

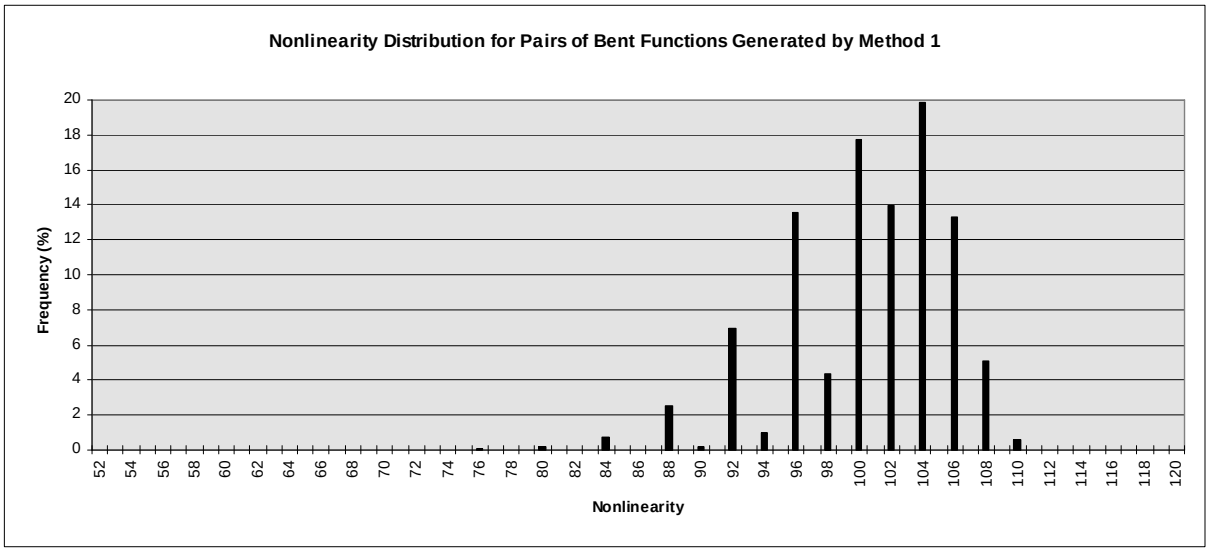
represents the Walsh transform of a function $d: \{0,1\}^n \to \{0,1\}$ then d is bent [13].

4.2 Maiorana Functions

Let $\pi(x)$ be a bijective mapping from $\{0,1\}^{n/2} \to \{0,1\}^{n/2}$, $g(x)$ be a function in $V_{n/2}$, and x_H and x_L denote the high $n/2$ and low $n/2$ bits of x respectively. Let the $\cdot$ operator denote the dot product $\left[a_{n/2-1}\ a_{n/2-2} \cdots a_0\right] \cdot \left[b_{n/2-1}\ b_{n/2-2} \cdots b_0\right] = a_{n/2-1}b_{n/2-1} \oplus a_{n/2-2}b_{n/2-2} \oplus \cdots \oplus a_0 b_0$. Then the function $f \in V_n: f(x) = f(x_H, x_L) = \pi(x_H) \cdot x_L \oplus g(x_H)$ with n even is bent, and is called a Maiorana function [7, 10].

4.3 Properties of Bent Functions Constructed by Method 1

A set of 100000 functions, $f_0 \ldots f_{99999}$, was generated using Method 1. For random, distinct, $i, j \in \{0,1,\ldots,99999\}$, the nonlinearity of $f_i \oplus f_j$ was calculated. The following chart shows the resulting nonlinearity distribution.



Two similar distributions are present. The lower one corresponds to nonlinearities divisible by 2 but not by 4, and the upper to nonlinearities divisible by four. This set of functions has the

greatest nonlinearity spread of all those considered in this paper. In our experiments, it was found that it is possible to build 8×25 s-boxes with nonlinearity 80 and 8×29 s-boxes with nonlinearity 76 with columns in this set. The minimum nonlinearity of a pair of functions was 52.

The set of functions was filtered by taking only groups of 4 functions forming 8×4 s-boxes with nonlinearity at least 100. It was hoped that if one function from a group of four had high nonlinearity when added to each member of a set of functions, the other functions in that group would also have high nonlinearity with respect to that set. Experimentally, it was found that such filtering did not support this hypothesis.

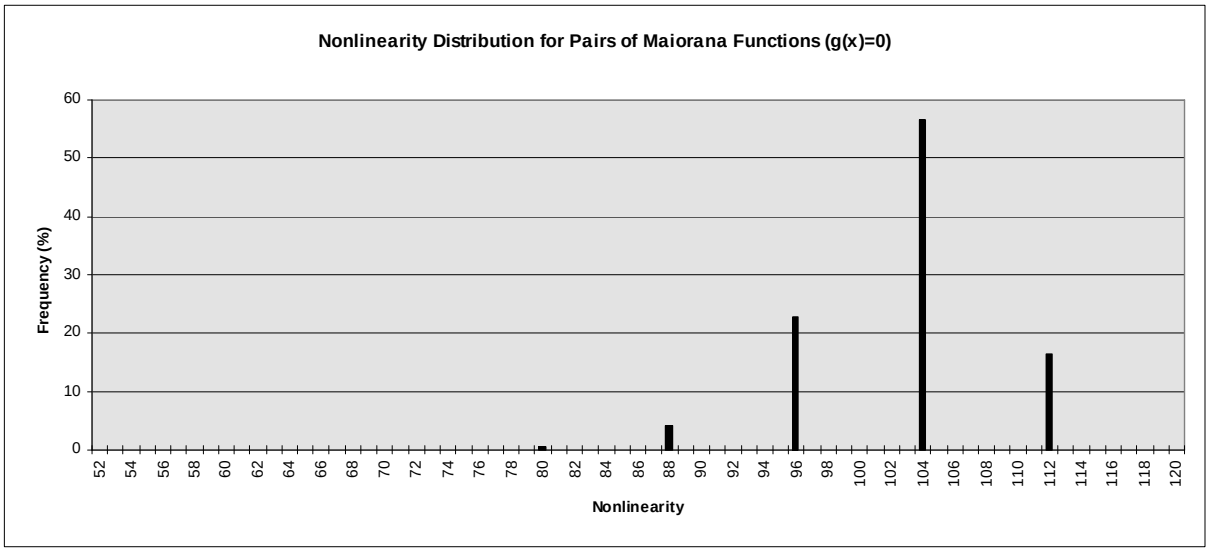
S-Boxes having more than three columns and built with functions constructed by Method 1 have a DBIC value of 64, the worst (highest) possible. This can be observed as follows. Let a_1, b_1 and a_2, b_2 be the truth tables corresponding to two pairs of bent functions in B_6. Let $(a_1, a_1, b_1 \oplus \alpha, b_1 \oplus \beta)$ and $(a_2, a_2, b_2 \oplus \chi, b_2 \oplus \delta)$ be elements of B_8 constructed by Method 1, where exactly one of $\alpha, \beta \in \{\mathbf{0},\mathbf{1}\}$ is **1** and exactly one of $\chi, \delta \in \{\mathbf{0},\mathbf{1}\}$ is **1**. The linear combination of these two functions is $(a_1 \oplus a_2, a_1 \oplus a_2, b_1 \oplus b_2 \oplus \alpha \oplus \chi, b_1 \oplus b_2 \oplus \beta \oplus \delta)$ which achieves a DBIC of 64 for an input change of 01000000 (binary) because the resulting change vector is:

$$\begin{aligned}(a_1 \oplus a_2 \oplus a_1 \oplus a_2, b_1 \oplus b_2 \oplus \alpha \oplus \chi \oplus b_1 \oplus b_2 \oplus \beta \oplus \delta) &= (\mathbf{0}, \alpha \oplus \chi \oplus \beta \oplus \delta) \\ &= (\mathbf{0},\mathbf{0}).\end{aligned}$$

This argument is valid for any rearrangement of the 64 bit blocks provided that both are rearranged in the same way. Thus, an s-box with columns constructed using Method 1 will have a DBIC of 64 if any two columns have the same block arrangement (ignoring complementation). Because there are only three distinct rearrangements, the result follows.

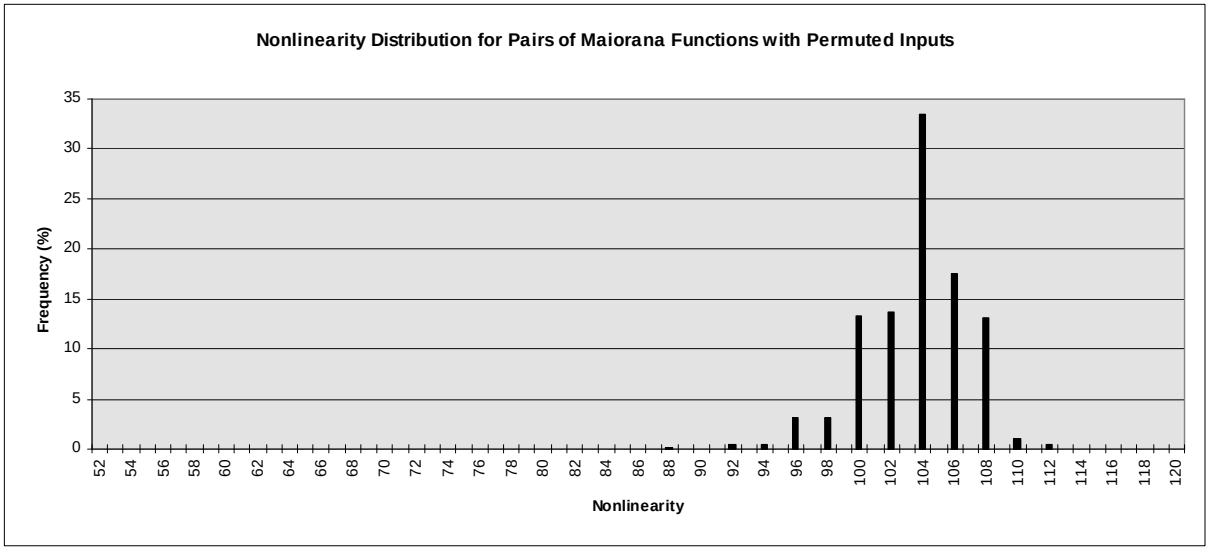
4.4 Properties of Maiorana Functions

For randomly generated pairs of Maiorana functions $m(x)$, $p(x)$ with $g(x) = \mathbf{0}$, the nonlinearity of $m \oplus p$ was calculated. The resulting nonlinearity distribution follows:



The fact that the nonlinearity of linear combinations of Maiorana functions is divisible by $2^{n/2-1}$ is explained in [11]. In our experiments, we found that Maiorana functions cannot be used to generate s-boxes larger than 8×17 of nonlinearity greater than 80 with Alg. 1. The lowest nonlinearity observed for a pair of Maiorana functions was 64.

Let $a,b,\ldots,h$ be a permutation of $0,1,\ldots,7$. Then it is easy to show that the function $f(x_7\ldots x_0) = f_m(x_a\ldots x_h)$, where $f_m()$ is a Maiorana function, is also bent. We will refer to these functions as Maiorana functions with permuted inputs. The nonlinearity distribution of pairs of these functions follows:



With this set of functions, our experiments generated s-boxes with nonlinearity 76 of size 8×30. This is the largest s-box of nonlinearity 76 or greater that was generated from any single set of functions described in this paper. With a nonlinearity of 80, the dimension 8×25 could not be exceeded using our pool of functions. The lowest nonlinearity observed for a pair of Maiorana functions with permuted inputs was 82, the highest of all sets considered here.

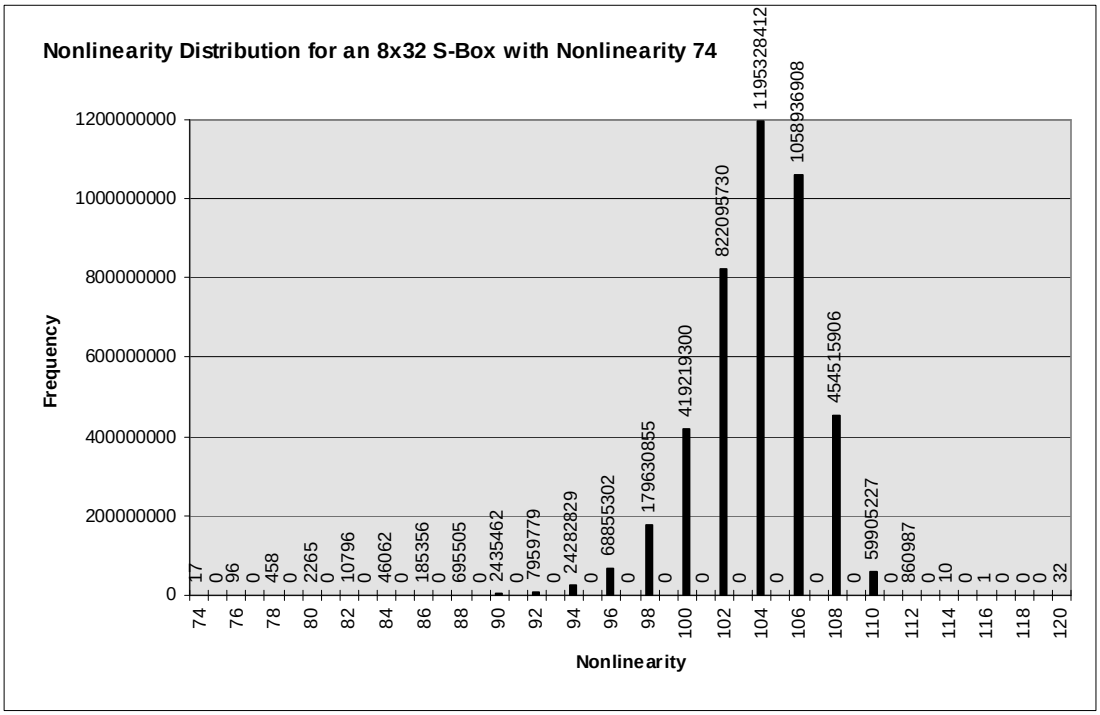
5. Bent Functions in S-Box Design

Experimentally, it was found that 8×32 s-boxes with nonlinearity 74 were best constructed by the following method:

1. build an 8×29 s-box with nonlinearity 76 from bent functions generated by Method 1
2. Append two Maiorana functions with permuted inputs, keeping the s-box nonlinearity at 76
3. Append a Maiorana function with permuted inputs, reducing the s-box nonlinearity to 74

Using this technique, construction of a single s-box takes 15 to 30 days on a Pentium 90. However, the computation can readily be distributed across many computers, reducing the construction time to a few hours.

The following chart shows a typical distribution of nonlinearities of all combinations of the columns of an s-box with nonlinearity 74 generated by themethod described above:

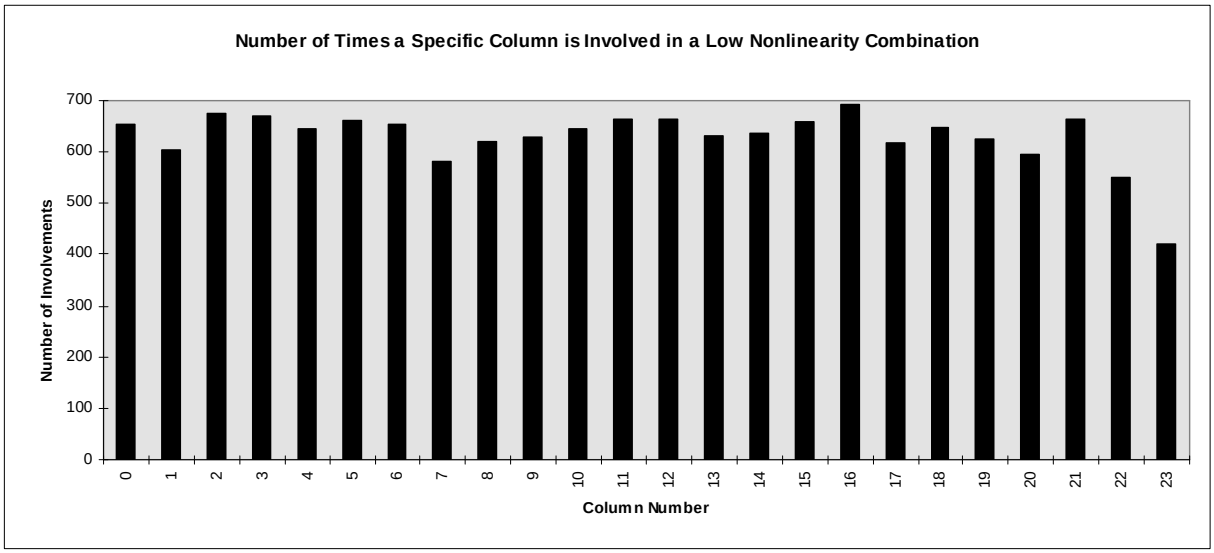


We observe that the nonlinearity of most linear combinations is much higher than 74. Only 17 linear combinations have nonlinearity 74, and 571 have nonlinearity below 80 (out of 2^{32} possible combinations). However, no 8×32 s-boxes with bent functions as columns with nonlinearity 76 or above have yet been found using this construction method.

6. Improvements on the Construction Algorithm

The algorithm described so far simply discards columns which do not have the desired minimum nonlinearity with respect to any combination of columns already in the s-box. Although this method works, it is disturbing that until the technique has found a new column, it is no more likely to find a suitable column than it was immediately after locating the previous one. The following development leads to an improvement over this situation which could possibly be exploited further.

Below is a plot of the number of times a particular s-box column is included in a linear combination having nonlinearity less than 80 during the construction process.



From the graph, it is clear that low nonlinearity cannot be attributed to a few s-box columns.

We next consider the number of times specific combinations of columns are involved in low nonlinearity combinations. An 8×10 s-box of nonlinearity 80 was constructed, and the number of times a combination of these 10 functions when combined with an eleventh bent function resulted in a nonlinearity under 80 was recorded. This was repeated three times. A summary of the results follows:

	Exp. 1	Exp. 2	Exp. 3
Number of functions tested	200 393	18 259	5 908
Combination producing low nonlinearity most often (lazy counting algorithm counter value given)	127	15	31
Maximum number of low nonlinearities caused by a single combination	63 633	2 533	1 058
Percentage of low nonlinearities caused by a single combination	31.8	13.9	17.9
Total number of combinations having generated a low nonlinearity (out of 1024)	115	328	208
Number of combinations which generate low nonlinearities in all experiments	36		
Number of combinations which generate low nonlinearities in last two experiments	162		

We observe that up to 30% of columns could be rejected by performing a nonlinearity test with a fixed combination of the existing s-box columns. Because the combination to be checked differed for all three experiments, the construction algorithm would need to determine the combination which most frequently is involved in low nonlinearities. This is difficult because the full nonlinearity test must be completed even if it is already known part way through the test that the nonlinearity of the s-box with the function being considered is lower than the minimum required.

6.1 Effective Use of Combinations and Subspaces

We now consider an s-box nonlinearity test which checks first that a candidate function has the desired nonlinearity with respect to

1. all combinations of columns not involving the most recently added one
2. all combinations of columns involving the most recently added one.

Suppose that two candidate functions f and g complete part 1 successfully and fail part 2. Then if the function $f \oplus g$ passes part 1, the most recently added column can be replaced with f and g can be added to the s-box. Verifying that $f \oplus g$ passes part 1 involves only half the effort required to verify that a new candidate function passes both parts 1 and 2. This scheme has been implemented and causes a noticeable decrease in the time required to find columns for large s-boxes. The method could be extended, for example, to save time when three functions have the desired nonlinearity for all combinations not involving two specific columns of an s-box.

6.2 Calculating the Nonlinearity of Functions

The speed of s-box construction is greatly dependent on the time required for a nonlinearity calculation. The nonlinearity of a function f can be computed using:

$$nl(f) = 2^{n-1} - \frac{1}{2} \max_{w \in \{0,1\}^n} |W(f)(w)|.$$

The Walsh transform of f can be calculated by multiplying the truth table of $(-1)^{f(x)}$ by a Hadamard Matrix of order n. This matrix is defined recursively by:

$$H_0 = 1$$
$$H_n = \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix}.$$

Let $f[a...b]$ represent the truth table of $(-1)^{f(x)}$ for inputs between a and b inclusive. The equation $H_n f[0...2^n - 1] = \begin{bmatrix} A+B \\ A-B \end{bmatrix}$, where $A = H_{n-1} f[0...2^{n-1} - 1]$ and $B = H_{n-1} f[2^{n-1}...2^n - 1]$, can be used to efficiently calculate the Walsh transform. In our current implementation, $H_4 f$ is calculated by table lookup. In the final stage of the recursion, $|A| + |B|$ is computed instead of $A + B$ and $A - B$ in view of the maximization specified in the formula for nonlinearity.

The improvements described in 6.1 and 6.2 combine to give a nonlinearity calculation requiring approximately $100 \mu s$ on a Pentium 90 (roughly 30-50 times better than a naive implementation of the nonlinearity calculation).

7. Conclusions

The following table summarizes the properties obtained for 8×32 s-boxes constructed according to the methods described in this paper and for random 8×32 s-boxes:

Property	Random S-Box	Constructed S-Box
I1'. Nonlinearity	72*	74
I2. Largest XOR table entry	2	2
I3. DMOSAC	17	0
I4'. $\text{DHOBIC}_{32,1}$	37	36
I5. Row weight distribution	Approximately binomial	Approximately binomial
I6. Row pair distribution	Approximately binomial	Approximately binomial
I7. Average column weight	128	128**

* See [16]

** Half the columns have weight 120 and the other half have weight 136

The constructed s-boxes are equivalent to random s-boxes with respect to properties I2, I5, and I6, and are superior to random s-boxes with respect to properties I1', I3, and I4'. It is therefore conjectured that using constructed s-boxes in CAST-like ciphers will increase security (compared with the use of random s-boxes). Note that although improvements in nonlinearity and $\text{DHOBIC}_{32,1}$ seem minor (74 vs. 72 and 36 vs. 37), it has so far proven impossible to construct 8×32 s-boxes with nonlinearity greater than 74 or $\text{DHOBIC}_{32,1}$ less than 36, and has, on the other hand, been almost trivial to construct 8×32 s-boxes with nonlinearity less than 74 or $\text{DHOBIC}_{32,1}$ greater than 36. Thus, these "minor" improvements may in fact represent significant advances in terms of s-box strength (especially considering that property I3 is so much improved and properties I2, I5, and I6 are not degraded). Also, the construction time for an s-box is only about twice the time required to find the nonlinearity and DBIC for a given s-box. Construction would then typically be advantageous over random generation if the probability of a randomly generated s-box having unsatisfactory nonlinearity or DBIC is not negligible. In most cases, the increase in security would justify the extra time required for construction.

The creation of large, cryptographically good s-boxes with bent functions as columns has proven more difficult than originally expected, although extensive experimentation (as summarized in this paper) has given rise to a method which appears to produce satisfactory

results. The current algorithm takes between 15 and 30 days on a Pentium 90 for an 8×32 s-box with the properties listed in the table above, and each additional column would cause the generation time to double. However, the process can readily be distributed over a number of computers to significantly reduce the time required.

Finally, the concept of dynamic distance presented here defines a quantitative measure of how close an s-box is to satisfying dynamic properties such as SAC, BIC, and their higher orders, and provides an intuitive, unified framework for these properties.

References

[1] C. M. Adams, *Constructing Symmetric Ciphers Using the CAST Design Procedure*, (submitted for publication).

[2] C. M. Adams, *Designing DES-Like Ciphers with Guaranteed Resistance to Differential and Linear Attacks*, Workshop Record of the Workshop on Selected Areas in Cryptography (SAC 95), May 18-19, 1995, pp. 133-144.

[3] C. M. Adams and S. E. Tavares, *Generating and Counting Binary Bent Sequences*, IEEE Transactions on Information Theory, vol. IT-36, 1990, pp. 1170-1173.

[4] C. M. Adams and S. E. Tavares, *The Use of Bent Sequences to Achieve Higher-Order Strict Avalanche Criterion in S-Box Design*, Technical Report TR 90-013, Dept. of Electrical Engineering, Queen's University, Kingston, Ontario, Jan., 1990.

[5] E. Biham, *On Matsui's Linear Cryptanalysis*, Advances in Cryptology - Proceedings of EUROCRYPT '94, Springer-Verlag, Berlin, 1995, pp. 341-355.

[6] E. Biham and A. Shamir, *Differential Cryptanalysis of DES-like Cryptosystems*, Advances in Cryptology: Proceedings of CRYPTO '90, Springer-Verlag, Berlin, 1991, pp. 1-21.

[7] J. F. Dillon, *A Survey of Bent Functions*, NSA Technical Journal, Special Issue, 1972, pp. 191-215.

[8] H. Feistel, *Cryptography and Computer Privacy*, Scientific American, 228 (1973), pp. 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), pp. 1545-1554.

[10] J. A. Maiorana, *A Class of Bent Functions*, R41 Technical Paper, June 1971. (see [7])

[11] S. Mister, *Notes on Maiorana Functions and S-Box Design*, (in progress).

[12] K. Nyberg, *Perfect Nonlinear S-boxes.* Advances in Cryptology - Proceedings of EUROCRYPT '91, Springer-Verlag, Berlin, 1991, pp. 378-385.

[13] B. Preneel, W. Van Leekwijck, L. Van Linden, R. Govaerts, and J. Vandewalle, *Propagation characteristics of boolean functions*, Advances in Cryptology: Proceedings of EUROCRYPT '90, Springer-Verlag, Berlin, 1991, pp. 161-173.

[14] O. S. Rothaus, *On 'Bent' Functions*, Journal of Combinatorial Theory, 20(A), 1976, pp. 300-305.

[15] A. F. Webster and S. E. Tavares, *On the Design of S-Boxes*, Advances in Cryptology: Proceedings of CRYPTO '85, Springer-Verlag, New York, 1986, pp. 523-534.

[16] A. Youssef, S. Tavares, S. Mister, C. Adams, *Linear Approximation of Injective S-boxes*, Electronics Letters, Vol. 31 No. 25, Dec. 7, 1995, pp. 2165-2166.