Serdica J. Computing **10** (2016), No 3–4, 245–262

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

ALGORITHMS FOR COMPUTING THE LINEARITY AND DEGREE OF VECTORIAL BOOLEAN FUNCTIONS*

Stefka Bouyuklieva, Iliya Bouyukliev

Abstract. In this article, we study two representations of a Boolean function which are very important in the context of cryptography. We describe Möbius and Walsh Transforms for Boolean functions in details and present effective algorithms for their implementation. We combine these algorithms with the Gray code to compute the linearity, nonlinearity and algebraic degree of a vectorial Boolean function. Such a detailed consideration will be very helpful for students studying the design of block ciphers, including PhD students in the beginning of their research.

1. Introduction. S-boxes are key building blocks in the design of block ciphers. They have to be chosen carefully to make the cipher resistant against all kinds of attacks. In particular there are well studied criteria that a good S-box has to fulfill to make the cipher resistant against differential and linear cryptanalyses. The mathematical structure of an S-box is closely connected with the

ACM Computing Classification System (1998): F.2.1, F.2.2.
Key words: Boolean function, Walsh transform, S-box, linearity, algorithms.
*Partially supported by grant DN 02/2/13.12.2016 of the Bulgarian National Science Fund.

Boolean functions, furthermore they are also called vectorial Boolean functions. To compute their parameters we need effective algorithms for the calculation of different elements and forms of the Boolean functions. The Algebraic Normal Form and the Walsh spectrum of the Boolean functions are very important in cryptographic applications. The Walsh spectrum measures the distance to the linear and affine functions. Some authors calculate the linearity using the Walsh spectrum, but others obtain the nonlinearity considering Reed-Muller codes. The relation between the linearity and nonlinearity of a Boolean function is given by an equality, and the minimum linearity corresponds to maximum nonlinearity. To understand this connection better, we consider in more details the Truth Tables of the linear Boolean functions. There are many publications in this direction but it is difficult to find a detailed description of the effective methods and algorithms for their computation, which gives the essence and the motivation in the explanation. Such a detailed consideration would be very helpful for students studying the design of block ciphers, including PhD students in the beginning of their research. In this paper, we give a summary of the different basic approaches and the corresponding algorithms for computing the Algebraic Normal Form and the Walsh spectrum of a Boolean function, and also the linearity and degree of an S-box.

A Boolean function f is a mapping from $\mathbb{F}_2^n$ into $\mathbb{F}_2$. The lexicographic order in $\mathbb{F}_2^n$ is very important in the representations of the Boolean functions. For the vectors in $\mathbb{F}_2^n$ it is defined by

$$(v_1, v_2, \ldots, v_n) \prec (w_1, w_2, \ldots, w_n)$$
$$\iff \exists j \in \{1, 2, \ldots, n\} : v_i = w_i \text{ for } i = 1, \ldots, j-1, \text{ and } v_j < w_j.$$

Example 1. In $\mathbb{F}_2^3$ we have

$$(000) \prec (001) \prec (010) \prec (011) \prec (100) \prec (101) \prec (110) \prec (111).$$

This ordering corresponds to the natural ordering of the integers if we consider the bijection between the set $\{0, 1, 2, \ldots, 2^n - 1\}$ of integers and $\mathbb{F}_2^n$, defined by

$$a \in \{0, 1, 2, \ldots, 2^n - 1\} \longmapsto \overline{a} = (a_1, a_2, \ldots, a_n) \in \mathbb{F}_2^n,$$

where $a = a_1.2^{n-1} + a_2.2^{n-2} + \cdots + a_{n-1}.2 + a_n$. This shows that the vector $\overline{a} \in \mathbb{F}_2^n$ is obtained as a binary representation of the integer a (with added zeros on the left if necessary). Obviously,

$$\overline{a} \prec \overline{b} \iff a < b, \quad a, b \in \mathbb{Z}, 0 \leq a, b \leq 2^n - 1.$$

By $\langle u, w\rangle$ we denote the *dot product* (or scalar product) of the vectors $v, w \in \mathbb{F}_2^n$:

$$\langle v, w\rangle = v_1w_1 \oplus v_2w_2 \oplus \cdots \oplus v_nw_n \in \mathbb{F}_2,\ \ v = (v_1, v_2, \ldots, v_n),\ w = (w_1, w_2, \ldots, w_n).$$

Any Boolean function f can be uniquely represented by a *Truth Table (TT)*, which is a vector whose coordinates are the function values of f after the lexicographic ordering of the inputs. We denote the Truth Table considered as a vector-column by $[f]$. We mean by (Hamming) *weight* $\mathrm{wt}(f)$ and support of a function f, the (Hamming) weight and the support of the corresponding vector $TT(f)$. Analogously, the (Hamming) *distance* between two functions (denoted by d_H) is equal to the (Hamming) distance between the corresponding Truth Tables. If f and g are two Boolean functions, then $(f \oplus g)(x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_n) \oplus g(x_1, x_2, \ldots, x_n)$ and therefore

$$TT(f \oplus g) = TT(f) \oplus TT(g),\ \ [f \oplus g] = [f] \oplus [g].$$

Example 2. Let $f : \mathbb{F}_2^3 \to \mathbb{F}_2$ be the Boolean function given by $f(x_1, x_2, x_3) = x_1x_2x_3 \oplus x_2x_3 \oplus x_1 \oplus 1$. Since $f(000) = 1$, $f(001) = 1$, $f(010) = 1$, and $f(011) = f(100) = f(101) = f(110) = f(111) = 0$, the Truth Table of f is $TT(f) = (11100000)$.

Another way of uniquely representing a Boolean function f is by means of a polynomial in n variables from the ring $\mathbb{F}_2[x_1, \ldots, x_n]/(x_1^2 \oplus x_1, \ldots, x_n^2 \oplus x_n)$ and is called its *algebraic normal form* (ANF) [6]. The considered ring consists of all binary polynomials in n variables whose monomials have the form $x_{i_1}x_{i_2}\cdots x_{i_k}$, $1 \le i_1 < i_2 < \cdots < i_k \le n$, $0 \le k \le n$. Denote by x^u the monomial $x_1^{u_1}x_2^{u_2}\ldots x_n^{u_n}$ where $u \in \mathbb{Z}$, $0 \le u \le 2^n - 1$, $\overline{u} = (u_1, u_2, \ldots, u_n) \in \mathbb{F}_2^n$. Then the Algebraic Normal Form of f is a polynomial

$$f(x) = f(x_1, x_2, \ldots, x_n) = \bigoplus_{u=0}^{2^n-1} a_u x^u. \tag{1}$$

The ANF of a Boolean function f is also called Zhegalkin's polynomial [12]. The degree of $\mathrm{ANF}(f)$ is called the *algebraic degree* $\deg(f)$ of f, and it is equal to the maximum number of variables of the terms x^u, or

$$\deg(f) = \max\{\mathrm{wt}(\overline{u}) | a_u = 1\},\ \text{where } f(x) = \bigoplus_{u=0}^{2^n-1} a_u x^u.$$

Example 3. Let $f(x_1, x_2, x_3) = x_1x_2x_3 \oplus x_2x_3 \oplus x_1 \oplus 1$. Then

$$f(x) = x^0 \oplus x^3 \oplus x^4 \oplus x^7,$$

where $x^0 = 1$, $x^3 = x_1^0 x_2^1 x_3^1$, $x^4 = x_1$, and $x^7 = x_1x_2x_3$. The degree of f is 3.

The Boolean functions of algebraic degree at most 1 play a special role in our investigations, therefore we present the following definition.

Definition 1. [3] A Boolean function in the form $f(x) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus \cdots \oplus a_nx_n$ is called *affine*. If $a_0 = 0$, namely $f(x) = a_1x_1 \oplus a_2x_2 \oplus \cdots \oplus a_nx_n = \langle a, x\rangle$, where $a = (a_1, a_2, \ldots, a_n)$ and $x = (x_1, x_2, \ldots, x_n)$, the function is called *linear*.

Obviously, ANF can be associated with the binary vector $(a_0, \ldots, a_{2^n-1}) \in \mathbb{F}_2^{2^n}$ whose coordinates are the coefficients in (1) following the lexicographical order. We will denote this vector considered as a column by $[A_f]$.

In Section 2 we study the connection between TT and ANF and present fast algorithms for calculating the vector $[f]$ knowing $[A_f]$ and vice versa. Section 3 is devoted to the Walsh transform of a Boolean function and methods for the calculation of its Walsh spectrum. In Section 4 we introduce vectorial Boolean functions (called S-boxes in cryptography) and discuss algorithms to calculate some their parameters.

2. Truth Table and Algebraic Normal Form. In this Section we present methods for calculating the Truth Table of a Boolean function if we know its ANF and vice versa. We know that both translations are equivalent to the Möbius transform [6]. The first method was given in the Introduction and it is based on the definition of TT (see Example 2).

To obtain a more efficient algorithm, we use the following transformations

$$TT(f) = TT\Big(\bigoplus_{u=0}^{2^n-1} a_u x^u\Big) = \bigoplus_{u=0}^{2^n-1} a_u TT(x^u)$$

$$= a_0TT(x^0) \oplus a_1TT(x^1) \oplus a_{2^n-1}TT(x^{2^n-1}) = [A_f]^T \begin{pmatrix} TT(x^0) \\ TT(x^1) \\ \vdots \\ TT(x^{2^n-1}) \end{pmatrix}.$$

Considering vector-columns, we have

$$[f] = TT(f)^T = ([x^0]\ [x^1]\ \cdots\ [x^{2^n-1}])[A_f] = A_n[A_f], \tag{2}$$

where $A_n = ([x^0]\ [x^1]\ \cdots\ [x^{2^n-1}])$. This proves the following theorem (see [3]):

Theorem 1. If f is a Boolean function in algebraic normal form given by the vector-column $[A_f]$ then $[f] = A_n[A_f] \pmod 2$ where $[f]$ is the Truth Table of f.

Let us focus on the matrix A_n. We begin with an example.

Example 4. Compose the matrices A_n for $n \leq 3$. Obviously, $A_0 = (1)$.

- For $n = 1$ we have $TT(x^0) = (11), TT(x^1) = (01)$,
- If $n = 2$ then $TT(x^0) = TT(1) = (1111)$, $TT(x^1) = TT(x_2) = (0101)$, $TT(x^2) = TT(x_1) = (0011)$, $TT(x^3) = TT(x_1x_2) = (0001)$,
- For $n = 3$ the Truth Tables are $TT(x^0) = (11111111)$, $TT(x^1) = (01010101)$, $TT(x^2) = (00110011)$, $TT(x^3) = (00010001)$, $TT(x^4) = (00001111)$, $TT(x^5) = (00000101)$, $TT(x^6) = (00000011)$, $TT(x^7) = (00000001)$.

Therefore

$$A_1 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}, \quad A_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Let $n \geq 2$ and $g_u(x) = x^u$ for $u = 0, 1, \ldots, 2^n - 1$. It is easy to see that for $u < 2^{n-1}$ we have $\overline{u} = (0, u_2, \ldots, u_n)$, hence $g_u(x) = x_2^{u_2} \ldots x_n^{u_n}$ and $g_u(0, x_2, \ldots, x_n) = g_u(1, x_2, \ldots, x_n) = g_u(x_2, \ldots, x_n)$. It follows that $TT(x^u) = (TT'(x^u)|TT'(x^u))$, where $TT'(x^u)$ is the Truth Table of $x_2^{u_2} \ldots x_n^{u_n}$, considered as a function of $n-1$ variables. In the case $u \geq 2^{n-1}$ we have $\overline{u} = (1, u_2, \ldots, u_n)$ and so $g_u(x) = x_1 x_2^{u_2} \ldots x_n^{u_n}$. Therefore $g_u(0, x_2, \ldots, x_n) = 0$ and $g_u(1, x_2, \ldots, x_n) = x_2^{u_2} \ldots x_n^{u_n} = g_{u-2^{n-1}}(x_2, \ldots, x_n)$. It follows that $TT(x^u) = (0 \ldots 0|TT'(x^{u-2^{n-1}}))$. This proves that

$$A_n = \begin{pmatrix} A_{n-1} & 0 \\ A_{n-1} & A_{n-1} \end{pmatrix},$$

and the matrices A_n can be defined recursively as

$$A_0 = (1), \quad A_1 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad A_n = \begin{pmatrix} A_{n-1} & 0 \\ A_{n-1} & A_{n-1} \end{pmatrix} = A_1 \otimes A_{n-1} \text{ for } n \geq 2,$$

where $\otimes$ denotes the Kronecker product, and 0 is the zero matrix of a suitable size.

Example 5. Let $f(x_1, x_2, x_3) = x_1x_2x_3 \oplus x_2x_3 \oplus x_1 \oplus 1$. Then

$$[f] = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 1&1&0&0&0&0&0&0\\ 1&0&1&0&0&0&0&0\\ 1&1&1&1&0&0&0&0\\ 1&0&0&0&1&0&0&0\\ 1&1&0&0&1&1&0&0\\ 1&0&1&0&1&0&1&0\\ 1&1&1&1&1&1&1&1 \end{pmatrix} \cdot \begin{pmatrix} 1\\0\\0\\1\\1\\0\\0\\1 \end{pmatrix} = \begin{pmatrix} 1\\1\\1\\0\\0\\0\\0\\0 \end{pmatrix}.$$

Hence $TT(f) = (11100000)$.

The matrix A_n is a binary matrix of size $2^n \times 2^n$ with determinant 1 and so $A_n \in \mathrm{SL}(2^n, \mathbb{F}_2)$. It is easy to calculate that $A_n^2 = I_{2^n}$, therefore $A_n^{-1} = A_n$ and so $[A_f] = A_n[f]$. This equality allows us to find the algebraic normal form of a Boolean function presented by its Truth Table.

Corollary 2. If f is a Boolean function of n variables then $[A_f] = A_n[f]$.

In fact, the vector $[A_f]$ can be considered as a Truth Table of another Boolean function. The map μ which transforms the Boolean function f into the Boolean function $\mu(f)$ presented by the Truth Table $[A_f]$, is called binary Möbius transform (some authors use also Zhegalkin transform, Positive polarity Reed-Muller transform [1]). Obviously, μ is a bijection (permutation) in the set of all Boolean functions in n variables. The binary Möbius transform can be considered also as a permutation of the vectors in $\mathbb{F}_2^{2^n}$ given by the matrix A_n. More theoretical results on Möbius transforms are given in [10].

Studying the matrix A_n in details, we can see another very fast simple algorithm of type divide-and-conquer to compute the ANF from the Truth Table (and vice versa) called the Fast Möbius Transform. The algorithm can be illustrated by a butterfly diagram as shown in Example 6. For a Boolean function of n variables, the algorithm consists of n steps and in any step a vector $v^{(i)} = (v_0^{(i)}, v_1^{(i)}, \ldots, v_{2^n-1}^{(i)}) \in \mathbb{F}_2^{2^n}$ is obtained, $i = 0, 1, \ldots, n$, $v^{(0)} = (a_0, a_1, \ldots, a_{2^n-1})$, $v^{(n)} = TT(f)$. These vectors can be calculated recursively by the formula

$$v_{i.2^j+s}^{(j)} = v_{i.2^j+s}^{(j-1)}, \quad v_{i.2^j+s+2^{j-1}}^{(j)} = v_{i.2^j+s}^{(j-1)} \oplus v_{i.2^j+2^{j-1}+s}^{(j-1)}, \tag{3}$$

where $s = 0, 1, \ldots, 2^{j-1} - 1$, $i = 0, \ldots, 2^{n-j} - 1$, $j = 1, 2, \ldots, n$. For the first and the last steps we have

- $v^{(1)} = (a_0, a_0 \oplus a_1, a_2, a_2 \oplus a_3, \ldots, a_{2^n-2}, a_{2^n-2} \oplus a_{2^n-1})$;
- $v_i^{(n)} = v_i^{(n-1)}$, $v_{i+2^{n-1}}^{(n)} = v_i^{(n-1)} \oplus v_{i+2^{n-1}}^{(n-1)}$, $i = 0, \ldots, 2^{n-1} - 1$.

Example 6.

(x_1, x_2, x_3)	ANF		Step 1		Step 2		Step 3
000	a_0	$\longrightarrow$	a_0	$\longrightarrow$	a_0	$\longrightarrow$	a_0
001	a_1	$\searrow$	$a_0 \oplus a_1$	$\longrightarrow$	$a_0 \oplus a_1$	$\longrightarrow$	$a_0 \oplus a_1$
010	a_2	$\longrightarrow$	a_2	$\searrow$	$a_0 \oplus a_2$	$\longrightarrow$	$a_0 \oplus a_2$
011	a_3	$\searrow$	$a_2 \oplus a_3$	$\searrow$	$a_0 \oplus a_1 \oplus a_2 \oplus a_3$	$\longrightarrow$	$a_0 \oplus a_1 \oplus a_2 \oplus a_3$
100	a_4	$\longrightarrow$	a_4	$\longrightarrow$	a_4	$\searrow$	$a_0 \oplus a_4$
101	a_5	$\searrow$	$a_4 \oplus a_5$	$\longrightarrow$	$a_4 \oplus a_5$	$\searrow$	$a_0 \oplus a_1 \oplus a_4 \oplus a_5$
110	a_6	$\longrightarrow$	a_6	$\searrow$	$a_4 \oplus a_6$	$\searrow$	$a_0 \oplus a_2 \oplus a_4 \oplus a_6$
111	a_7	$\searrow$	$a_6 \oplus a_7$	$\searrow$	$a_4 \oplus a_5 \oplus a_6 \oplus a_7$	$\searrow$	$a_0 \oplus a_1 \oplus \cdots \oplus a_7$

Instead of $n + 1$ vectors we can use only one array v. In the beginning $v = v^{(0)}$. In any following step the array is partitioned into intervals (of length 2^i in the ith step), and the first half in the interval does not change, but the second half is Xored with the first half. The complexity of this algorithm is $O(n2^n)$. A more precise estimation is given in [1] where the basic time complexity is $\theta(n2^{n-1})$.

Algorithm 1: Fast Möbius Transform

Input: The Truth Table TT of the Boolean function f, with 2^n entries
Output: The Algebraic Normal Form ANF of the Boolean function f, with 2^n entries

```
size ← 1; ANF ← TT;
while (size < 2^n) do
    s ← size; size ← 2 * size;
    for pos from s − 1 by size to 2^n − 1 do
        for j from pos + 1 to pos + s do
            ANF[j] ← (ANF[j] ⊕ ANF[j − s])
        end for
    end for
end while.
```

3. Walsh transform. Associated with the Boolean function f is the function $(-1)^f = 1 - 2f$ whose function values belong to the set $\{-1; 1\}$. The

corresponding vector that contains the functions values of $(-1)^f$ is called the Polarity Truth Table (PTT) of the function f [3]. In the following, the $2^n \times 1$ column vector $[(-1)^f]$ represents the transpose of $PTT(f)$. We use PTT to obtain the Walsh spectrum of a Boolean function.

Definition 2. [3] Walsh (Walsh–Hadamard) transform f^W of the Boolean function f is the integer valued function $f^W : \mathbb{F}_2^n \to \mathbb{Z}$, defined by

$$f^W(a) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus \langle a,x \rangle}.$$

The Walsh transform of a Boolean function f is the Fourier transform of the signed function $(-1)^f$ (see [6]). The values of f^W are called Walsh coefficients. To understand Walsh coefficients, we calculate

$$f(x) \oplus \langle a,x \rangle = f(x_1, x_2, \ldots, x_n) \oplus a_1x_1 \oplus a_2x_2 \oplus \cdots \oplus a_nx_n = f(x) \oplus f_a(x),$$

where $f_a(x) = a_1x_1 \oplus a_2x_2 \oplus \cdots \oplus a_nx_n$. Hence

$$\begin{aligned} f^W(a) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus \langle a,x \rangle} &= \sum_{x \in \mathbb{F}_2^n, f(x) = f_a(x)} 1 + \sum_{x \in \mathbb{F}_2^n, f(x) \neq f_a(x)} (-1) \\ &= |\{x \in \mathbb{F}_2^n, f(x) = f_a(x)\}| - |\{x \in \mathbb{F}_2^n, f(x) \neq f_a(x)\}| \\ &= 2^n - 2d_H(f, f_a). \end{aligned}$$

For any Boolean function f and any vector $a \in \mathbb{F}_2^n$ we have $-2^n \leq f^W(a) \leq 2^n$. The functions $f_a(x) = \langle a,x \rangle$ and $\overline{f}_a(x) = \langle a,x \rangle \oplus 1$ have the maximal and minimal Walsh coefficients, namely $f_a^W(a) = 2^n$ and $\overline{f}_a^W(a) = -2^n$. Moreover, if $\overline{f}(x) = f(x) \oplus 1$ then

$$\overline{f}^W(a) = \sum_{x \in \mathbb{F}_2^n} (-1)^{\overline{f}(x) \oplus \langle a,x \rangle} = -\sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus \langle a,x \rangle} = -f^W(a).$$

Consider the vectors of $\mathbb{F}_2^n$ ordered lexicographically. Then we can order the Walsh coefficients $f^W(a)$ and consider them as coordinates of a vector. This vector is called Walsh spectrum of the Boolean function and denoted by $[W_f]$ (considered as a column). So $[W_f]^T = (f^W(\overline{0}), f^W(\overline{1}), \ldots, f^W(\overline{2^n - 1}))$. As in the previous section, we will present different methods to calculate the Walsh spectrum of a Boolean function.

Example 7. Let us calculate the Walsh spectrum of $f(x_1, x_2, x_3) = 1 \oplus x_1 \oplus x_2x_3 \oplus x_1x_2x_3$ in two ways. For the first one we use directly the definition.

a	$\langle a, x\rangle$	$f(x) \oplus \langle a, x\rangle$	$TT(f(x) \oplus \langle a, x\rangle)$	$f^W(a)$
000	0	$1 \oplus x_1 \oplus x_2x_3 \oplus x_1x_2x_3$	(11100000)	$-3+5=\ \ 2$
001	x_3	$1 \oplus x_1 \oplus x_3 \oplus x_2x_3 \oplus x_1x_2x_3$	(10110101)	$-5+3=-2$
010	x_2	$1 \oplus x_1 \oplus x_2 \oplus x_2x_3 \oplus x_1x_2x_3$	(11010011)	$-5+3=-2$
011	$x_2 \oplus x_3$	$1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_2x_3 \oplus x_1x_2x_3$	(10000110)	$-3+5=\ \ 2$
100	x_1	$1 \oplus x_2x_3 \oplus x_1x_2x_3$	(11101111)	$-7+1=-6$
101	$x_1 \oplus x_3$	$1 \oplus x_3 \oplus x_2x_3 \oplus x_1x_2x_3$	(10111010)	$-5+3=-2$
110	$x_1 \oplus x_2$	$1 \oplus x_2 \oplus x_2x_3 \oplus x_1x_2x_3$	(11011100)	$-5+3=-2$
111	$x_1 \oplus x_2 \oplus x_3$	$1 \oplus x_2 \oplus x_3 \oplus x_2x_3 \oplus x_1x_2x_3$	(10001001)	$-3+5=\ \ 2$

In the second way we calculate the Truth Tables of the functions $f_a(x)$ and the distances to $f(x)$.

a	$f_a(x) = \langle a, x\rangle$	$TT(f_a(x))$	$d_H(f, f_a)$	$f^W(a) = 8 - 2d_H(f, f_a)$
000	0	(00000000)	3	2
001	x_3	(01010101)	5	-2
010	x_2	(00110011)	5	-2
011	$x_2 \oplus x_3$	(01100110)	3	2
100	x_1	(00001111)	7	-6
101	$x_1 \oplus x_3$	(01011010)	5	-2
110	$x_1 \oplus x_2$	(00111100)	5	-2
111	$x_1 \oplus x_2 \oplus x_3$	(01101001)	3	2

In both ways we obtain $[W_f]^T = (2, -2, -2, 2, -6, -2, -2, 2)$.

The Walsh spectrum of a Boolean function measures its distance to the linear and affine functions. To understand this connection better, consider in more details the Truth Tables of the linear Boolean functions. Let S_n be the matrix whose columns are the Truth Tables of all linear Boolean functions ordered lexicographically. Then

$$S_n = ([\langle \overline{0}, x\rangle]\ [\langle \overline{1}, x\rangle]\ \cdots\ [\langle \overline{2^n-1}, x\rangle]) = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & \langle \overline{1}, \overline{1}\rangle & \cdots & \langle \overline{2^n-1}, \overline{1}\rangle \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \langle \overline{1}, \overline{2^n-1}\rangle & \cdots & \langle \overline{2^n-1}, \overline{2^n-1}\rangle \end{pmatrix}$$

Obviously, $S_n(i,j) = \langle \overline{i}, \overline{j}\rangle = \langle \overline{j}, \overline{i}\rangle = S_n(j,i)$, $0 \le i, j \le 2^n - 1$, which proves that this matrix is symmetric and therefore its rows are the Truth Tables of all linear functions. Making a connection with Coding Theory, we see that

actually this matrix consists of all codewords in the $[2^n, n, 2^{n-1}]$ binary simplex code with a zero coordinate added in the beginning of each codeword.

If we take $\langle \overline{u}, x\rangle$ instead of x^u for $u = 0, 1, \ldots, 2^n - 1$ and replace the column $[x^u]$ by $[\langle \overline{u}, x\rangle]$, the matrix A_n goes to S_n. So we can expect that as in the previous section, after some transformations we can have a more effective algorithm.

$$f^W(a) = \sum_{x\in\mathbb{F}_2^n} (-1)^{f(x)\oplus\langle a,x\rangle} = \sum_{x\in\mathbb{F}_2^n} (-1)^{\langle a,x\rangle}(-1)^{f(x)}$$

$$\Rightarrow [W_f] = \begin{pmatrix} f^W(\overline{0}) \\ f^W(\overline{1}) \\ \vdots \\ f^W(\overline{2^n-1}) \end{pmatrix} = \begin{pmatrix} \sum\limits_{x\in\mathbb{F}_2^n} (-1)^{f(x)} \\ \sum\limits_{x\in\mathbb{F}_2^n} (-1)^{\langle \overline{1},x\rangle}(-1)^{f(x)} \\ \vdots \\ \sum\limits_{x\in\mathbb{F}_2^n} (-1)^{\langle \overline{2^n-1},x\rangle}(-1)^{f(x)} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & (-1)^{\langle \overline{1},\overline{1}\rangle} & \cdots & (-1)^{\langle \overline{1},\overline{2^n-1}\rangle} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & (-1)^{\langle \overline{2^n-1},\overline{1}\rangle} & \cdots & (-1)^{\langle \overline{2^n-1},\overline{2^n-1}\rangle} \end{pmatrix} \begin{pmatrix} (-1)^{f(\overline{0})} \\ (-1)^{f(\overline{1})} \\ \vdots \\ (-1)^{f(\overline{2^n-1})} \end{pmatrix}$$

$$= H_n[(-1)^f],$$

where $H_n = ((-1)^{\langle \overline{i},\overline{j}\rangle})_{i,j}$ is a $2^n \times 2^n$ matrix. It is obvious that the matrix H_n can be obtained from S_n by replacing zeros with 1's and ones with -1's.

Taking in mind that the first coordinate of $\overline{i}$ is 0 if $i < 2^{n-1}$ and 1 otherwise, we have

$$\langle \overline{i}, \overline{j}\rangle = \begin{cases} \langle \overline{i}, \overline{j-2^{n-1}}\rangle, & \text{if } \ i < 2^{n-1},\ j \geq 2^{n-1} \\ \langle \overline{i-2^{n-1}}, \overline{j-2^{n-1}}\rangle, & \text{if } \ i \geq 2^{n-1},\ j < 2^{n-1} \\ 1 \oplus \langle \overline{i-2^{n-1}}, \overline{j-2^{n-1}}\rangle, & \text{if } \ i \geq 2^{n-1},\ j \geq 2^{n-1} \end{cases}$$

It follows that

$$H_n = \begin{pmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{pmatrix},$$

and the matrices H_n can be defined recursively as

$$H_0 = (1),\ \ H_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},\ \ H_n = \begin{pmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{pmatrix} = H_1 \otimes H_{n-1} \text{ for } n \geq 2.$$

The matrices H_n are Hadamard matrices of Sylvester type called also Sylvester matrices or Walsh matrices [5]. Using that H_n is symmetric and $H_n H_n^T = 2^n I_{2^n}$ we obtain that $H_n^{-1} = \frac{1}{2^n} H_n$.

Example 8. Let $TT(f) = (11100000)$. Then

$$[W_f] = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ -2 \\ -2 \\ 2 \\ -6 \\ -2 \\ -2 \\ 2 \end{pmatrix}.$$

To calculate the Walsh spectrum of a Boolean function, one can use also the Fast Walsh transform which is similar to Fast Möbius transform and the corresponding algorithm has the same complexity $O(n2^n)$. We use again a vector of length 2^n which we transform in n steps. In the beginning $v^{(0)} = [(-1)^f]$. In the i-th step the vector (the table) is partitioned into intervals of length 2^i, as the first half in any interval is the sum of the two halves and the second is their subtraction. The recurrent relation now is

$$(4) \qquad v^{(j)}_{i2^j+s} = v^{(j-1)}_{i2^j+s} + v^{(j-1)}_{i2^j+2^{j-1}+s}, \quad v^{(j)}_{i2^j+s+2^{j-1}} = v^{(j-1)}_{i2^j+s} - v^{(j-1)}_{i2^j+2^{j-1}+s},$$

whereby $s = 0, 1, \ldots, 2^{j-1} - 1$, $i = 0, \ldots, 2^{n-j} - 1$, $j = 1, 2, \ldots, n$, and $v^{(j)}$ is the vector obtained in the jth step. A pseudo code of the corresponding algorithm is given below. A somewhat different version is given in [2].

Example 9.

(x_1, x_2, x_3)	$[f]$		Step 1		Step 2		Step 3
000	t_0	↘	$t_0 + t_1$	↘	$t_0 + t_1 + t_2 + t_3$	↘	$t_0 + t_1 + t_2 + t_3 + t_4 + t_5 + t_6 + t_7$
001	t_1	↗	$t_0 - t_1$	↘	$t_0 - t_1 + t_2 - t_3$	↘	$t_0 - t_1 + t_2 - t_3 + t_4 - t_5 + t_6 - t_7$
010	t_2	↘	$t_2 + t_3$	↗	$t_0 + t_1 - t_2 - t_3$	↘	$t_0 + t_1 - t_2 - t_3 + t_4 + t_5 - t_6 - t_7$
011	t_3	↗	$t_2 - t_3$	↗	$t_0 - t_1 - t_2 + t_3$	↘	$t_0 - t_1 - t_2 + t_3 + t_4 - t_5 - t_6 + t_7$
100	t_4	↘	$t_4 + t_5$	↘	$t_4 + t_5 + t_6 + t_7$	↗	$t_0 + t_1 + t_2 + t_3 - t_4 - t_5 - t_6 - t_7$
101	t_5	↗	$t_4 - t_5$	↘	$t_4 - t_5 + t_6 - t_7$	↗	$t_0 - t_1 + t_2 - t_3 - t_4 + t_5 - t_6 + t_7$
110	t_6	↘	$t_6 + t_7$	↗	$t_4 + t_5 - t_6 - t_7$	↗	$t_0 + t_1 - t_2 - t_3 - t_4 - t_5 + t_6 + t_7$
111	t_7	↗	$t_6 - t_7$	↗	$t_4 - t_5 - t_6 + t_7$	↗	$t_0 - t_1 - t_2 + t_3 - t_4 + t_5 + t_6 - t_7$

Algorithm 2: Fast Walsh Transform

Input: The Polarity Truth Table PTT of the Boolean function f, with 2^n entries

Output: The Walsh spectrum W_f of the Boolean function f, with 2^n entries

```
size ← 1; W_f ← PTT;
while (size < 2^n) do
    s ← size; size ← 2 * size;
    for pos from s − 1 by size to 2^n − 1 do
        for j from pos + 1 to pos + s do
            temp ← −W_f[j] + W_f[j − s];
            W_f[j − s] ← W_f[j] + W_f[j − s];
            W_f[j] ← temp;
        end for
    end for
end while.
```

4. Linearity and nonlinearity of a Boolean function. The linearity of a Boolean function is defined via the Walsh transform as

Definition 3. [6] *Linearity* $Lin(f)$ of the Boolean function f is the maximum absolute value of an Walsh coefficient of f:

$$Lin(f) = \max\{|f^W(a)| \mid a \in \mathbb{F}_2^n\}.$$

For example, the linearity of the function f given by $TT(f) = (11100000)$ is 6 (see Example 8). It is very easy to calculate the linearity of a Boolean function if we have its Walsh spectrum. The maximum possible value of $Lin(f)$ is 2^n and is attained if and only if f is an affine function. Moreover, the Parseval's Equality $\sum_{a\in\mathbb{F}_2^n} (f^W(a))^2 = 2^{2n}$ gives that $Lin(f) \geq 2^{n/2}$ [6]. Functions attaining this lower bound are called bent functions.

Another important parameter which is closely connected with linearity is nonlinearity.

Definition 4. [6] *Nonlinearity* $nl(f)$ of the Boolean function f is the minimum Hamming distance from f to the nearest affine function:

$$nl(f) = \min\{d_H(f, g) \mid g \text{ – affine function}\}.$$

Example 10. Let $f(x_1, x_2, x_3) = x_1x_2x_3 \oplus x_2x_3 \oplus x_1 \oplus 1 \in \mathcal{F}_3$. Obviously $nl(f) \geq 1$, since f is not an affine function. On the other hand

$$d_H(f, x_1 \oplus 1) = \mathrm{wt}(f \oplus x_1 \oplus 1) = \mathrm{wt}(x_1x_2x_3 \oplus x_2x_3) = 1.$$

Hence $nl(f) = 1$.

The relation between the linearity and nonlinearity of the Boolean function f is given by the equality $Lin(f) = 2^n - 2nl(f)$ [6]. The minimum linearity corresponds to maximum nonlinearity.

The nonlinearity and even the Walsh spectrum of a Boolean function can be calculated using linear codes. Actually, the set of the Truth Tables of all affine Boolean functions coincides with the set of codewords of the Reed-Muller code of first order $RM(1, n)$. This is a linear $[2^n, n + 1, 2^{n-1}]$ code with a generator matrix

$$G(RM(1,n)) = \begin{pmatrix} TT(1) \\ TT(x_1) \\ TT(x_2) \\ \vdots \\ TT(x_n) \end{pmatrix},$$

and the codewords are the linear combinations of the rows of the matrix with coefficients from $\mathbb{F}_2$. The code $RM(1, n)$ is obtained from the simplex code by adding the all ones vector to its generator matrix. This means that besides all codewords from the matrix S_n, the Reed-Muller code $RM(1, n)$ contains their complements.

Example 11. The Reed-Muller code of first order of length 8 has a generator matrix

$$G(RM(1,3)) = \begin{pmatrix} 11111111 \\ 00001111 \\ 00110011 \\ 01010101 \end{pmatrix}.$$

Let C be a linear code of length n and $v \in \mathbb{F}_2^n$. The Hamming distance from v to the code is defined by $d_H(v, C) = \min\{d_H(v, x) \mid x \in C\}$. This definition and the structure of the Reed-Muller code show that the nonlinearity of the Boolean function $f \in \mathcal{F}_n$ is equal to the distance from its Truth Table $TT(f)$ to $RM(1, n)$, i. e., $nl(f) = d_H(TT(f), RM(1, n))$. This means that we can use algorithms for calculating the distance from a vector to a code (or for minimum distance of a linear code) to find the nonlinearity of a Boolean function without

having the Walsh spectrum. Some algorithms for calculation of linearity, nonlinearity and other important parameters of a Boolean function are given in [4].

5. Linearity and nonlinearity of vectorial Boolean functions.

The term *S-box* is most often used in cryptography, but is dedicated to the vectorial functions whose role is to introduce confusion into the system.

Definition 5. [7] Let n and m be two positive integers. The functions from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$ are called $(n;m)$-functions. When the numbers m and n are not specified, $(n;m)$-functions are called multi-output Boolean functions, vectorial Boolean functions or S-boxes.

A vectorial Boolean function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ can be represented by the vector $(f_1, f_2, \ldots, f_m)$, where f_i are Boolean functions of n variables, $i = 1, 2, \ldots, m$. The functions f_i are called the *coordinate functions* of the S-box. S-boxes play a fundamental role for the security of nearly all modern block ciphers. They have to be chosen carefully to make the cipher resistant against all kinds of attacks. In particular there are well studied criteria that a good S-box has to fulfill to make the cipher resistant against differential and linear cryptanalyses.

In order to study the cryptographic properties of an S-box related to the linearity, we need to consider all non-zero linear combinations of the coordinates of the S-box, denoted by $S_b = b \cdot F = b_1 f_1 \oplus \cdots \oplus b_m f_m$, where $b = (b_1, \ldots, b_m) \in \mathbb{F}_2^m$. These are the *component functions* of the S-box F. The *Walsh spectrum* of F is defined as the collection of all Walsh spectra of the component functions of F. The *linearity* and *nonlinearity* of F are defined as (see [7])

$$Lin(F) = \max_{b \in \mathbb{F}_2^m \setminus \{0\}} Lin(b \cdot F), \qquad nl(F) = \min_{b \in \mathbb{F}_2^m \setminus \{0\}} nl(b \cdot F).$$

Another important parameter related to S-boxes is the *algebraic degree* of the (n, m) S-box F. Some authors define it as $\deg(F) = \max_{b \in \mathbb{F}_2^m} \deg(b \cdot F)$ (see for example [3] and [7]), but others use as $\deg(F)$ the minimum among these degrees [8]. Therefore we define a maximal and a minimal algebraic degree of the vectorial Boolean function F as:

$$\deg_{\max}(F) = \max_{b \in \mathbb{F}_2^m} \deg(b \cdot F) = \max\{\deg(f_1), \ldots, \deg(f_m)\},$$

$$\deg_{\min}(F) = \min_{b \in \mathbb{F}_2^m \setminus \{0\}} \deg(b \cdot F).$$

Algorithm 3: Minimal Algebraic Degree of an S-box

Input: The $m \times 2^n$ array $SAlg$ representing the S-box
Output: The minimal algebraic degree $degmin$ of the S-box
for i from 1 to m do $t[i] \leftarrow i + 1$;
for j from 0 to $2^n - 1$ do $Ac[j] \leftarrow 0$ end for;
$i \leftarrow 1$; $degmin \leftarrow n$;
while $(i \neq m + 1)$ do
  for j from 0 to $2^n - 1$ do
    $Ac[j] \leftarrow Ac[j] \oplus SAlg[i][j]$;
  end for;
  $degAc \leftarrow 0$; $j \leftarrow 0$;
  for j from 0 to $2^n - 1$ do
    if $(Ac[Ord\_deg[j]] = 1)$ then $degAc \leftarrow Weights\_deg[j]$; break end if;
  end for;
  if $(0 < degAc < degmin)$ then $degmin \leftarrow degAc$ end if;
  end for;
  $t[0] \leftarrow 1$; $t[i-1] = t[i]$; $t[i] \leftarrow i + 1$;
  $i = t[0]$;
end while.

It is easy to calculate the maximal algebraic degree as it is the maximum among the degrees of the coordinate functions. To calculate the linearity and the minimal algebraic degree of an S-box, we need a fast algorithm for generating all non-zero linear combinations of given binary vectors. To do this we use the binary Gray code (algorithms implementing the binary Gray code are given in [11]).

Algorithm 3 calculates the minimal algebraic degree of an S-box given by the matrix $SAlg$ of size $m \times 2^n$ where the rows are the algebraic normal forms of the coordinate Boolean functions presented by their vectors of coefficients. The array t shows how the Gray code works. The vector Ac is the algebraic normal form of the current component function and $degAc$ is its algebraic degree. We use also two auxiliary arrays Ord_deg and $Weights\_deg$ with 2^n entries each. The array Ord_deg consists of the integers $0, 1, \ldots, 2^n - 1$ but ordered so that $\mathrm{wt}(\overline{Ord\_deg[i]}) \geq \mathrm{wt}(\overline{Ord\_deg[i+1]})$, $i = 0, 1, \ldots, 2^n - 2$, and the array $Weights\_deg$ consists of the corresponding weights, namely $Weights\_deg[i] = \mathrm{wt}(\overline{Ord\_deg[i]})$, $i = 0, 1, \ldots, 2^n - 1$.

In the generation of the auxiliary arrays Ord_deg and $Weights\_deg$ we use a recursive algorithm. Algorithm 4 shows how we could generate these two arrays.

Algorithm 4: Generation of the auxiliary arrays.

Input: The integer n.
Output: The arrays Ord_deg and $Weights\_deg$ with 2^n entries.

```
Function Gen_Ord_deg(lc, level, ind, j);
if level = lc then
    count ← count + 1;
    Ord_deg[count] ← ind;
    Weights_deg[count] ← n − level;
    Return;
end if;
for i from j to n − 1 do
    indh ← ind ⊕ 2^i;
    if (level < lc) then Gen_Ord_deg(lc, level + 1, indh, i + 1) end if;
end for;
end Gen_Ord_deg;
Main Function
  count ← −1; ind ← 2^n − 1;
  for lc from 0 to n − 1 do Gen_Ord_deg(lc, 0, ind, 0) end for;
  Ord_deg[count + 1] ← 0;
  Weights_deg[count + 1] ← 0.
end Main.
```

Example 12. For $n = 3$ we have

$$Ord\_deg = (7, 6, 5, 3, 4, 2, 1, 0), \quad Weights\_deg = (3, 2, 2, 2, 1, 1, 1, 0).$$

Algorithm 5 calculates the linearity of an S-box presented by an $m \times 2^n$ matrix whose rows are the Truth Tables of the coordinate Boolean functions. The vectors TT and PTT in the algorithm are the Truth Table and Polarity Truth Table of the current component function.

6. Conclusions. We presented five algorithms for calculating some of the parameters of Boolean functions and vectorial Boolean functions that are most important in cryptography. The first two algorithms implement the fast Möbius and fast Walsh transforms for Boolean functions. Algorithms 3 and 5 compute the algebraic degree and linearity of a vectorial Boolean function (S-box). These algorithms are useful for researchers studying the design of block ciphers, especially for PhD students.

Algorithm 5: Linearity of an S-box

Input: STT—an $m \times 2^n$ matrix whose rows are the Truth Tables of the coordinate functions
Output: The linearity Lin of the S-box
for i from 1 to m do $t[i] \leftarrow i + 1$;
for j from 0 to $2^n - 1$ do $TT[j] \leftarrow 0$ end for;
$i \leftarrow 1$; $Lin \leftarrow 0$;
while $(i \neq m + 1)$ do
  for j from 0 to $2^n - 1$ do
    $TT[j] \leftarrow TT[j] \oplus STT[i][j]$;
    if $(TT[j] = 1)$ then $PTT[j] \leftarrow -1$ else $PTT[j] \leftarrow 1$ end if;
  end for;
  FastWalshTransform(PTT, W_f);
  for j from 0 to $2^n - 1$ do
    if $(Abs(W_f[j]) > Lin)$ then $Lin \leftarrow Abs(W_f[j])$ end if;
  end for;
  $t[0] \leftarrow 1$; $t[i-1] = t[i]$; $t[i] \leftarrow i + 1$;
  $i = t[0]$;
end while.

REFERENCES

[1] Bakoev V., K. Manev. Fast computing of the positive polarity Reed-Muller transform over GF(2) and GF(3). In: Proc. of the XI Intern. Workshop ACCT, Pamporovo, Bulgaria, 2008, 13–21.

[2] Bouyukliev I., D. Bikov. Applications of the binary representation of integers in algorithms for boolean functions. In: Mathematics and education in mathematics, Proceedings of the Forty Fourth Spring Conference of the Union of Bulgarian Mathematicians, 2015, 161–166.

[3] Braeken A. Cryptographic Properties of Boolean Functions and S-Boxes. PhD thesis, Katholieke Universiteit Leuven, Belgium, 2006.

[4] Çalik Ç. Computing Cryptographic Properties of Boolean Functions from the Algebraic Normal Form Representation. PhD thesis, Middle East Technical University, Ankara, Turkey, 2013.

[5] Cameron P. Encyclopaedia of Design Theory, 2004. `http://designtheory.org/library/encyc/`, 23 October 2017.

[6] Carlet C. Boolean Functions for Cryptography and Error Correcting Codes. In: C. Crama, P. L. Hammer (eds). Boolean Models and Methods in Mathematics, Computer Science, and Engineering. Cambridge University Press, 2010, 257–397.

[7] Carlet C. Vectorial Boolean Functions for Cryptography. In: C. Crama, P. L. Hammer (eds). Boolean Models and Methods in Mathematics, Computer Science, and Engineering. Cambridge University Press, 2010, 398–469.

[8] Fuller J. Analysis of affine equivalent Boolean functions for cryptography. PhD thesis, Queensland University of Technology, Australia, 2003.

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

[10] Pieprzyk J., H. Wang, X.-M. Zhang. Möbius transforms, coincident Boolean functions and non-coincidence property of Boolean functions. *Int. J. Comput. Math.*, **88** (2011), No 7, 1398–1416.

[11] Reingold E. M., J. Nievergelt, N. Deo. Combinatorial Algorithms: Theory and Practice. Prentice-Hall, Englewood Cliffs, NJ, 1977.

[12] Zhegalkin I. I. On the Technique of Calculating Propositions in Symbolic Logic. *Matematicheskii Sbornik*, **43** (1927), 9–28.

Stefka Bouyuklieva
Faculty of Mathematics and Informatics
Veliko Tarnovo University
Veliko Tarnovo, Bulgaria
e-mail: stefka@uni-vt.bg

Iliya Bouyukliev
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
P. O. Box 323
Veliko Tarnovo, Bulgaria
e-mail: iliyab@math.bas.bg

Received July 17, 2016
Final Accepted February 6, 2017