ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 6, June 2013

ALGORITHM TO DETERMINE THE FIXED DEGREE POLYNOMIAL OF BOOLEAN FUNCTION FOR CRYPTOGRAPHY

Nguyen Huu Khanh Nhan

Lecturer, Dept. of Electrical and Electronic Engineering, Ton Duc Thang University, Ho Chi Minh City, Vietnam

ABSTRACT: Every Boolean function is uniquely defined by a polynomial modulo 2. The degree of a Boolean function is the degree of its defining polynomial. In cryptography, the Boolean functions of fixed degree played important role, for example, 1 or 2 degrees. Therefore, in finding algorithms that recognize properties of Boolean functions polynomials by their values vectors, it makes sense consider only algorithms that have lower complexity order. In this paper, we propose a linear complexity algorithm which determines the vector values a Boolean function given, it is a polynomial of fixed degree, and if so constructing this polynomial.

Keywords: Boolean function, fixed degree polynomial, algebraic normal form, numerical normal form.

I. INTRODUCTION

Vectorial Boolean Functions for Cryptography which follows, the set {0, 1} will be most often endowed with the structure of field (and denoted by F_2), and the set F_2^n of all binary vectors of length *n* will be viewed as an F_2-vectorspace. We shall denote simply by 0 the null vector in F_2^n. The vectorspace F_2^n will sometimes be also endowed with the structure of field { the field F_2^n (also denoted by $GF(2^n)$) [1]; indeed, this field being an *n*-dimensional vectorspace over F_2, each of its elements can be identified with the binary vector of length *n* of its coordinates relative to a fixed basis. The set of all Boolean functions f: F_2^n -> F_2 will be denoted as usual by BF_n. The Hamming weight $w_H(x)$ of a binary vector $x \in \mathrm{F}_2^n$ being the number of its nonzero coordinates, the Hamming weight $w_H(f)$ of a Boolean function *f* on F_2^n is the size of the support of the function , i.e. the set $\{x \in \mathrm{F}_2^n \,/\, f(x) \neq 0\}$. The Hamming distance $d_H(f, g)$ between two functions *f* and *g* is the size of the set $\{x \in \mathrm{F}_2^n \,/\, f(x) \neq g(x)\}$. Thus it equals $w_H(f \oplus g)$.

Some additions of bits will be considered in $\mathbf{Z}$ (in characteristic 0) and denoted then by +, and sometimes will be computed modulo 2 and denoted by $\oplus$. These two dierent notations will be necessary because some representations of Boolean functions will live in characteristic 2 and some representations of the same functions will live in characteristic 0. But the additions of elements of the finite field F_{2^n} will be denoted by +, as it is usual in mathematics. So, for simplicity (since F_2^n will often be identified with F_{2^n}) and because there will be no ambiguity, we shall also denote by + the addition of vectors of F_2^n when $n > 1$.

Many constructions of Boolean functions with properties relevant to cryptography are recursive [2, 4, 5,]. The efficiency of the constructions relies heavily on the use of appropriate functions of small dimensions. Another important method for construction is the random and heuristic search approach [2, 3]. As equivalence classes are used to provide restricted input of such optimization algorithms, it is very important to identify which equivalence classes obtain functions with desired properties. Methods bounding the degree of polynomial that present boolean functions have been important tools in complexity theory. These techniques have been uesd to obtain several results that shed light on the complexity of boolean functions. In particular, such polynomial degree lower bounds consequences for the constant-depth circuit complexity of the associated boolean functions.

Let the Boolean function written vector of its values with $N = 2^n$ coordinates, where *n* - number of variables of the function. It is known that the values vector of a Boolean function of a polynomial can be found with complexity *O (N logN)* [7]. Therefore, in finding algorithms that recognize properties of Boolean functions polynomials by their values vectors, it makes sense consider only algorithms that have lower order of complexity. In this paper we propose a linear

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 6, June 2013

complexity algorithm which determines the vector values a Boolean function given, it is a polynomial of fixed degree, and if so constructing this polynomial.

II. REPRESENTATION OF BOOLEAN FUNCTIONS

Among the classical representations of Boolean functions, the one which is most usually used in cryptography and coding is the n-variable polynomial representation over F_2, of the form:

$$f(x) = \bigoplus_{I \in P(N)} a_I \left(\prod_{i \in I} x_i \right) = \bigoplus_{I \in P(N)} a_I x^I , \qquad (1)$$

Where *P(N)* denotes the power set of $N = \{1, ..., n\}$. Every coordinate x_i appears in this polynomial with exponents at most 1, because every bit in F_2 equals its own square. This representation belongs to $F_2[x_1,...,x_n]/(x_1^2 \oplus x_1,...,x_n^2 \oplus x_n)$. It is called the *Algebraic Normal Form* (in brief the ANF).

Another possible representation of this same ANF uses an indexation by means of vectors of F_2^n instead of subsets of *N*; for any such vector *u*, we denote by a_u what is denoted by $a_{supp(u)}$ in relation (1) (where *supp(u)* denotes the support of *u*), we have the equivalent representation:

$$f(x) = \bigoplus_{u \in \mathbb{F}_2^n} a_u \left(\prod_{j=1}^{n} x_j{}^{u_j} \right) . \qquad (2)$$

The monomial $\prod_{j=1}^{n} x_j^{u_i}$ is often denoted by x^u.

2.1. Relationship between a Boolean function and its ANF.

The product $x^I = \prod_{i \in I} x_i$ is nonzero if and only if x_i is nonzero (i.e. equals 1) for every $i \in I$, that is, if *I* is included in the support of *x*; hence, the Boolean function $f(x) = \oplus_{I \in P(N)} a_I x^I$ takes value.

$$f(x) = \bigoplus_{I \subseteq supp(x)} a_I , \qquad (3)$$

where *supp(x)* denotes the support of *x*. If we use the notation $f(x) = \oplus_{u \in F_2^n} a_u x^u$, we obtain the relation $f(x) = \oplus_{u \le x} a_u$, where $u \le x$ means that *supp(u)* $\subseteq$ *supp(x)* (we say that *u* is covered by *x*). A Boolean function f^0 can be associated to the ANF of *f*: for every $x \in F_2^n$, we set $f^0(x) = a_{supp(x)}$, that is, with the notation $f(x) = \oplus_{u \in F_2^n} a_u x^u : f^0(u) = a^u$. Relation (3) shows that *f* is the image of f^0 by the so-called *binary Mobius Transform*.

The converse is also true:

Proposition 1: *Let f be a Boolean function on* F_2^n *and let* $\oplus_{I \in P(N)} a_I x^I$ *be its ANF. We have:*

$$\forall I \in P(N), a_I = \bigoplus_{x \in F_2^n / supp(x) \subseteq I} f(x). \qquad (4)$$

Proof. Let us denote $\bigoplus_{x \in F_2^n / supp(x) \subseteq I} f(x).$ by b_I and consider the function $g(x) = \oplus_{I \in P(N)} b_I x^I$. We have

$$g(x) = \bigoplus_{I \subseteq supp(x)} b_I = \bigoplus_{I \subseteq supp(x)} \left(\bigoplus_{y \in \mathbb{F}_2^n / \, supp(y) \subseteq I} f(y) \right)$$

and thus

$$g(x) = \bigoplus_{y \in \mathbb{F}_2^n} f(y) \left(\bigoplus_{I \in \mathcal{P}(N) / \, supp(y) \subseteq I \subseteq supp(x)} 1 \right) .$$

The sum $\bigoplus_{I\in P(N)/supp(y)\subseteq I\subseteq supp(x)} 1$ is null if $y \neq x$, since the set $\{I \in P(N)/\operatorname{supp}(y) \subseteq I \subseteq \operatorname{supp}(x)\}$ contains $2^{w_H(x)-w_H(y)}$ elements if *supp(y)* $\subseteq$ *supp(x)*, and none otherwise. Hence, *g* = *f* and, by uniqueness of the ANF, b_I = a_I for every *I*.

2.2. The degree of the ANF.

It is denoted by $d^0 f$ and is called the *algebraic degree* of the function (this makes sense thanks to the existence and uniqueness of the ANF): $d^0 f = \max\{|I|/a_I \neq 0\}$, where |*I*| denotes the size of *I*. Some authors also call it the nonlinear order of *f*. According to Relation (4), we have:

Proposition 2: *The algebraic degree $d^0 f$ of any n-variable Boolean function f equals the maximum dimension of the subspaces $\{x \in F_2^n / supp(x) \subseteq I\}$ on which f takes value 1 an odd number of times.*

The algebraic degree is an affine invariant (it is invariant under the action of the general affine group): for every affine isomorphism *L*:

$$\begin{pmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{pmatrix} \in F_2^n \mapsto M \times \begin{pmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{pmatrix} \oplus \begin{pmatrix} a_1 \\ a_2 \\ . \\ . \\ a_n \end{pmatrix} \in F_2^n$$

(where *M* is a nonsingular *n x n* matrix over F_2). We have $d^0(f\ o\ L) = d^0 f$. Indeed, the composition by *L* clearly cannot increase the algebraic degree, since the coordinates of *L(x)* have degree 1. Hence we have $d^0(f\ o\ L) \leq d^0 f$ (this inequality is more generally valid for every affine homomorphism). And applying this inequality to *f o L* in the place of *f* and to L^{-1} in the place of *L* hows the inverse inequality. Two functions *f* and *f o L* where *L* is an F_2 - linear automorphism of F_2^n (in the case $a_1 = a_2 = \ldots = a_n = 0$ above) will be called linearly equivalent and two functions *f* and *f o L*, where *L* is an affine automorphism of F_2^n, will be called *affinely equivalent*.

The algebraic degree being an affine invariant, Proposition 2 implies that it also equals the maximum dimension of all the affine subspaces of F_2^n on which *f* takes value 1 an odd number of times.

III. THE REPRESENTATION OVER THE REALS

Boolean function has proved itself to be useful for characterizing several cryptographic criteria [8, 9]. It represents Boolean functions, and more generally real-valued functions on F_2^n (that are called *n-variable* pseudo-Boolean functions) by elements of $F_2^n[x_1,\ldots,x_n]/(x_1^2 - x_1,\ldots,x_n^2 - x_n)$ (or of $\mathbf{Z}[x_1,\ldots,x_n]/(x_1^2 - x_1,\ldots,x_n^2 - x_n)$ for integer-valued functions). We shall call it the *Numerical Normal Form* (NNF).

The existence of this representation for every pseudo-Boolean function is easy to show with the same arguments as for the ANFs of Boolean functions (writing $1 - x_i$ instead of $1 \oplus x_i$). The linear mapping from every element of the 2^n-*th* dimensional F_2^n - vectorspace $F_2^n[x_1,\ldots,x_n]/(x_1^2 - x_1,\ldots,x_n^2 - x_n)$ to the corresponding pseudo-Boolean function on F_2^n, it is therefore one to one (the F_2^n - vectorspace of pseudo-Boolean functions on F_2^n having also dimension 2^n). We deduce the uniqueness of the NNF.

We call the degree of the NNF of a function its numerical degree. Since the ANF is the *mod 2* version of the NNF, the numerical degree is always bounded below by the algebraic degree. It is shown in [4] that, if a Boolean function *f* has no ineffective variable (i.e. if it actually depends on each of its variables), then the numerical degree of *f* is greater than or equal to $\log_2 n - \log_2 \log_2 n$.

The numerical degree is not an affine invariant. But the NNF leads to an affine invariant (see a proof of this fact in [8]; see also [10]) which is more discriminant than the algebraic degree:

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 6, June 2013

Denition 1: *Let f be a Boolean function on* $\mathbb{F}_2^n$. *We call generalized degree of f the sequence* $(d_i)_{i\geq 1}$ *defined as follows: for every* $i \geq 1$, d_i *is the smallest integer* $d_i > d_{i-1}$ *(if* $i > 1$*) such that, for every multi-index I of size strictly greater than d, the coefficient* λ_I *of* x^I *in the NNF of f is a multiple of* 2^i.

Example: the generalized degree of any nonzero ane function is the sequence of all positive integers.

Similarly as for the ANF, a (pseudo-) Boolean function $f(x) = \sum_{I\in P(N)} \lambda_I x^I$ takes value:

$$f(x) = \sum_{I \subseteq supp(x)} \lambda_I. \tag{5}$$

But, contrary to what we observed for the ANF, the reverse formula is not identical to the direct formula:

Proposition 3: *Let f be a pseudo-Boolean function on* $\mathbb{F}_2^n$ *and let its NNF be* $\sum_{I\in P(N)} \lambda_I x^I$. *Then:*

$$\forall I \in P(N),$$
$$\lambda_I = (-1)^{|I|}.\sum_{x\in \mathbb{F}_2^n | supp(x)\subseteq I} (-1)^{w_H(x)} f(x). \tag{6}$$

Thus, function *f* and its NNF are related through the Mobius transform over integers.

Proof. Let us denote the number $(-1)^{|I|}.\sum_{x\in \mathbb{F}_2^n | supp(x)\subseteq I} (-1)^{w_H(x)} f(x)$ by μ_I and consider the function $g(x) = \sum_{I\in P(N)} \mu_I x^I$. We have

$$g(x) = \sum_{I\subseteq supp(x)} \mu_I =$$
$$= \sum_{I\subseteq supp(x)} \left((-1)^I \sum_{y\in \mathbb{F}_2^n | supp(y)\subseteq I} (-1)^{w_H(y)} f(y) \right)$$

and thus

$$g(x) = \sum_{y\in \mathbb{F}_2^n} (-1)^{w_H(y)} f(y) \times$$
$$\times \left(\sum_{I\in P(N)/supp(y)\subseteq I\subseteq supp(x)} (-1)^{|I|} \right)$$

The sum $\sum_{I\in P(N)/supp(y)\subseteq I\subseteq supp(x)} (-1)^{|I|}$ is null if $supp(y) \not\subseteq supp(x)$. It is also null if *supp(y)* is included in *supp(x)*, but different. Indeed, denoting $|I| - w_H(y)$ by *i*, it equals

$$\pm \sum_{i=0}^{w_H(x)-w_H(y)} \binom{w_H(x) - w_H(y)}{i} (-1)^i =$$
$$= \pm (1-1)^{w_H(x)-w_H(y)} = 0$$

Hence, $g = f$ and, by uniqueness of the NNF, we have $\mu_I = \lambda_I$ for every *I*.

Proposition 4: *Any polynomial* $P \in \mathbb{F}_2^n[x_1,...,x_n]/(x_1^2 - x_1,...,x_n^2 - x_n)$ *is the NNF of an integer-valued function if and only if all of its coefficients are integers. Assuming that this condition is satisffied, P is the NNF of a Boolean function if and only if:* $\sum_{x\in \mathbb{F}_2^n} P^2(x) = \sum_{x\in \mathbb{F}_2^n} P(x)$.

Proof. The first assertion is a direct consequence of relations (5) and (6). If all the coefficients of *P* are integers, then we have $P^2(x) \geq P(x)$ for every *x*; this implies that the 2^n equalities, expressing that the corresponding function is Boolean, can be reduced to the single one $\sum_{x\in F_2^n} P^2(x) = \sum_{x\in F_2^n} P(x)$

IV. APPROACH AND CONSTRUCT ALGORITHM.

Let $B = \{0, 1\}$, V^n - set of vectors of length n with the coordinates of the set B. Boolean function is a mapping.

$$f^n : V^n \to B, \qquad n = 0, 1, 2, \dots$$

The set of all Boolean functions depending on n variables, denoted as F_n.

Monotone elementary conjunction is called the product of distinct variables without negations. The rank of a monotone elementary conjunction is the number of variables. We assume a degenerate monotone elementary conjunctions of rank 0. Every Boolean function can be written uniquely form $\sum_{i=1}^{l} X_i$, where X_i - distinct monotone elementary conjunctions, the summation is mod 2. The degree of the polynomial is the greatest rank of its terms.

We say that a function $f(x_1, \dots, x_n)$ belongs to the class C_m, $m = 0, 1, 2, \dots$, if it is given a polynomial of degree m. Obviously, the class C_0 only contains constants 0 and 1, class C_1 is the class L of linear functions.

We call the derivative of $f(x_1, \dots, x_n)$ with respect to x_i function $f_{xi}(x_1, \dots, x_n)$, equal

$$f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) \oplus f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n).$$

Theorem 1. *The function $f(x_1, \dots, x_n)$ belongs to the class C_m, $m > 1$, if and only if for each variable x_i, $i = 1, \dots, n$, function $f_{xi}(x_1, \dots, x_n)$ belongs to C_{m-1}.*

Proof. Follows from the definition of a derivative and a unique representation of a Boolean function by a polynomial.

Let the Boolean function $f(x_1, \dots, x_n)$ is given a vector of its values α_f on the sets listed in lexicographic order. The number of coordinates α_f equal to $N = 2^n$.

Theorem 2. *There exists an algorithm to the vector α_f with complexity $O(N)$ determines whether the function $f(x_1, \dots, x_n)$ be linear, and, if so, it builds a polynomial.*

Proof. Consider the following algorithm.

```
$i := 1$;
$f_i(x_i, \dots, x_n) := f(x_1, \dots, x_n)$;
$p(x_i, \dots, x_n) := f(0, \dots, 0)$.
Beginning of the cycle.
1. Building a derivative $f_{i\,xi}(x_i, \dots, x_n)$: divide the vector $\alpha_{f\,i}$ in two and summarize the coordinate-wise. At the same time will be spent $2^{n-i+1}$ operations.
2. if
  • $f_{xi}(x_i, \dots, x_n) \equiv 1$, then $p(x_1, \dots, x_n) := p(x_i, \dots, x_n) \oplus x_i$;
  • $f_{xi}(x_i, \dots, x_n) \equiv 0$, then $p(x_1, \dots, x_n)$ remain unchanged;
  • Otherwise - the algorithm terminates and the answer is "nonlinear function - linearly.
3. $i := i + 1$, $f_i(x_i, \dots, x_n) = f_{i-1}(0, x_i, \dots, x_n)$,. To construct a vector $\alpha_{f\,i}$ requires $2^{n-i+1}$ operations.
4. if
  • $i > n$, then the algorithm terminates, the answer is "linear function" and it's written in the polynomial $p(x_1, \dots, x_n)$;
  • Otherwise - go to the top of the loop.
```

The correctness of the algorithm follows from Theorem 1.

We calculate the complexity of the algorithm. it is

$$2^n + 2^{n-1} + \dots + 1 \leqslant 2 \cdot 2^n = O(N).$$

Theorem 3. *Given a number $m > 1$. There exists an algorithm for vector α_f complexity $O(N)$ determines whether an function $f(x_1, \dots, x_n)$ to the class C_m, and, if so, it builds a polynomial.*

Proof. We construct the desired algorithm A_m inductively. The basis of induction. Let $m = 1$. Then, as the algorithm A_1 will consider the algorithm described in Theorem 2. Its complexity is equal to $2 \,.\, 2^n$.

Inductive step. Suppose that the algorithm A_{m-1} has already constructed, and its complexity is equal to $C_{m-1}.\ 2^n$, where C_{m-1} - is a constant. Consider an algorithm A_m is as the following:

```
$i := 1$;
$f_i(x_i, \dots, x_n) := f(x_1, \dots, x_n)$;
$p(x_i, \dots, x_n) := f(0, \dots, 0)$.
Beginning of the cycle.
1. Building a derivative $f_{i\,xi}(x_i, \dots, x_n)$: divide the vector $\alpha_{f\,i}$ in two and summarize the coordinate-wise. At the same time will be spent $2^{n-i+1}$ operations.
2. With algorithm $A_{m-1}$ checks whether the function $f_{xi}(x_i, \dots, x_n)$ class $C_{m-1}$.
if
  • Answer "yes" and a polynomial $p(x_1, \dots, x_n)$, then
    $P(x_1, \dots, x_n) := P(x_1, \dots, x_n) \oplus x_i . p(x_1, \dots, x_n)$.
```

• Otherwise, the algorithm terminates and the answer is "function does not belong to C_m”.
In Step 2, we have spent $C_{m-1}.2^{n-i+1}$ operations.
3. $i: = i + 1, f_i (x_i, ..., x_n) = f_{i-1} (0, x_i, ..., x_n)$. For constructing vector $\alpha_{f\,i}$ requires 2^{n-i+1} operations.
4. if
• $i > n$, then the algorithm terminates, the answer is "The function of class C_m” and it’s written in the polynomial $P(x_1, ..., x_n)$;
• Otherwise - go to the top of the loop. The correctness of the algorithm follows from Theorem 1. We calculate the complexity of the algorithm.

It is $(C_{m-1}+1).(2^n+2^{n-1}+...+1)\le 2.(C_{m-1}+1).2^n = C_m.2^n$ where C_m - a constant. So $C_m = 2.(C_{m-1}+1), C_1 = 2$. It is easy to calculate that $C_m = 2.2^m$. Consequently, the complexity of the algorithm is $(2.2^m).2^n = O(N)$, since the number m - fixed.

V. CONCLUSION.

In this paper we have shown that many classical notions, constructs algorithms that recognize properties of Boolean functions polynomials by their values vectors, it makes sense consider only algorithms that have lower complexity order. Linear complexity algorithm which determines the vector values a Boolean function given, it is a polynomial of fixed degree. These results are applied for the theory of cryptographic properties of Boolean functions.

REFERENCES

[1] Claude Carlet, “Boolean Functions for Cryptography and Error Correcting Codes”, Lecture notes, University of Paris 8.
[2] Y.V. Taranikov, “On Resilient Functions with Maximum Possible Nonlinearity”, Indocrypt 2000, LNCS 1977, Springer-Verlag, pp. 19-30, 2000.
[3] W. Millan, A. Clark, E. Dawson, “Heuristic Design of Cryptographically Strong Balanced Boolean Functions”, Eurocrypt 98, LNCS 1403, Springer-Verlag, pp. 489-499, 1998.
[4] S. Maitra, P. Sarkar, “Construction of Nonlinear Boolean Functions with Important Cryptographic Properties”, Eurocrypt 2000, LNCS 1807, Springer-Verlag, pp. 485-506, 2000.
[5] P. Stanica, S.H. Sung, “Boolean Functions with Five Controllable Cryptogrpahic Properties, Designs, Codes and Cryptography”, Vol. 31, pp. 147-157, 2004.
[6] Mart De Graaf and Paul Valiant, “Polynomial representations of symmetric partial boolean functions”, Journal on Discrete Mathematics, Volume 19 Issue 2, pp. 481 – 488, 2005.
[7] C. Carlet, “On the coset weight divisibility and nonlinearity of resilient and correlation-immune functions”, Proceedings of SETA'01 (Sequences and their Applications 2001), Discrete Mathematics and Theoretical Computer Science, 2001.
[8] C. Carlet and P. Guillot, “A new representation of Boolean functions, Proceedings of AAECC'13”, Lecture Notes in Computer Science 1719, 1999.
[9] C. Carlet and P. Guillot. Bent, “Resilient functions and the Numerical Normal Form”. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 56, 2001.
[10] An Braeken, Ventzislav Nikov, Svetla Nikova, and Bart Prenee, “On Boolean Functions with Generalized Cryptographic Properties”, Progress in Cryptology - INDOCRYPT 2004, Lecture Notes in Computer Science Volume 3348, pp. 120-135, 2005.