Nonlinear Boolean Permutations

Abdurashid Mamadolimov1*, Herman Isa1, Miza Mumtaz Ahmad2 and Moesfa Soheila Mohamad1
1Information Security Cluster, Malaysian Institute of Microelectronic Systems (MIMOS Bhd.), Technology Park Malaysia, 57000 Kuala Lumpur, Malaysia
2School of Mathematical Sciences, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia
*E-mail: rashid.mdolimov@mimos.my

ABSTRACT
A Boolean permutation is called nonlinear if it has at least one nonlinear component function. All nonlinear Boolean permutations and their complements are called non-affine Boolean permutations. Any non-affine Boolean permutation is a potential candidate for bijective S-Box of block ciphers. In this paper, we find the number of n-variable non-affine Boolean permutations up to multiplicative n and show a simple method of construction of non-affine Boolean permutations. However, non-affinity property is not sufficient for S-Boxes. Nonlinearity is one of the basic properties of an S-Box. The nonlinearity of Boolean permutation is a distance between set of all non-constant linear combinations of component functions and set of all non-affine Boolean functions. The cryptographically strong S-Boxes have high nonlinearity. In this paper, we show a method of construction of 8-variable highly nonlinear Boolean permutations. Our construction is based on analytically design (8, 1), (8, 2), and (8, 3) highly nonlinear vectorial balanced functions and random permutation for other component functions.

Keywords: Boolean permutation, S-Box, block cipher, nonlinearity

INTRODUCTION
A Boolean function is a map from \(F_2^n\) to \(F_2\) and a vectorial Boolean function is a map from \(F_2^n\) to \(F_2^m\). Vectorial Boolean functions are usually called S-Boxes and are used as basic component of block ciphers. For example, the S-Boxes used in Data Encryption Standard (DES) have \(n=6\) and \(m=4\) and the S-Box used in the Advanced Encryption Standard (AES) has \(n=m=8\). Large S-Boxes \((n \geq 8, m \geq 8)\) are stronger than small one. Usually, \(n \geq m\) and if \(n=m\) then vectorial Boolean function is called Boolean transformation. If Boolean transformation is bijective, then it is called Boolean permutation. In cryptology, Boolean permutations are called bijective S-Boxes. Each component function of Boolean permutation is a Boolean function. Boolean permutation is called linear (affine) if each component function is linear (affine). In bijective S-Boxes, non-affine Boolean permutations are used.

In this paper, we find bounds for the number of \(n\)-variable non-affine Boolean permutations. Lower and upper bounds differs by multiple of \(n\). These bounds prove that the set of all affine Boolean permutations is a very small subset of all Boolean permutations. Therefore, non-affine Boolean permutations are not rare.

Received: 1 August 2010
Accepted: 22 June 2011
*Corresponding Author
The nonlinearity of Boolean permutation is the (Hamming) distance between the set of all non-constant linear combinations of component functions and the set of all non-affine Boolean functions. Non-affinity property alone is insufficient for S-Boxes. The linear cryptanalysis introduced by Matsui (1994), which is based on finding affine approximation to the action of cipher (Matsui, 1994). The linear attack on a function is successful if nonlinearity of a function is low. Highly nonlinear functions possess the best resistance to the linear attack. Therefore, nonlinearity is an essential property for Boolean permutations used as bijective S-Boxes.

Upper bounds of nonlinearity for different classes of functions exist (Maxwell, 2005; Budaghyan, 2005; Sulak, 2006). For \( n \)-variable Boolean permutations, when \( n \) is even, functions with nonlinearity \( 2^{n-1} - 2 \frac{n}{2} \) are known. It is conjectured that this value is the highest possible nonlinearity for the \( n \) even case. In this case each component function’s highest nonlinearity is \( 2^{n-1} - 2 \frac{n}{2} - 2 \).

The problem in constructing highly nonlinear bijective S-Boxes has been studied in (Cui and Cao, 2007; Jin et al., 2006; Sakalauskas and Luksys, 2007). Methods of construction could be separated into two groups: analytic and algorithmic. Almost all analytical methods of construction are based on finite field theory. Vectorial Boolean functions can be considered also as a map from finite field \( F_2 \) to finite field \( F_2^m \). The power map \( x \mapsto x^d \), where \( x \in F_2^m \), has been systematically studied in (Budaghyan, 2005). A power map is a Boolean permutation if and only if \( \gcd (d, 2^n - 1) = 1 \). Some highly nonlinear power permutations are known (Budaghyan, 2005). For example, the power permutation \( x \mapsto x^r \), \( x \in F_2^n \), which in fact is the inverse function, has the known highest nonlinearity when \( n \) is even. AES’s S-Box is based on 8-variable inverse function. The second group of construction of highly nonlinear Boolean permutations is algorithmic (Watanabe et al., 2007; Clark et al., 2005; Fuller et al., 2005; Seberry et al., 1993). Usually, algorithmic method is done by increasing the nonlinearity in steps. As a rule, the cryptographic properties of such algorithmic S-Boxes are not optimal.

We suggest a construction of Boolean permutations by analytically designing \((8, 1), (8, 2)\) and \((8, 3)\) highly nonlinear vectorial balanced functions and randomly permuting other component functions.

**PRELIMINARIES**

Let \( F_2 \) be the finite field with two elements and let \(( F_2^n, \oplus) \) be the vector space over \( F_2 \), where \( \oplus \) is used to denote the addition operator over both \( F_2 \) and the vector space \( F_2^n \) (Pieprzyk, 1989; Nyberg, 1993; 1994).

An \( n \)-variable Boolean function (filter) is a map

\[
f = f(x_1, \ldots, x_n) : F_2^n \to F_2
\]

The (Hamming) weight \( wt(f) \) of a Boolean function \( f \) on \( n \) variables is the weight of this string, that is, the size of the support \( sp(f) = \{ x \in F_2^n : f(x) = 1 \} \) of the function. The function \( f \) is said to be balanced if \( wt(f) = 2^{n-1} \). The (Hamming) distance between two Boolean functions \( f \) and \( g \) is \( d(f, g) = | \{ x : f(x) \neq g(x) \} | \). Clearly, the distance between Boolean functions \( f \) and \( g \) is equal to the weight of sum of these functions i.e., \( d(f, g) = wt(f \oplus g) \).

If we denote by \( B(n) \) the set of all \( n \)-variable Boolean functions then we have \( |B(n)| = 2^{2^n} \).

An \((n,m)\) vectorial Boolean function (S-Box) is a map.
\[ F = F(x_1, \ldots, x_n) = (f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)) : F_2^1 \rightarrow F_2^m. \] Clearly, each component function \( f_i, i = 1, \ldots, m \) is an \( n \)-variable Boolean function. An \((n,n)\) vectorial Boolean function is called \( n \)-variable Boolean transformation.

An \((n,m)\) vectorial Boolean function is called balanced if it takes every value of \( F_2^m \) the same number of times. If a Boolean transformation is balanced then it takes every value of \( F_2^n \) one time. A balanced \( n \)-variable Boolean transformation is called \( n \)-variable Boolean permutation. Clearly, \( n \)-variable Boolean permutation is bijective function from \( F_2^n \) into itself.

Let \( F = F(x_1, \ldots, x_n) = (f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)) \) be a Boolean transformation and let \( c \cdot F = c_1f_1 \oplus \cdots \oplus c_nf_n, c = (c_1, \ldots, c_n) \in F_2^n, c \neq 0 \) be non-constant linear combination of component functions. A Boolean transformation is a Boolean permutation if and only if each non-constant linear combination of component functions is balanced.

If we denote the set of all \((n,m)\) vectorial Boolean functions, \( n \)-variable Boolean transformations and \( n \)-variable Boolean permutations as \( BF(n,m) \), \( BT(n) \) and \( BP(n) \) respectively, then we have \[ |BF(n,m)| = 2^{nm}, |BT(n)| = 2^{n-1} \text{ and } |BP(n)| = 2^n! \]

The unique representation of \( n \)-variable Boolean function \( f \) as a polynomial over \( F_2 \) in \( n \) variables of the form \( f(x_1, \ldots, x_n) = \sum_{a \in F_2^n} c(a) \prod_{i=1}^n x_i^a \) is called the algebraic normal form (ANF) of \( f \). The degree of the ANF of \( f \) is denoted by \( d(f) \) and is called the algebraic degree of the function \( f \).

An \( n \)-variable Boolean function is called linear (affine) if its ANF is affine if \( d(f) \leq 1 \) and \( f \) is linear if it is affine and \( f(0) = 0 \). An \((n,m)\) vectorial Boolean function \( F = (f_1, \ldots, f_m) \) is called linear (affine) if each component function \( f_1, \ldots, f_m \) is linear (affine). In this paper we concentrate on non-affine Boolean permutations.

Let \( A(n) \) be the set of all \( n \)-variable affine Boolean functions. The nonlinearity \( N_f \) of an \( n \)-variable Boolean function \( f \) is defined as \( N_f = \min_{g \in A(n)} d(f, g) \), i.e., the nonlinearity of \( f \) is a distance between \( f \) and the set \( A(n) \) of all \( n \)-variable affine Boolean functions. Clearly, \( N_f = 0 \) if and only if \( f \) is an affine function. It is known that for any \( n \)-variable Boolean function \( f \), the nonlinearity \( N_f \) satisfies the following relation: \( N_f \leq 2^{n-1} - 2^{\lceil n/2 \rceil - 1} \). Functions achieving the equality are called bent functions which exist when \( n \) is even. However, bent functions are not balanced. If \( f \) be a balanced \( n \)-variable Boolean function \((n \geq 3)\). Then the nonlinearity of function \( f \) is given by

\[ N_f \begin{cases} 2^{n-1} - 2^{\lceil n/2 \rceil - 1} - 2, & n \text{ even} \\ \lceil 2^{n-1} - 2^{\lceil n/2 \rceil - 1} \rceil, & n \text{ odd} \end{cases} \]

where \( \lceil x \rceil \) denotes the largest even integer less than or equal to \( x \).

We can compute nonlinearity \( N_f \) of an \( n \)-variable Boolean function \( f \) by following way:

\[ N_f = \min_{(c_0, c_1, \ldots, c_n) \in F_2^n} wt(f \oplus c_0 \oplus c_1 x_1 \oplus \cdots \oplus c_n x_n). \]

The nonlinearity \( N_F \) of an \((n,m)\) vectorial Boolean function \( F \) is defined as \( N_F = \min_{c \in F_2^m, c \neq 0} N_{c,F} \)

In the other words, the nonlinearity of function \( F \) is a distance between the set of all non-constant linear combinations of component functions of \( F \) and the set \( A(n) \) of all \( n \)-variable affine Boolean functions. This shows that \( N_F = 0 \) if \( F \) is affine. However, the condition \( N_F = 0 \) does not explain the affinity of \( F \). It is known that for any \((n,m)\) vectorial Boolean function \( F \), the nonlinearity, \( N_F \) satisfies \( N_f \leq 2^{n-1} - 2^{\lceil n/2 \rceil - 1} \). Functions achieving the equality are called perfectly nonlinear and...
can exist only when $n$ is even and $m \leq \frac{n}{2}$. If $n$ is odd and $n=m$ then we have $N_r \leq 2^{n+1} - 2^\frac{n}{2}$.

Functions with nonlinearity $2^{n+1} - 2^\frac{n}{2}$ are known for even $n$ and $n=m$, and it is conjectured that this value is the highest possible nonlinearity.

**THE NUMBER OF NON-AFFINE BOOLEAN PERMUTATIONS**

We denote the set of all non-affine $n$-variable Boolean permutations by $NABP(n)$. Note that $NABP(n) \subset BP(n) \subset BT(n)$.

**Theorem.** Let
$$\mu(n) = 2^n \cdot \left(2^{n+1} - 2\right).$$
Then the number of non-affine Boolean permutations satisfies
$$\mu(n) \leq |NABP(n)| \leq n \cdot \mu(n)$$

**Proof:** For proving the left side of inequality, it is enough to show that we can construct $\mu(n)$ different non-affine $n$-variable Boolean permutations. Clearly, an $n$-variable Boolean permutation is just permutation of $F^n_2$ vectors. Our method of construction contains two steps:

i. Choose balanced non-affine $n$-variable Boolean function as first component function $f_1$ of Boolean permutation.

ii. Choose two permutations of $F^n_2$ vectors and set the permuted vectors as values of $(0, f_2, \ldots, f_n)$ and $(1, f_2, \ldots, f_n)$, respectively.

The resulting function $F = (f_1, f_2, \ldots, f_n)$ is a non-affine Boolean permutation.

Any non-constant affine function is balanced. Since, $|A(n)| = 2^{n+1}$ and the number of constant affine functions is 2, the number of balanced affine Boolean functions is $2^{n+1} - 2$ while the number of $n$-variable balanced Boolean functions is $\left(\frac{2^n}{2^{n+1}}\right)$. Therefore the number of balanced non-affine Boolean function is $\left(\frac{2^n}{2^{n+1}}\right) - (2^{n+1} - 2)$. The number of permutations in step ii) is $(2^{n+1} - 1)^2$. Thus, we have
$$\left(2^{n+1} - 2\right)\cdot \left(2^n - (2^{n+1} - 2)\right) = 2^n \cdot \left(2^{n+1} - 2\right) = \mu(n)$$
distinct non-affine Boolean permutations.

To prove the right side of inequality, we first construct $n \cdot \mu(n)$ non-affine Boolean permutations. Then we show that each non-affine Boolean permutation can be obtained by our construction. In the above construction if we take $i$-th component as balanced non-affine fixed function for each $i=1,2,\ldots,n$ then we have $n \cdot \mu(n)$ non-affine Boolean permutations. Let $F = (f_1, \ldots, f_n)$ be any non-affine Boolean permutation. Then F has at least one non-affine component function $f_i$. Clearly, the Boolean permutation $F = (f_1, \ldots, f_n)$ can be obtained by permuting the vectors of $F^n_2$ such that in the obtained Boolean permutations $i$-th component function is same with $f_i$.

Table 1 showed us the number of functions of three classes for some small $n$. 
Nonlinear Boolean Permutations

TABLE 1
The number of functions of three classes

<table>
<thead>
<tr>
<th>n</th>
<th>BT(n)</th>
<th>BP(n)</th>
<th>NABP(n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>256</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>16,777,216</td>
<td>40,320</td>
<td>≥ 32,256</td>
</tr>
<tr>
<td>8</td>
<td>≈ 10^{14}</td>
<td>≈ 10^{13}</td>
<td>≈ 10^{12}</td>
</tr>
</tbody>
</table>

CONSTRUCTION HIGHLY NONLINEAR 8-VARIABLE BOOLEAN PERMUTATIONS

In the above construction we choose non-affine Boolean function as first component and after permuting of $F_2^{8-1}$ vectors by first component we obtain non-affine Boolean permutation. If we choose a highly nonlinear Boolean function as the first component we can obtain highly nonlinear Boolean permutation.

We extend the construction to having highly nonlinear balanced (8, 2) and (8, 3) vectorial Boolean functions as first two and first three component functions of Boolean permutation, respectively.

Note that 8-variable highly nonlinear Boolean permutation is eight 8-variable Boolean functions $f_i(x_1,\ldots,x_8)$, $i=1,\ldots,8$, where each non-constant linear combination of these functions is balanced and has high nonlinearity.

We note that the highest known nonlinearity for 8-variable Boolean permutations is 112, while each their component function’s highest nonlinearity is 118.

The following construction is filled in.

A. Design of the First Component Function

We want to design highly nonlinear balanced Boolean function. We consider the function

$$g_1(x_1,\ldots,x_8) = x_1x_2 \oplus x_3x_4 \oplus x_5x_6 \oplus x_7x_8$$

This function is bent and $N_{g_1} = \text{wt}(g_1) = 120$. The function

$$g_2(x_2,x_4,x_6,x_8) = x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_1x_2x_3 \oplus x_5x_6x_7$$

is a balanced function and has highest nonlinearity ($N_{g_2} = 4$) in $F_2^4$. Let

$$f_1(x_1,\ldots,x_8) = \begin{cases} g_2(x_2,x_4,x_6,x_8), & \text{if } (x_1,x_3,x_5,x_7) \\ g_1(x_1,\ldots,x_8), & \text{otherwise} \end{cases}$$

Since $g_1(0,x_2,0,x_4,0,x_6,0,x_8) = 0$ for all $(x_2,x_4,x_6,x_8) \in F_2^4$, $\text{wt}(g_1)=120$ and $g_2$ is balanced then $f_1$ is a balanced function. We have $N_{f_1} = 116$. We use the function $f_1$ as first component function for Boolean permutation. Note that instead of $g_2$ we can use any balanced Boolean function in $F_2^4$ with nonlinearity 4. The total number of such functions is 10920.

B. Design of the First Two Component Functions

We want to design two functions, $f_1$ and $f_2$, where $f_1$, $f_2$ and $f_1 + f_2$ are balanced and have high nonlinearity.
We consider the following two functions:

\[ f'(x_1, \ldots, x_n) = x_1 x_2 \oplus x_1 x_3 \oplus x_1 x_4 \oplus x_2 x_3 \oplus x_2 x_4 \oplus x_3 x_4 \]

\[ f'(x_1, \ldots, x_n) = x_1 x_2 \oplus x_1 x_3 \oplus x_1 x_4 \oplus x_2 x_3 \oplus x_2 x_4 \oplus x_3 x_4 \]

We have \( wt(f') = 136, \) \( wt(f_2) = 118, \) \( wt(f_1 + f_2') = 122 \) and \( \min\{N_2, N_1, N_{2+1}\} = 118. \) We change 8 values of function \( f' : f'(5), f'(20), f'(30), f'(119), f'(155), f'(212), f'(240) \) from 1 to 0. We also change 10 values of function \( f' : f'(30), f'(32), f'(38), f'(78), f'(103), f'(119), f'(140), f'(167), f'(212), f'(240) \) from 0 to 1. Let obtained functions be \( f \) and \( f_2, \) respectively. Then we have \( wt(f_1) = wt(f_2) = wt(f) = 128 \) and \( N_2 = N_1 = N_{2+1} = 112. \)

C. Design of the First Three Component Functions

We want to design three \( f_1, f_2 \) and \( f_3 \) functions, where functions \( f_1, f_2, f_3, f_1 + f_2, f_1 + f_3, f_2 + f_3 \) and \( f_1 + f_2 + f_3 \) are balanced and have high nonlinearity.

Let \( f_1, f_2 \) be the functions from section B and \( f_3 \) be \( f_1 \) from section A. Then we have

\[ wt(f_1) = wt(f_2) = wt(f_3) = wt(f_1 + f_2) = wt(f_1 + f_3) = 128, \]

\[ wt(f_1 + f_2 + f_3) = 142, wt(f_1 + f_2 + f_3) = 130 \] and

\[ N_2 = N_1 = N_{2+1} = 112, N_1 = 116, N_{2+1} = 106, N_{2+1} = 108. \]

We change (0 becomes 1, 1 becomes 0) 14 values of function \( f_1 : f_1'(11), f_1'(12), f_1'(87), f_1'(88), f_1'(90), f_1'(91), f_1'(106), f_1'(107), f_1'(114), f_1'(115), f_1'(158), f_1'(159), f_1'(185), f_1'(186). \) Let the resulting function be \( f_3. \) Then we will have

\[ wt(f_3) = wt(f_3) = wt(f_3) = wt(f_1 + f_2) = wt(f_1 + f_3) = wt(f_1 + f_2 + f_3) = 128 \] and

\[ N_2 = N_1 = N_{2+1} = 112, N_1 = 108, N_{2+1} = 106, N_{2+1} = 104. \]

D. Random Generation and Result of Experiment

We have three types of constructions, \( A, B \) and \( C. \) In Case A two random permutations of \( F_2^n \) vectors set as values of \((0, f_2, \ldots, f_n) \) and \((1, f_2, \ldots, f_n) \). The number of 8-variable Boolean permutations obtained in this way is \((128!)^2 \). In Case B four random permutations of \( F_2^n \) vectors set as values of \((0, 0, f_3, \ldots, f_n), (0, 1, f_3, \ldots, f_n), (1, 0, f_3, \ldots, f_n), (1, 1, f_3, \ldots, f_n) \). The number of 8-variable Boolean permutations obtained in this way is \((64!)^4 \). While in Case C eight random permutations of \( F_2^n \) vectors set as values of \((0,0,0, f_3, \ldots, f_n), (0,0,1, f_3, \ldots, f_n), \ldots, (1,1,1, f_3, \ldots, f_n) \). The number of 8-variable Boolean permutations obtained in this way is \((32!)^8 \).

The results of experiment with 100,000 generations for each of the three cases are showed in Table 2 and Fig. 1.
Nonlinear Boolean Permutations

### TABLE 2
Nonlinearity distribution

<table>
<thead>
<tr>
<th>Nonlinearity</th>
<th>Frequency</th>
<th>Case A</th>
<th>Case B</th>
<th>Case C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-74</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>76</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>78</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>80</td>
<td>9</td>
<td>1</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>82</td>
<td>67</td>
<td>15</td>
<td>69</td>
<td>0</td>
</tr>
<tr>
<td>84</td>
<td>291</td>
<td>66</td>
<td>274</td>
<td>0</td>
</tr>
<tr>
<td>86</td>
<td>1141</td>
<td>287</td>
<td>1117</td>
<td>0</td>
</tr>
<tr>
<td>88</td>
<td>4129</td>
<td>1350</td>
<td>3946</td>
<td>0</td>
</tr>
<tr>
<td>90</td>
<td>13029</td>
<td>5230</td>
<td>12880</td>
<td>0</td>
</tr>
<tr>
<td>92</td>
<td>31565</td>
<td>17103</td>
<td>30831</td>
<td>0</td>
</tr>
<tr>
<td>94</td>
<td>38420</td>
<td>38331</td>
<td>39062</td>
<td>0</td>
</tr>
<tr>
<td>96</td>
<td>11192</td>
<td>37617</td>
<td>11622</td>
<td>0</td>
</tr>
<tr>
<td>98</td>
<td>150</td>
<td>0</td>
<td>178</td>
<td>0</td>
</tr>
<tr>
<td>100-112</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>100000</td>
<td>100000</td>
<td>100000</td>
<td>0</td>
</tr>
</tbody>
</table>

*Fig. 1: Nonlinearity distribution*
In Table 3, the comparison of our generated Boolean permutation and known 8-variable bijective S-Boxes is shown. Note that the first seven S-Boxes in this table are based on inverse function in $F_2^n$ and they are linearly equivalent to each other.

### TABLE 3
Comparison of $8 \times 8$ Bijective S-Boxes

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AES</td>
<td>112</td>
<td>Finite field</td>
</tr>
<tr>
<td>2</td>
<td>Grand-Cru</td>
<td>112</td>
<td>Finite field</td>
</tr>
<tr>
<td>3</td>
<td>Mugi</td>
<td>112</td>
<td>Finite field</td>
</tr>
<tr>
<td>4</td>
<td>Q</td>
<td>112</td>
<td>Finite field</td>
</tr>
<tr>
<td>5</td>
<td>Scream</td>
<td>112</td>
<td>Finite field</td>
</tr>
<tr>
<td>6</td>
<td>Camellia</td>
<td>112</td>
<td>Finite field</td>
</tr>
<tr>
<td>7</td>
<td>Square</td>
<td>112</td>
<td>Finite field</td>
</tr>
<tr>
<td>8</td>
<td>Hierocrypt</td>
<td>106</td>
<td>Unknown</td>
</tr>
<tr>
<td>9</td>
<td>Skipjack</td>
<td>100</td>
<td>Unknown</td>
</tr>
<tr>
<td>11</td>
<td>Khazad</td>
<td>96</td>
<td>Random</td>
</tr>
<tr>
<td>12</td>
<td>Anubis</td>
<td>94</td>
<td>Random</td>
</tr>
</tbody>
</table>

### CONCLUSIONS
Non-affine Boolean permutations are not rare. However, highly nonlinear ($\geq 98$, in $n=8$ case) Boolean permutations are not many.

We suggest a new method of construction of highly nonlinear 8-variable Boolean permutations. Our construction technique combines both the analytical and random approach, by analytically design the first three component functions and randomly generate the remaining components.

From each 8-variable S-Box, approximately $10^{19}$ linearly equivalent S-Boxes can be obtained and all these S-Boxes have the same nonlinearity. It is not possible to consider S-Box as “new”, if it is linearly equivalent to one of the known cipher’s S-Box. However, the S-Box obtained from our construction S-Box with parameters in Table 3 is a new one.

The suggested construction can be generalized for any S-Box size.

### REFERENCES


Nonlinear Boolean Permutations


