Finding Low Degree Annihilators for a Boolean Function Using Polynomial Algorithms

Vladimir Bayev *

Abstract. Low degree annihilators for Boolean functions are of great interest in cryptology because of algebraic attacks on LFSR-based stream ciphers. Several polynomial algorithms for construction of low degree annihilators are introduced in this paper. The existence of such algorithms is studied for the following forms of the function representation: algebraic normal form (ANF), disjunctive normal form (DNF), conjunctive normal form (CNF), and arbitrary formula with the Boolean operations of negation, conjunction, and disjunction. For ANF and DNF of a Boolean function f there exist polynomial algorithms that find the vector space $A_d(f)$ of all annihilators of degree $\leqslant d$. For CNF this problem is NP-hard. Nevertheless author introduces one polynomial algorithm that constructs some subspace of $A_d(f)$ having formula that represents f.

Keywords. Boolean function, low degree annihilator, polynomial algorithm, recursive algorithm.

Algebraic immunity is an important cryptographic characteristic of a Boolean function. Low algebraic immunity of a function means that this function has an annihilating multiplier of low algebraic degree. The problem of annihilator seeking was initially discussed in [3] and [5].

Let $\mathbb{F}_2$ be the field of two elements, $V_n = \mathbb{F}_2^n$ be the vector space of n-tuples over $\mathbb{F}_2$, $\mathcal{F}_n$ be the set of all functions $\mathbb{F}_2^n \to \mathbb{F}_2$. By $\deg f$ denote algebraic degree of a Boolean function $f \in \mathcal{F}_n$. A Boolean function $g \in \mathcal{F}_n$ is called an annihilator of $f \in \mathcal{F}_n$ if $f \cdot g = 0$. We shall use the following notation:

$$A_d(f) := \{g \in \mathcal{F}_n | f \cdot g = 0,\ \deg g \leqslant d\}.$$

In [5] two algorithms for computation of $A_d(f)$ are introduced. First of them is deterministic and has complexity that bounded from above by some polynomial in 2^n. The other algorithm is probabilistic. Its time of computation has the mathematical expectation that bounded from above by some polynomial in n. But this algorithm has nonzero probability of wrong result. Besides, the algorithm assumes quick random access to input data.

In this paper we introduce several deterministic algorithms such that their complexity bounded from above by some polynomial in n and in length of a function representation.

*Moscow State University, the Faculty of Computational Mathematics and Cybernetics, `vbayev@yandex.ru`

We parameterize functions from $\mathcal{F}_n$ by words of finite length in alphabet $\{0,1\}$. This means that for some set of words $Y_n \subset \{0,1\}^*$ we consider a map $\varphi_n : Y_n \to \mathcal{F}_n$. In these terms, a Boolean function is determined by some pair (n, y), where $n \in \mathbb{N},\ y \in Y_n$. We shall use only "reasonable" maps φ_n. There should exist a polynomial algorithm with input (n, y, x) (here $n \in \mathbb{N},\ y \in Y_n,\ x \in V_n$) such that this algorithm computes the value $\varphi_n(y)(x)$.

Theorem 1. ([2]) Let y be a list of all monomials in polynomial representation of a Boolean function $f_y \in \mathcal{F}_n$, i. e., f_y is equal to the sum of all monomials from the list y. Then there exists an algorithm with the following features. This algorithm has input (n, d, y), it computes a basis of the vector space $A_d(f_y)$, and its time complexity is $O(M_y \cdot (S_n^d)^3)$, where M_y is the number of monomials in the list y and $S_n^d = \sum_{k=0}^{d} C_n^k$.

Proposition 1. For arbitrary $f_1, f_2 \in \mathcal{F}_n$ the following relations of vector subspaces of $\mathcal{F}_n$ hold:

$$A_d(f_1) + A_d(f_2) \subset A_d(f_1 \cdot f_2),$$

$$A_d(f_1 + 1) + A_d(f_2 + 1) \subset A_d(f_1 \vee f_2 + 1),$$

$$A_d(f_1) \cap A_d(f_2) = A_d(f_1 \vee f_2),$$

$$A_d(f_1 + 1) \cap A_d(f_2 + 1) = A_d(f_1 \cdot f_2 + 1).$$

The proof is straightforward.

For $x, \alpha \in V_n,\ x = (x_1, \ldots, x_n),\ \alpha = (\alpha_1, \ldots, \alpha_n)$ we denote

$$x^\alpha := \prod_{i=1}^{n} x_i^{\alpha_i},$$

where

$$x_i^{\alpha_i} := \begin{cases} x_i, & \alpha_i = 1 \\ 1, & \alpha_i = 0. \end{cases}$$

Also, by $B_{n,d}$ denote the set $\{f \in \mathcal{F}_n | \deg f \leqslant d\}$.

Theorem 2. There exists an algorithm with the following features. The input of this algorithm is DNF (disjunctive normal form) that corresponds to a function $f \in \mathcal{F}_n$. The output of this algorithm is a basis of the vector space $A_d(f)$. Finally, the time complexity of this algorithm is bounded from above by some polynomial in n and in length of DNF.

Proof. For any $\alpha \in V_n$ we can compute a basis $\mathcal{B}$ of the vector space $A_d(x^\alpha)$ using the algorithm from theorem 1. It takes $O\left((S_n^d)^3\right)$ bit operations. Each basis vector $b \in \mathcal{B}$ is represented in the form of b's coordinates in monomial basis of $B_{n,d}$. Let $\sigma \in V_n$ be an arbitrary vector. Consider the map $\varphi_\sigma : B_{n,d} \to B_{n,d}$ that is given by the formula $\varphi_\sigma(g)(x) = g(x + \sigma)$. It is clear that for any $g \in A_d(x^\alpha)$ its image $\varphi_\sigma(g)$ belongs to $A_d((x+\sigma)^\alpha)$. Moreover, φ_σ gives isomorphism $A_d(x^\alpha) \cong A_d((x+\sigma)^\alpha)$. The linear map φ_σ has the matrix Φ_σ of size $S_n^d \times S_n^d$. It is easy to construct a polynomial algorithm that computes this matrix. Thus $\{\Phi_\sigma \cdot b | b \in \mathcal{B}\}$ is the basis of $A_d((x+\sigma)^\alpha)$. So, we can obtain polynomial algorithm that computes the basis of $A_d((x+\sigma)^\alpha)$.

Let $f \in \mathcal{F}_n$ be represented in the form of DNF:

$$f(x) = \bigvee_{k=1}^{T} (x + \sigma^k)^{\alpha^k},$$

where $\sigma^k, \alpha^k \in V_n$ $(k = 1, ..., T)$. Then by proposition 1,

$$A_d(f) = \bigcap_{k=1}^{T} A_d\left((x + \sigma^k)^{\alpha^k}\right).$$

Therefore, having bases of $A_d\left((x + \sigma^k)^{\alpha^k}\right)$, we can compute a basis of $A_d(f)$ via methods of linear algebra. The time complexity of such algorithm is bounded from above by polynomial in n and in T. ■

Theorem 3. Let $f \in \mathcal{F}_n$ be represented in the form of CNF (conjunctive normal form). Consider the problem of computing of a basis of $A_d(f)$, having CNF of f. We claim that for every $d \geqslant 0$ this problem is NP-hard.

Proof. It is clear that

$$f = 0 \Leftrightarrow A_d(f) = B_{n,d} \Leftrightarrow \dim A_d(f) = S_n^d.$$

Thus the problem of computing of a basis of $A_d(f)$, having CNF of f, is polynomial-time reducible to CNF-satisfiability problem, which is NP-complete. ■

Now, let a Boolean function $f \in \mathcal{F}_n$ be given by a formula F such that this formula consists of symbols of variables, brackets, and the Boolean operations $\neg, \&, \vee$. We want to search for low degree annihilators recursively. Sometimes we shall replace the operation $\neg$ by "+1". Let F' be some subformula of F, f' be the Boolean function that corresponds to F'. In this notation, for every subformula F' we shall obtain a pair of vector spaces

$$G_d(f') \subset A_d(f' + 1),\ H_d(f') \subset A_d(f'), \tag{1}$$

These vector spaces are given by their basis functions. As above, each basis function is represented in the form of its coordinates in monomial basis of $B_{n,d}$.

In the leaves of recursion tree we have the functions of the form $f_i(x_1, \ldots, x_n) = x_i$. In this case, there exists an algorithm such that its time complexity is polynomial in n and this algorithm computes bases of the following vector spaces:

$$A_d(x_i + 1) = \{g \cdot x_i | g \in \mathcal{F}_n,\ g \text{ does not depend on } x_i,\ \deg g \leqslant d - 1\},$$

$$A_d(x_i) = \{g \cdot (x_i + 1) | g \in \mathcal{F}_n,\ g \text{ does not depend on } x_i,\ \deg g \leqslant d - 1\}.$$

Therefore, we can assign $G_d(f_i) := A_d(f_i + 1),\ H_d(f_i) := A_d(f_i)$.

Let a subformula be of the form $f' = f_1 + 1 = \neg f_1$. Suppose recursive condition (1) holds for the function f_1. Then, if we make the following assignments

$$H_d(f') := G_d(f_1),$$

$$G_d(f') := H_d(f_1),$$

recursive condition (1) holds for the function f'.

Let a subformula be of the form $f' = f_1 \cdot f_2$. Suppose (1) holds for the functions f_1 and f_2. By definition, put $G_d(f') := G_d(f_1) \cap G_d(f_2)$, $H_d(f') := H_d(f_1) + H_d(f_2)$. Using proposition 1 and recursive condition (1) for f_1 and f_2, we obtain

$$G_d(f') \subset A_d(f_1+1) \cap A_d(f_2+1) = A_d(f_1 \cdot f_2 + 1) = A_d(f'+1),$$

$$H_d(f') \subset A_d(f_1) + A_d(f_2) \subset A_d(f_1 \cdot f_2) = A_d(f').$$

Finally, let a subformula be of the form $f' = f_1 \vee f_2$. Suppose (1) holds for the functions f_1 and f_2. By definition, put $G_d(f') := G_d(f_1) + G_d(f_2)$, $H_d(f') := H_d(f_1) \cap H_d(f_2)$. Again, using proposition 1 and recursive condition (1) for f_1 and f_2, we obtain

$$G_d(f') \subset A_d(f_1+1) + A_d(f_2+1) \subset A_d(f_1 \vee f_2 + 1) = A_d(f'+1),$$

$$H_d(f') \subset A_d(f_1) \cap A_d(f_2) = A_d(f_1 \vee f_2) = A_d(f').$$

We can use this recursive algorithm to compute bases of the vector subspaces $G_d(f) \subset A_d(f+1)$, $H_d(f) \subset A_d(f)$. It is easy to check that the time complexity of this algorithm is polynomial in n and in length of the formula F.

But this algorithm has a drawback. The vector subspaces $G_d(f), H_d(f)$ might be equal to $\{0\}$, while $A_d(f+1)$ and $A_d(f)$ are nontrivial. In some cases the inclusion $A_d(f_1) + A_d(f_2) \subset A_d(f_1 \cdot f_2)$ is the equality. The remaining part of this paper contains two theorems about this property.

Theorem 4. Let $f_1, f_2 \in \mathcal{F}_n$ be nonzero affine functions such that $f_1 \neq f_2$ and $f_1 \neq f_2 + 1$. Then the vector space $A_1(f_1 \cdot f_2)$ is the following direct sum

$$A_1(f_1 \cdot f_2) = A_1(f_1) \oplus A_1(f_2).$$

Proof. If $\ell \in \mathcal{F}_n$ is an arbitrary nonzero affine function then $A_1(\ell) = \{0, \ell+1\}$. Hence the sum of subspaces $A_1(f_1)$, $A_1(f_2)$ is direct. We have to prove that $\dim A_1(f_1 \cdot f_2) = 2$.

It is easy to prove that for the functions f_1, f_2 there exists an invertible affine map $\tau : V_n \to V_n$ such that

$$\ell_1(x_1, \ldots, x_n) := f_1 \circ \tau(x_1, \ldots, x_n) = x_1,$$
$$\ell_2(x_1, \ldots, x_n) := f_2 \circ \tau(x_1, \ldots, x_n) = x_1 + x_2.$$

Since τ is invertible, we have the following isomorphisms:

$$A_1(f_1) \cong A_1(f_1 \circ \tau) = A_1(\ell_1),$$

$$A_1(f_2) \cong A_1(f_2 \circ \tau) = A_1(\ell_2),$$

$$A_1(f_1 \cdot f_2) \cong A_1((f_1 \cdot f_2) \circ \tau) = A_1((f_1 \circ \tau) \cdot (f_2 \circ \tau)) = A_1(\ell_1 \cdot \ell_2).$$

Represent $g \in A_1(\ell_1 \cdot \ell_2)$ in the following form

$$g(x_1, \ldots, x_n) = a_0 + \sum\nolimits_{i=1}^{n} a_i x_i.$$

It is obvious that $a_i = 0$ for any $i \geqslant 3$. Then

$$g \in A_1(\ell_1 \cdot \ell_2) \Leftrightarrow$$

$$g \cdot \ell_1 \cdot \ell_2 = 0 \Leftrightarrow$$

$$(a_0 + a_1x_1 + a_2x_2) \cdot x_1 \cdot (x_1 + x_2) = 0 \Leftrightarrow$$

$$a_0x_1 + a_1x_1 + a_2x_1x_2 + a_0x_1x_2 + a_1x_1x_2 + a_2x_1x_2 = 0 \Leftrightarrow$$

$$\begin{cases} a_0 + a_1 = 0 \\ a_2 + a_0 + a_1 + a_2 = 0 \end{cases} \Leftrightarrow$$

$$a_0 + a_1 = 0.$$

Thus we have three coefficients a_0, a_1, a_2 and one equation $a_0 + a_1 = 0$. Therefore $\dim A_1(f_1 \cdot f_2) = \dim A_1(\ell_1 \cdot \ell_2) = 2$. ∎

Theorem 5. Let $f_1, f_2 \in \mathcal{F}_n$ be nonzero functions such that f_2 does not depend on the first m variables and f_1 does not depend on the last $n - m$ variables. Then the vector space $A_1(f_1 \cdot f_2)$ is the following direct sum

$$A_1(f_1 \cdot f_2) = A_1(f_1) \oplus A_1(f_2).$$

Proof. It is clear that $A_1(f_1) \cap A_1(f_2) = \{0\}$. Let us show that any Boolean function $\ell \in A_1(f_1 \cdot f_2)$ can be represented in the form $\ell = \ell_1 + \ell_2$, where $\ell_1 \in A_1(f_1)$, $\ell_2 \in A_1(f_2)$. Consider $z = (z_1, \ldots, z_n) \in V_n$. By x denote $(z_1, \ldots, z_m)$, by y denote $(z_{m+1}, \ldots, z_n)$. In this notation we have $(x, y) = z$. Let $\ell \in A_1(f_1 \cdot f_2)$ be given by

$$\ell(z) = \sum_{i=1}^{n} a_i z_i + b.$$

Then ℓ can be represented in the form

$$\ell(z) = \ell'(x) + \ell''(y),$$

where

$$\ell'(x) = \sum_{i=1}^{m} a_i z_i, \quad \ell''(y) = \sum_{i=m+1}^{n} a_i z_i + b.$$

Hence

$$\ell \in A_1(f_1 \cdot f_2) \Leftrightarrow$$

$$\forall x \forall y \quad \ell(x, y) \cdot f_1(x) \cdot f_2(y) = 0 \quad \Leftrightarrow$$

$$\forall x \forall y \quad \ell'(x) \cdot f_1(x) \cdot f_2(y) + \ell''(y) \cdot f_1(x) \cdot f_2(y) = 0 \tag{2}$$

There are only two possibilities:

(a) $\boxed{\forall x \quad \ell'(x) \cdot f_1(x) = 0}$: The condition $f_1 \neq 0$ means that $\exists x_0 : f_1(x_0) = 1$. Substituting x_0 for x in (2), we get

$$\forall y \quad 0 \cdot f_2(y) + \ell''(y) \cdot 1 \cdot f_2(y) = 0 \quad \Leftrightarrow$$

$$\forall y \quad \ell''(y) \cdot f_2(y) = 0.$$

Thus we have $\ell' \in A_1(f_1)$ and $\ell'' \in A_1(f_2)$.

(b) $\boxed{\exists x_0 : \ell'(x_0) \cdot f_1(x_0) = 1}$: In this case $f_1(x_0) = 1$. If we replace x by x_0 in (2), we obtain

$$\forall y \quad 1 \cdot f_2(y) + \ell''(y) \cdot 1 \cdot f_2(y) = 0 \quad \Leftrightarrow$$

$$\forall y \quad (\ell''(y) + 1) \cdot f_2(y) = 0 \quad \Leftrightarrow$$

$$\forall y \quad \ell''(y) \cdot f_2(y) = f_2(y).$$

If we combine the last equation with (2), we get

$$\forall x \forall y \quad \ell'(x) \cdot f_1(x) \cdot f_2(y) + f_1(x) \cdot f_2(y) = 0 \quad \Leftrightarrow$$

$$\forall x \forall y \quad (\ell'(x) + 1) \cdot f_1(x) \cdot f_2(y) = 0.$$

The condition $f_2 \neq 0$ means that $\exists y_0 : f_2(y_0) = 1$. Therefore

$$\forall x \quad (\ell'(x) + 1) \cdot f_1(x) = 0.$$

Finally, we obtain $\ell' + 1 \in A_1(f_1)$, $\ell'' + 1 \in A_1(f_2)$, and $(\ell' + 1) + (\ell'' + 1) = \ell$. ■

References

[1] F. Armknecht: *On the Existence of low-degree Equations for Algebraic Attacks*, Cryptology ePrint Archive: Report 2004/185, http://eprint.iacr.org/2004/185

[2] V.V. Bayev: *On Some Algorithms for Constructing Annihilators of Low Degree for Boolean Functions*, to be published in J. "Discrete Mathematics" (in Russian).

[3] N. Courtois, W. Meier: *Algebraic Attacks on Stream Ciphers with Linear Feedback*, Eurocrypt 2003, LNCS 2656, pp. 345-359, Springer, 2003.

[4] N. Courtois: *Fast Algebraic Attacks on Stream Ciphers with Linear Feedback*, Crypto 2003, LNCS 2729, pp. 176-194, Springer, 2003.

[5] W. Meier, E. Pasalic, C. Carlet: *Algebraic Attacks and Decomposition of Boolean Functions*, Eurocrypt 2004, LNCS 3027, pp. 474-491, Springer, 2004.