Towards an Efficient Algorithm to find Annihilators by Solving a Set of Homogeneous Linear Equations

Deepak Kumar Dalai and Subhamoy Maitra
Applied Statistics Unit, Indian Statistical Institute
203 B T Road, Kolkata 700 108, INDIA
Email: {deepak_r, subho}@isical.ac.in

Abstract

In this paper we study in detail how to design an efficient algorithm for checking whether a Boolean function has an annihilator at a specific degree by solving a set of homogeneous linear equations. The strategy for this purpose is first to form a set of homogeneous linear equations from the truth table of the function (the construction step) and then solve that set of equations (the solution step). The algorithm presented by Meier, Pasalic and Carlet in Eurocrypt 2004 solves the construction step in $\frac{1}{8}(\sum_{i=0}^{d}\binom{n}{i}\sum_{j=0}^{i-1}\binom{n}{j})(2^n-\sum_{i=0}^{d}\binom{n}{i})$ time for deciding whether there exists any annihilator at a degree $\leq d$ for an n variable random balanced Boolean function. We show that it is possible to asymptotically improve the construction step in $\frac{1}{4}(\sum_{i=0}^{d}\binom{n}{i})(2^n-\sum_{i=0}^{d}\binom{n}{i})$ time. The solution step requires to solve the set of homogeneous linear equations (basically to calculate the rank) and this can be improved only when the number of variables and the number of equations are minimized. We note that this is possible if one can find an affine transformation over $f(x)$ to get $h(x) = f(Bx + b)$ such that $|\{x|h(x) = 1, wt(x) \leq d\}|$ is maximized. Though we could not discover a deterministic way to find such an affine transformation efficiently, we present an efficient heuristic towards that. Using our techniques, we could analyse the annihilators of any random balanced Boolean functions on 16 variables in around 2 hours in a Pentium 4 personal computer with 1 GB RAM.

Keywords: Boolean Functions, Annihilators, Algebraic Normal Form, Homogeneous Linear Equations.

1 Introduction

Results on algebraic attacks have received a lot of attention recently in studying the security of crypto systems [2,3,7,9,11,12,14–16,20]. Boolean function is one of the important primitives to be used in crypto systems and in terms of algebraic attacks the annihilators of a Boolean function play the most serious role [4–6,8,10,17–19,21].

Denote the set of all n-variable Boolean functions by B_n. Any Boolean function can be uniquely represented as a multivariate polynomial over $GF(2)$, called the algebraic normal form (ANF), as

$$f(x_1,\ldots,x_n) = a_0 + \sum_{1\leq i\leq n} a_i x_i + \sum_{1\leq i<j\leq n} a_{i,j}x_i x_j + \ldots + a_{1,2,\ldots,n}x_1 x_2 \ldots x_n,$$

where the coefficients $a_0, a_i, a_{i,j}, \ldots, a_{1,2,\ldots,n} \in \{0,1\}$. The algebraic degree, $\deg(f)$, is the number of variables in the highest order term with non zero coefficient. Given $f \in B_n$, a

nonzero function $g \in B_n$ is called an annihilator of f if $f * g = 0$. A function f should not be used if either f or $1 + f$ has a low degree annihilator. It is also known [15, 21] that for any function either f or $1 + f$ must have an annihilator at the degree $\lceil \frac{n}{2} \rceil$. Thus the target of a good design is to use a function f such that neither f nor $1 + f$ has an annihilator at a degree $< \lceil \frac{n}{2} \rceil$. Thus there is a need to construct such functions and the first one in this direction appeared in [18]. Later symmetric functions with this property has been presented in [19] followed by [8]. However, all these constructions are not good in terms of other cryptographic properties.

Thus there is a need to study the Boolean functions, which are rich in terms of other cryptographic properties, in terms of their annihilators. One has to find out the annihilators of a given Boolean function for this. So far the known published algorithm presented in this direction is [21, Algorithm 2]. In [8], there is an efficient algorithm to find the annihilators of symmetric Boolean functions, but symmetric Boolean functions are not cryptographically promising.

Very recently a more efficient algorithm than [21, Algorithm 2] has been proposed [1] in this direction. The algorithm presented in [1] can be used efficiently to find out relationships for algebraic and fast algebraic attacks. Note that the basic idea in our effort is to reduce the size of the matrix UA^r (used to solve the system of homogeneous linear equations, see Lemma 2 in Section 3) as far as possible. In [1], matrix triangularization has been exploited nicely to solve the annihilator finding problem (of degree d for an n-variable function) in $O(\binom{n}{d})^2$ time complexity. We expect that the triangularization strategy [1] can be applied on the reduced UA^r proposed by us to get further improvement.

As it has been attempted so far, we will also follow the well known strategy where first we form a set of homogeneous linear equations from the truth table of the function (the construction step) and then solve that set of equations (the solution step).

For checking annihilators up to degree d for a random balanced function on n-variables, we require $\frac{1}{4}(\sum_{i=0}^{d} \binom{n}{i})(2^n - \sum_{i=0}^{d} \binom{n}{i})$ time in the construction step which is asymptotically better than [21, Algorithm 2] which takes $\frac{1}{8}(\sum_{i=0}^{d} \binom{n}{i} \sum_{j=0}^{i-1} \binom{n}{j})(2^n - \sum_{i=0}^{d} \binom{n}{i})$ time. This is explained in Section 3.

The next one is the solution step where the well known Gaussian elimination technique takes roughly $O(t^3)$ time when we consider a $t \times t$ matrix in the solution step. There are algorithms with better time complexity like Strassen's algorithm which takes $O(t^{\log_2 7}) \approx O(t^{2.807})$ time and also the one by Coppersmith and Winograd in [13] that takes $O(t^{2.376})$ time. Gröbner bases algorithms like F5 may also be interesting, but it has not been explored yet in this particular context [9]. Towards simplicity and for comparison, we find it better to analyse the time complexity in terms of the well known Gaussian elimination technique for solution of homogeneous linear equations. However, it should be noted that the solution step may be improved further to find out the complete annihilator space based on the recent work proposed in [1].

Note that, for improvement in the solution step our main target is to reduce t, where we are considering the $t \times t$ matrix. With the reduction of t, any kind of algorithm to solve the system will work better. This we present in Section 4. We note that the solution step can be executed much more efficiently if one can find an affine transformation over $f(x)$ to get $h(x) = f(Bx + b)$ such that $|\{x|h(x) = 1, wt(x) \leq d\}|$ is maximized. Basically as $|\{x|h(x) = 1, wt(x) \leq d\}|$ increases, the matrix size (used to solve the system of homogeneous linear equations) decreases and hence the solution step becomes much faster. Till now we could

not discover a deterministic way to find such an affine transformation efficiently. However we present an effective heuristic towards this.

2 Preliminaries

Consider all the n-variable Boolean functions of degree at most d, i.e., $\mathcal{R}(n,d)$, the Reed-Muller code of order d and length 2^n. Any Boolean function can be seen as a multivariate polynomial over $GF(2)$. Note that $\mathcal{R}(n,d)$ is a vector subspace of the vector space $\mathcal{B}_n$, the set of all n-variable Boolean functions. Now if we consider the elements of $\mathcal{R}(n,d)$ as the multivariate polynomials over $GF(2)$, then the standard basis is the set of all nonzero monomials of degree $\leq d$. That is, the standard basis is

$$S_{n,d} = \{x_{i_1} \dots x_{i_l} : 1 \leq l \leq d \text{ and } 1 \leq i_1 < i_2 < \cdots < i_l \leq n\} \cup \{1\},$$

where we consider that the input variables of the Boolean functions are $x_1, \dots, x_n$.

The ordering among the monomials is considered in lexicographic ordering ($<_l$) as usual, i.e., $x_{i_1}x_{i_2}\dots x_{i_k} <_l x_{j_1}x_{j_2}\dots x_{j_l}$ if either $k < l$ or $k = l$ and there is $1 \leq p \leq k$ such that $i_k = j_k, i_{k-1} = j_{k-1}, \dots, i_{p+1} = j_{p+1}$ and $i_p < j_p$. So, the set $S_{n,d}$ is a totally ordered set with respect to this lexicographical ordering ($<_l$). Using this ordering we refer the monomials according their order, i.e., the k-th monomial as m_k, $1 \leq k \leq \sum_{i=0}^{d} \binom{n}{i}$ following the convention $m_l <_l m_k$ if $l < k$.

Definition 1 *Given $n > 0$, $0 \leq d \leq n$, we define a mapping $v_{n,d} : \{0,1\}^n \mapsto \{0,1\}^{\sum_{i=0}^{d}\binom{n}{i}}$, such that $v_{n,d}(x) = (m_1(x), m_2(x), \dots, m_{\sum_{i=0}^{d}\binom{n}{i}}(x))$, where $m_i(x)$ is the ith monomial as in the lexicographical ordering ($<_l$) evaluated at the point $x = (x_1, x_2, \dots, x_n)$.*

To evaluate the value of the t-th coordinate of $v_{n,d}(x_1, x_2, \dots, x_n)$ for $1 \leq t \leq \sum_{i=0}^{d}\binom{n}{i}$, i.e., $[v_{n,d}(x_1, \dots, x_n)]_t$, one requires to calculate the value of the monomial m_t (either 0 or 1) at $(x_1, x_2, \dots, x_n)$. Now we define a matrix $M_{n,d}$ with respect to a n-variable function f. To define this we need another similar ordering ($<^l$) over the elements of vector space $\{0,1\}^n$. We say for $u, v \in \{0,1\}^n$, $u <^l v$ if either $wt(u) < wt(v)$ or $wt(u) = wt(v)$ and there is some $1 \leq p \leq n$ such that $u_n = v_n, u_{n-1} = v_{n-1}, \dots, u_{p+1} = v_{p+1}$ and $u_p = 0, v_p = 1$.

Definition 2 *Given $n > 0$, $0 \leq d \leq n$ and an n-variable Boolean function f, we define a $wt(f) \times \sum_{i=0}^{d}\binom{n}{i}$ matrix*

$$M_{n,d}(f) = \begin{bmatrix} v_{n,d}(X_1) \\ v_{n,d}(X_2) \\ \vdots \\ v_{n,d}(X_{wt(f)}) \end{bmatrix}$$

where any X_i is an n-bit vector and $supp(f) = \{X_1, X_2, \dots, X_{wt(f)}\}$ and $X_1 <^l X_2 <^l \cdots <^l X_{wt(f)}$. We denote by $supp(f)$ the set of inputs for which f outputs 1.

Note that the matrix $M_{n,d}(f)$ is the transpose of the restricted generator matrix for Reed-Muller code of length 2^n and order d, $\mathcal{R}(d,n)$, to the support of f (see also [9, Page 7].

Any row of the matrix $M_{n,d}(f)$ corresponding to an input vector $(x_1, \dots, x_n)$ is

$$\overbrace{1}^{0-degree} \quad \overbrace{x_1, \dots, x_i, \dots, x_n}^{1-degree} \quad \dots \quad \overbrace{x_1 \dots x_d, \dots, x_{i_1} \dots x_{i_d}, \dots, x_{n-d+1} \dots x_n}^{d-degree}.$$

Each column of the matrix is represented by a specific monomial and each entry of the column tells whether that monomial is satisfied by the input vector which identifies the row, i.e., the rows of this matrix correspond to the evaluations of the monomials having degree at most d on support of f. As already discussed, here we have one-to-one correspondence from the input vectors $x = (x_1, \ldots, x_n)$ to the row vectors $v_{n,d}(x)$ of length $\sum_{i=0}^{d} \binom{n}{i}$. So, each row is fixed by an input vector.

2.1 Annihilator of f and rank of the matrix $M_{n,d}(f)$

Let f be an n variable Boolean function. We are interested to find out the lowest degree annihilators of f. Let $g \in B_n$ be an annihilator of f, i.e., $f(x_1, \ldots, x_n) * g(x_1, \ldots, x_n) = 0$. In terms of truth table, this means that the function f AND g will be a constant zero function, i.e., for each vector $(x_1, \ldots, x_n) \in \{0,1\}^n$, the output of f AND g will be zero. That means,

$$g(x_1, \ldots, x_n) = 0 \text{ if } f(x_1, \ldots, x_n) = 1. \tag{1}$$

Suppose degree of the function g is $\leq d$, then the ANF of g is of the form $g(x_1, \ldots, x_n) = a_0 + \sum_{i=0}^{n} a_i x_i + \cdots + \sum_{1 \leq i_1 < i_2 \cdots < i_d \leq n} a_{i_1,\ldots,i_d} x_{i_1} \cdots x_{i_d}$ where the subscripted a's are from $\{0,1\}$ and not all of them are zero. Following Equation 1, we get the following $wt(f)$ many homogeneous linear equations

$$a_0 + \sum_{i=0}^{n} a_i x_i + \cdots + \sum_{1 \leq i_1 < i_2 \cdots < i_d \leq n} a_{i_1,\ldots,i_d} x_{i_1} \cdots x_{i_d} = 0, \tag{2}$$

considering the input vectors $(x_1, \ldots, x_n) \in supp(f)$. Note that this is a system of homogeneous linear equations on a's with $\sum_{i=0}^{d} \binom{n}{i}$ many a's as variables. The matrix form of this system of equations is

$$M_{n,d}(f)\ A^{tr} = O, \tag{3}$$

where $A = (a_0, a_1, a_2, \ldots, a_{n-d+1,\ldots,n})$, the row vector of coefficients of the monomials which are ordered according to the lexicographical order $<_l$. Each nonzero solution of the system of equations formed by Equation 2 gives an annihilator g of degree $\leq d$. This is basically the Algorithm 1 presented in [21]. Since the number of solutions of this system of equations are connected to the rank of the matrix $M_{n,d}(f)$, it is worth to study the rank and the set of linear independent rows/columns of matrix $M_{n,d}(f)$. If the rank of matrix $M_{n,d}(f)$ is equal to $\sum_{i=0}^{d} \binom{n}{i}$ (i.e., number of columns) then the only solution is the zero solution. So, for this case f has no annihilator of degree $\leq d$. This implies that the number of rows $\geq$ number of columns, i.e., $wt(f) \geq \sum_{i=0}^{d} \binom{n}{i}$ which is the Theorem 1 in [17]. If the rank of matrix is equal to $\sum_{i=0}^{d} \binom{n}{i} - k$ for $k > 0$ then the number of linearly independent solutions of the system of equations is k which gives k many linearly independent annihilators of degree $\leq d$ and $2^k - 1$ many number of annihilators of degree $\leq d$. However, to implement algebraic attack one needs only linearly independent annihilators. Hence, finding the degree of lowest degree annihilator of either f or $1 + f$, one can use the following algorithm.

Algorithm 1
for$(i = 1$ *to* $\lceil \frac{n}{2} \rceil - 1)$ {
    find the rank r_1 *of the matrix* $M_{n,i}(f)$*;*
    find the rank r_2 *of the matrix* $M_{n,i}(1 + f)$*;*

if $\min\{r_1, r_2\} < \sum_{j=0}^{i} \binom{n}{j}$ *then output* i*;*
}
output $\lceil \frac{n}{2} \rceil$*;*

Since either f or $1+f$ has an annihilator of degree $\leq \lceil \frac{n}{2} \rceil$, we are interested only to check till $i = \lceil \frac{n}{2} \rceil$. This algorithm is equivalent to Algorithm 1 in [21].

The simplest and immediate way to solve the system of these equations or find out the rank of $M_{n,d}(f), M_{n,d}(1+f)$ is the Gaussian elimination process. To check the existence or to enumerate the annihilators of degree $\leq \lceil \frac{n}{2} \rceil$ for a balanced function, the complexity is approximately $(2^{n-2})^3$. Considering this time complexity, it is not possible to check annihilators of a function of 20 variables or more using the presently available computing power. However, one should note that the matrix $M_{n,d}(f)$ has pretty good structure for a particular n and d, which we explore in this paper towards a better algorithm (that is solving the set of homogeneous linear equations in efficient way by decreasing the size of the matrix involved).

3 Faster algorithm for constructing the set of homogeneous linear equations

In this section we present a faster algorithm in this direction which is an improvement of Algorithm 2 presented in [21] in terms of constructing the set of homogeneous linear equations. First we present a technical result.

Theorem 1 *Let g be an n variable Boolean function defined as $g(x) = 1$ iff $wt(x) \leq d$ for $0 \leq d \leq n$. Then $M_{n,d}(g)^{-1} = M_{n,d}(g)$, i.e., $M_{n,d}(g)$ is an idempotent matrix.*

Proof: Suppose $M_{n,d}(g)M_{n,d}(g) = \mathcal{F}$. Then the i-th row and j-th column entry of $\mathcal{F}$ (denoted by $\mathcal{F}_{i,j}$) is the scalar product of i-th row and j-th column of $M_{n,d}(g)$. Suppose the i-th row is $v_{n,d}(x)$ for $x \in \{0,1\}^n$ having $x_{q_1}, \ldots, x_{q_l}$ as 1 and others are 0. Further consider that the j-th column is the evaluation of the monomial $x_{r_1} \ldots x_{r_k}$ at the input vectors belonging to the support of g. If $\{r_1, \ldots, r_k\} \not\subseteq \{q_1, \ldots, q_l\}$ then $\mathcal{F}_{ij} = 0$. Otherwise, $\mathcal{F}_{i,j} = \binom{l-k}{0} + \binom{l-k}{1} + \cdots + \binom{l-k}{l-k} \bmod 2 = 2^{l-k} \bmod 2$. So, $\mathcal{F}_{i,j} = 1$ iff $\{x_{r_1}, \ldots, x_{r_k}\} = \{x_{q_1}, \ldots, x_{q_l}\}$. That implies, $\mathcal{F}_{i,j} = 1$ iff $i = j$ i.e., $\mathcal{F}$ is identity matrix. Hence, $M_{n,d}(g)$ is its own inverse. ■

Example 1 *Let us present an example of $M_{n,d}(g)$ for $n = 4$ and $d = 2$. We have $\{1, x_1, x_2, x_3, x_4, x_1x_2, x_1x_3, x_2x_3, x_1x_4, x_2x_4, x_3x_4\}$, the list of 4 variable monomials of degree ≤ 2 in ascending order ($<_l$).*

Similarly, $\{(0,0,0,0), (1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1), (1,1,0,0), (1,0,1,0), (0,1,1,0), (1,0,0,1), (0,1,0,1), (0,0,1,1)\}$ present the 4 dimensional vectors of weight ≤ 2

in ascending order ($<^l$). So the matrix

$$M_{4,2}(g) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

One may check that $M_{4,2}(g)$ is idempotent.

Lemma 1 *Let A be a nonsingular $m \times m$ binary matrix where the row vectors are denoted as $v_1, v_2, \ldots, v_m$. Let U be a $k \times m$ binary matrix, $k \leq m$, where the rows are denoted as $u_1, u_2, \ldots, u_k$. Let $W = UA^{-1}$, a $k \times m$ binary matrix. Consider that a matrix A' is formed from A by replacing the rows $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$ of A by the vectors $u_1, u_2, \ldots, u_k$. Further consider that a $k \times k$ matrix W' is formed by taking the i_1-th, i_2-th, $\ldots, i_k$-th columns of W (out of m columns). Then A' is nonsingular iff W' is nonsingular.*

Proof: Without loss of generality, we can take $i_1 = 1, i_2 = 2, \ldots, i_k = k$. So, the row vectors of A' are $u_1, \ldots, u_k, v_{k+1}, \ldots, v_m$.

We first prove that if the row vectors of A' are not linearly independent then the row vectors of W' are also not linearly independent. As the row vectors of A' are not linearly independent, we have $\alpha_1, \alpha_2, \ldots, \alpha_m \in \{0, 1\}$ (not all zero) such that $\sum_{i=1}^{k} \alpha_i u_i + \sum_{i=k+1}^{m} \alpha_i v_i = 0$. If $\alpha_i = 0$ for all i, $1 \leq i \leq k$ then $\sum_{i=k+1}^{m} \alpha_i v_i = 0$ which implies $\alpha_i = 0$ for all i, $k+1 \leq i \leq m$ as $v_{k+1}, v_{k+2}, \ldots, v_m$ are linearly independent. So, all α_i's for $1 \leq i \leq k$ can not be zero.

Further, we have

$$UA^{-1} = W, \text{ i.e., } U = WA, \text{ i.e., } \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_k \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{pmatrix}.$$

So, $u_i = w_i \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{pmatrix}$.

Hence, $\sum_{i=1}^{k} \alpha_i u_i = \sum_{i=1}^{k} \alpha_i w_i \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{pmatrix} = r \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{pmatrix}$

where $r = (r_1, r_2, \ldots, r_m) = \sum_{i=1}^{k} \alpha_i w_i$.

If the restricted matrix W' were nonsingular, the vector $r' = (r_1, r_2, \ldots, r_k)$ is non zero as $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ is not all zero. Hence, $\sum_{i=1}^{k} \alpha_i u_i + \sum_{i=k+1}^{m} \alpha_i v_i = 0$, *i.e.*, $\sum_{i=1}^{k} r_i v_i + \sum_{i=k+1}^{m} (r_i + \alpha_i) v_i = 0$. This contradicts that $v_1, v_2, \ldots, v_m$ are linearly independent as $r' = (r_1, r_2, \ldots, r_k)$ is nonzero. Hence W' must be singular. This proves one direction.

On the other direction if the restricted matrix W' is singular then there are $\beta_1, \beta_2, \ldots, \beta_k$ not all zero such that $\sum_{i=0}^{k} \beta_i w_i = (0, \ldots, 0, s_{k+1}, \ldots, s_m)$.

Hence, $\sum_{i=0}^{k} \beta_i u_i = \sum_{i=1}^{k} \beta_i w_i \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{pmatrix} = s_{k+1} v_{k+1} + \cdots + s_m v_m$, i.e., $\sum_{i=0}^{k} \beta_i u_i + \sum_{i=k+1}^{m} s_i v_i = 0$ which says matrix A' is singular. ■

In the following result we present the Lemma 1 in more general form.

Theorem 2 *Let A be a nonsingular $m \times m$ binary matrix with m-dimensional row vectors $v_1, v_2, \ldots, v_m$ and U be a $k \times m$ binary matrix with m-dimensional row vectors $u_1, u_2, \ldots, u_k$. Consider $W = UA^{-1}$, a $k \times m$ matrix. The matrix A', formed from A by removing the rows $v_{i_1}, v_{i_2}, \ldots, v_{i_l}$ $(l \leq m)$ from A and adding the rows $u_1, u_2, \ldots, u_k$ $(k \geq l)$, is of rank m iff the rank of restricted $k \times l$ matrix W' including only the i_1-th, i_2-th, $\ldots$, i_l-th columns of W is l.*

Proof: Here, the rank of matrix W' is l. So, there are l many rows of W', say $w'_{p_1}, \ldots, w'_{p_l}$ which are linearly independent. So, following the Lemma 1 we have the matrix A'' formed by replacing the rows $v_{i_1}, \ldots, v_{i_l}$ of A by $u_{p_1}, \ldots, u_{p_l}$ is nonsingular, i.e., rank is m. Hence the matrix A' where some more rows are added to A'' has rank m. The other direction can also be shown similar to the proof of the other direction in Lemma 1. ■

Now using Theorem 1 and Theorem 2, we describe a faster algorithm to check the existence of annihilators of certain degree d of a Boolean function f. Suppose g be the Boolean function described in Theorem 1, i.e., $supp(g) = \{x | 0 \leq wt(x) \leq d\}$. In Theorem 1, we have already shown that $M_{n,d}(g)$ is nonsingular matrix (in fact it is idempotent). Let $\{x | wt(x) \leq d$ and $f(x) = 0\} = \{x_1, x_2, \ldots, x_l\}$ and $\{x | wt(x) > d$ and $f(x) = 1\} = \{y_1, y_2, \ldots, y_k\}$. Then we consider $M_{n,d}(f)$ as A, $v_{n,d}(x_1), \ldots, v_{n,d}(x_l)$ as $v_{i_1}, \ldots, v_{i_l}$ and $v_{n,d}(y_1), \ldots, v_{n,d}(y_k)$ as $u_1, \ldots, u_k$. Then following Theorem 2 we can ensure whether $M_{n,d}(f)$ is nonsingular. If it is nonsingular, then there is no annihilator of degree $\leq d$, else there are annihilator(s). We may write this in a more concrete form as the following corollary to Theorem 2.

Corollary 1 *Let f be an n variable Boolean function. Let A^r be the restricted matrix of $A = M_{n,d}(g)$, by taking the columns corresponding to the monomials $x_{i_1} x_{i_2} \ldots x_{i_l}$ such that $l \leq d$ and $f(x) = 0$ when $x_{i_1} = 1, x_{i_2} = 1, \ldots, x_{i_l} = 1$ and rest of the input variables are 0. Further $U = \begin{pmatrix} v_{n,d}(y_1) \\ v_{n,d}(y_2) \\ \vdots \\ v_{n,d}(y_k) \end{pmatrix}$, where $\{y_1, \ldots, y_k\} = \{x | wt(x) > d$ and $f(x) = 1\}$. If rank of UA^r is l then there is no annihilator of degree $\leq d$, else there are annihilator(s) of degree $\leq d$.*

Proof: As per Theorem 2, here $W = UA^{-1} = UA$, since A is idempotent following Theorem 1 and hence W' is basically UA^r. Thus the proof follows. ■

Now we can use the following technique for fast computation of the matrix multiplication UA^r. For this we first present a technical result and its proof is similar in the line of the proof of Theorem 1.

Proposition 1 *Consider the function g as in Theorem 1. Let $y \in \{0,1\}^n$ such that i_1, i_2, ..., i_p-th places are 1 and other places are 0. Consider the j-th monomial $m_j = x_{j_1}x_{j_2}\dots x_{j_q}$ according the ordering $<_l$. Then the j-th entry of $v_{n,d}(y)M_{n,d}(g)$ is 0 if $\{j_1,\dots,j_q\} \not\subseteq \{i_1,\dots,i_p\}$ else the value is $\sum_{i=0}^{d-q}\binom{p-q}{i} \bmod 2$.*

Following Proposition 1, we can get each row of U as some $v_{n,d}(y)$ and each column of A^r as m_j and construct the matrix UA^r. One can precompute the sums $\sum_{i=0}^{d-q}\binom{p-q}{i} \bmod 2$ for $d+1 \le p \le n$ and $0 \le q \le d$, and store them and the total complexity for calculating them is $O(d^2(n-d))$. These sums will be used to fill up the matrix UA^r which is an $l \times k$ matrix according to Corollary 1. Let us denote $\mu_f = |\{x|wt(x) \le d, f(x) = 1\}|$ and $\nu_f = |\{x|wt(x) > d, f(x) = 1\}|$. Then $wt(f) = \mu_f + \nu_f$ and the matrix UA^r is of dimension $\nu_f \times (\sum_{i=0}^{d}\binom{n}{i} - \mu_f)$. Clearly $O(d^2(n-d))$ can be neglected with respect to $\nu_f \times (\sum_{i=0}^{d}\binom{n}{i} - \mu_f)$. Thus we have the following result.

Lemma 2 *Consider U and A^r as in Corollary 1. The time (and also space) complexity to construct the matrix UA^r is of the order of $\nu_f \times (\sum_{i=0}^{d}\binom{n}{i} - \mu_f)$.*

To make this algorithm as probabilistic (equivalent to Algorithm 2 of [21]), we need to consider the matrix $U = \begin{pmatrix} v_{n,d}(y_1) \\ v_{n,d}(y_2) \\ \vdots \\ v_{n,d}(y_k) \end{pmatrix}$, where $k = \sum_{i=0}^{d}\binom{n}{i} - \mu_f$ and $\{y_1,\dots,y_k\} \in \{x|wt(x) > d \text{ and } f(x) = 1\}$. So, the matrix UA^r is a $(\sum_{i=0}^{d}\binom{n}{i} - \mu_f) \times (\sum_{i=0}^{d}\binom{n}{i} - \mu_f)$ sqaure matrix. If the rank of matrix UA^r is $(\sum_{i=0}^{d}\binom{n}{i} - \mu_f)$ then returns no annihilator of degree d. So, both the time and space complexities to construct equations of this probabilistic algorithm is $(\sum_{i=0}^{d}\binom{n}{i} - \mu_f) \times (\sum_{i=0}^{d}\binom{n}{i} - \mu_f)$.

3.1 Comparision with Meier et al [21] algorithm

Here we compare the time and space complexity of our strategy with [21, Algorithm 2]. In paper [21], Algorithm 2 is probablistic. In this draft we study the time and space complexity of the algorithm along with it's determinstic version. Using these algorithms we check whether there exist annihilators of degree $\le d$ of an n-variable function f. As we have already described, ANF of any n-variable function g of degree d is of the form $g(x_1,\dots,x_n) = a_0 + \sum_{i=0}^{n} a_i x_i + \dots + \sum_{1\le i_1<i_2\dots<i_d\le n} a_{i_1,\dots,i_d}x_{i_1}\cdots x_{i_d}$. First we present the exact probabilistic algorithm [21, Algorithm 2].

Algorithm 2

1. *Initialize weight $w = 0$.*
2. *For all x's of weight w with $f(x) = 1$, substitute each x in $g(x) = 0$ to derive a linear equation on the coefficients of g, with a single coefficient of weight w. Use this equation to express this coefficient iteratively by coefficients of lower weight.*
3. *If $w < d$, increment w by 1 and go to step 2.*
4. *Choose random arguments x of arbitrary weight such that $f(x) = 1$ and substitute in $g(x) = 0$, until there are same number of equations as unknowns.*

5. *Solve the linear system. If there is no solution, output no annihilator of degree d, but if there is a solution then it is not clear whether there is an annihilator of degree d or not.*

Next we present the deterministic version of the original probabilistic algorithm [21, Algorithm 2].

Algorithm 3

1. *Initialize weight $w = 0$.*
2. *For all x's of weight w with $f(x) = 1$, substitute each x in $g(x) = 0$ to derive a linear equation in the coefficients of g, with a single coefficient of weight w. Use this equation to express this coefficient iteratively by coefficients of lower weight.*
3. *If $w < d$, increment w by 1 and go to step 2.*
4. *Substitute x such that $wt(x) > d$ and $f(x) = 1$ in $g(x) = 0$ to get linear equation in the coefficient of g.*
5. *Solve the linear system. Output no annihilator of degree d iff there is no non zero solution.*

Since first three steps of both algorithms are same, we initially study the time and space complexity of both the algorithms for first three steps for a randomly choosen balanced function f. In step 2, we apply x, such that weight of $x \leq d$ and $f(x) = 1$, in $g(x)$ and hence we get a linear equation in the coefficient of g such that a single coefficient of that weight is expressed as linear combination of its lower weight coefficients. Here we consider a particular w for each iteration. As f is random and balanced, one can expect that there are $\frac{1}{2}\binom{n}{w}$ many input vectors of weight w in set $supp(f)$. For each $x = (x_1, \ldots, x_n) \in supp(f)$ where $x_{i_1}, \ldots, x_{i_w}$ are 1 and others are 0 of weight w, we will get linear equation of the form

$$a_{i_1,\ldots,i_w} = a_0 + \sum_{j=1}^{w} a_{i_j} + \cdots + \sum_{\{k_1,\ldots,k_{w-1}\}\subset\{i_1,\ldots i_w\}} a_{k_1,\ldots,k_{w-1}}. \tag{4}$$

To store one equation we need $\sum_{i=0}^{w}\binom{n}{i}$ many memory bits (some places will be 0, some will be 1). There are $\sum_{i=0}^{w-1}\binom{w}{i}$ many coefficients in the right hand side of the Equation 4. As f is random, one can expect that half of them can be eliminated using the equations obtained by lower weight input support vectors. So, $\sum_{i=0}^{w}\binom{n}{i} + \frac{1}{2}\sum_{i=0}^{w-1}(\binom{w}{i}\sum_{j=0}^{i-1}\binom{n}{j})$ order of computation is required to establish an equation. Here w varies from 0 to d and there are approximately $\frac{1}{2}\sum_{w=0}^{d}\binom{n}{w}$ many support vectors of weight $\leq d$. Hence at the starting of step 4 the space complexity is $S1 = \frac{1}{2}\sum_{w=0}^{d}(\binom{n}{w}\sum_{i=0}^{w}\binom{n}{i})$ and time complexity is $T1 = \frac{1}{2}\sum_{w=0}^{d}(\binom{n}{w}(\sum_{i=0}^{w}\binom{n}{i} + \frac{1}{2}\sum_{i=0}^{w-1}\binom{w}{i}\sum_{j=0}^{i-1}\binom{n}{j}))$.

Now we study the time and space complexity for steps 4 and 5 in both probabilistic and deterministic version. To represent each equation for the system of equation one needs $\sum_{w=0}^{d}\binom{n}{w}$ memory bits.

First we consider the probabilistic one. For probabilistic case one has to choose appoximately $\frac{1}{2}\sum_{w=0}^{d}\binom{n}{w}$ many support input vectors of weight $> d$. Hence each linear equation obtained from these vectors contains atleast $\sum_{i=0}^{d}\binom{d+1}{i}$ many coefficients of g and half of

them can be eliminated using the equations obtained in previous steps. So, to get each equation one needs atleast $\sum_{w=0}^{d}\binom{n}{w} + \frac{1}{2}\sum_{i=0}^{d}(\binom{d+1}{i}\sum_{j=0}^{i-1}\binom{n}{j})$ computations. Hence the space complexity during 4th step is $SP2 \geq \frac{1}{2}(\sum_{w=0}^{d}\binom{n}{w})^2$ and time complexity is $TP2 \geq \frac{1}{2}\sum_{w=0}^{d}\binom{n}{w}(\sum_{w=0}^{d}\binom{n}{w} + \frac{1}{2}\sum_{i=0}^{d}(\binom{d+1}{i}\sum_{j=0}^{i-1}\binom{n}{j}))$. Finally, to generate system of homogenuous linear equations one requires $SP = S1 + SP2 \geq \frac{1}{2}\sum_{w=0}^{d}(\binom{n}{w}\sum_{i=0}^{w}\binom{n}{i}) + \frac{1}{2}(\sum_{w=0}^{d}\binom{n}{w})^2$ memory bits and $TP = T1 + TP2 \geq \frac{1}{2}\sum_{w=0}^{d}(\binom{n}{w}(\sum_{i=0}^{w}\binom{n}{i} + \frac{1}{2}\sum_{i=0}^{w-1}\binom{w}{i}\sum_{j=0}^{i-1}\binom{n}{j}))) + \frac{1}{2}\sum_{w=0}^{d}\binom{n}{w}(\sum_{w=0}^{d}\binom{n}{w} + \frac{1}{2}\sum_{i=0}^{d}(\binom{d+1}{i}\sum_{j=0}^{i-1}\binom{n}{j})))$ computations. Then in step 5, we have to solve $\frac{1}{2}\sum_{w=0}^{d}\binom{n}{w}$ many linear equations with same number of variables. To solve this system one needs $TP3 = (\frac{1}{2}\sum_{w=0}^{d}\binom{n}{w})^{\omega}$ computations where ω is the degree for solving linear system of equations. For example, for Guassian elimination $\omega = 3$, Strassen's proccess $\omega = 2.8$ and Coppersmith and Winograd process $\omega = 2.3$.

Now we study space and time complexity for deterministic one. Since f is balanced, there are approximately $2^{n-1} - \frac{1}{2}\sum_{w=0}^{d}\binom{n}{w} = \frac{1}{2}\sum_{w=d+1}^{n}\binom{n}{w}$ many support vectors having weight $> d$ and these many are considered to find out equations. Hence each linear equation obtained from these vectors of weight $w > d$ contains $\sum_{i=0}^{d}\binom{w}{i}$ many coefficients of g and half of them can be eliminated using the equations obtained in steps 1, 2 and 3. To get this equation one needs $\sum_{i=0}^{d}\binom{n}{i} + \frac{1}{4}\sum_{i=0}^{d}(\binom{w}{i}\sum_{j=0}^{i-1}\binom{n}{j})$ computations. Hence the total space complexity during 4th step is $SD2 = \frac{1}{4}\sum_{w=d+1}^{n}\binom{n}{w}\sum_{w=0}^{d}\binom{n}{d})$ and time complexity is $TD2 = \frac{1}{2}\sum_{w=d+1}^{n}\binom{n}{w}(\sum_{i=0}^{d}\binom{n}{i} + \frac{1}{4}\sum_{i=0}^{d}(\binom{w}{i}\sum_{j=0}^{i-1}\binom{n}{j}))$. Finally, to generate homogenuous linear equations one needs $SD = S1 + SD2 = \frac{1}{2}\sum_{w=0}^{d}(\binom{n}{w}\sum_{i=0}^{w}\binom{n}{i}) + \frac{1}{4}\sum_{w=d+1}^{n}\binom{n}{w}\sum_{w=0}^{d}\binom{n}{d})$ memory bits and $TD = T1 + TD2 = \frac{1}{2}\sum_{w=0}^{d}(\binom{n}{w}(\sum_{i=0}^{w}\binom{n}{i} + \frac{1}{2}\sum_{i=0}^{w-1}\binom{w}{i}\sum_{j=0}^{i-1}\binom{n}{j}))) + \frac{1}{2}\sum_{w=d+1}^{n}\binom{n}{w}(\sum_{i=0}^{d}\binom{n}{i} + \frac{1}{4}\sum_{i=0}^{d}(\binom{w}{i}\sum_{j=0}^{i-1}\binom{n}{j}))$ computations. Further, in step 5, we have to solve $\frac{1}{2}\sum_{w=d+1}^{n}\binom{n}{w}$ many linear equations with $\frac{1}{2}\sum_{w=0}^{d}\binom{n}{w}$ number of variables. To solve this system one needs $TD3 = (\frac{1}{2}\sum_{w=d+1}^{n}\binom{n}{w})^{\omega}$ computations.

The system of equations generated by our strategy as well as Meier et al [21] algorithms are same. So, it takes same complexities to solve them. Only difference is during generation of the system of equations. In the following table we show the complexities for both algorithms for generating the system of equations.

	Space	Time
Meier's algorithm	$\frac{1}{2}\sum_{w=0}^{d}(\binom{n}{w}\sum_{i=0}^{w}\binom{n}{i})) + \frac{1}{2}(\sum_{w=0}^{d}\binom{n}{w})^2$	$\frac{1}{2}\sum_{w=0}^{d}(\binom{n}{w}(\sum_{i=0}^{w}\binom{n}{i} + \frac{1}{2}\sum_{i=0}^{w-1}\binom{w}{i}\sum_{j=0}^{i-1}\binom{n}{j}))$ $+\frac{1}{2}\sum_{w=0}^{d}\binom{n}{w}(\sum_{w=0}^{d}\binom{n}{w} + \frac{1}{2}\sum_{i=0}^{d}(\binom{d+1}{i}\sum_{j=0}^{i-1}\binom{n}{j}))$
Our algorithm	$\frac{1}{4}(\sum_{w=0}^{d}\binom{n}{w})^2$	$\frac{1}{4}(\sum_{w=0}^{d}\binom{n}{w})^2$

Table 1: Time and Space complexity comparision of Probabilistic algorithms to generate equations.

	Space	Time
Meier's algorithm	$\frac{1}{2}\sum_{w=0}^{d}(\binom{n}{w}\sum_{i=0}^{w}\binom{n}{i})) + \frac{1}{4}\sum_{w=d+1}^{n}\binom{n}{w}\sum_{w=0}^{d}\binom{n}{d})$	$\frac{1}{2}\sum_{w=0}^{d}(\binom{n}{w}(\sum_{i=0}^{w}\binom{n}{i} + \frac{1}{2}\sum_{i=0}^{w-1}\binom{w}{i}\sum_{j=0}^{i-1}\binom{n}{j}))$ $+\frac{1}{2}\sum_{w=d+1}^{n}\binom{n}{w}(\sum_{i=0}^{d}\binom{n}{i} + \frac{1}{4}\sum_{i=0}^{d}(\binom{w}{i}\sum_{j=0}^{i-1}\binom{n}{j}))$
Our algorithm	$\frac{1}{4}\sum_{w=d+1}^{n}\binom{n}{w}\sum_{w=0}^{d}\binom{n}{w}$	$\frac{1}{4}\sum_{w=d+1}^{n}\binom{n}{w}\sum_{w=0}^{d}\binom{n}{w}$

Table 2: Time and Space complexity comparision of Deterministic algorithms to generate equations.

4 Solving the set of homogeneous linear equations

To check for the annihilators, we need to compute the rank of the matrix UA^r. Following Lemma 2, it is clear that the size of the matrix UA^r will decrease if μ_f increases and ν_f decreases. Let B be an $n \times n$ nonsingular binary matrix and b be an n-bit vector. Note that $f(x)$ has an annihilator at degree d iff $f(Bx+b)$ has an annihilator at degree d. Thus one will try to get the affine transformation on the input variables of $f(x)$ to get $h(x) = f(Bx + b)$ such that $|\{x|h(x) = 1, wt(x) \leq d\}|$ is maximized. This is because in this case μ_h will be maximized and ν_h will be minimized and hence the dimension of the matrix UA^r, i.e., $\nu_f \times (\sum_{i=0}^{d} \binom{n}{i} - \mu_f)$ will be minimized. This will indeed decrease the complexity at the construction step (discussed in the previous section). More importantly, it will decrease the complexity to solve the system of homogeneous linear equations.

Example 2 *We present an example for this purpose. Consider the 5-variable Boolean function f constructed using the method presented in [18] such that neither f nor $1 + f$ has an annihilator at a degree < 3. The standard truth table representation of the function is 01010110010101100101011001101001, i.e., the outputs are corresponding to the inputs which are of increasing value. One can check that $|\{x \in \{0,1\}^5 \mid f(x) = 1 \ \& \ wt(x) < 3\}| = 6$. Now if we consider the function $h(x) = f(Bx + b)$ such that $B = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \end{bmatrix}$, and $b = \{1, 1, 1, 0, 0\}$, then $|\{x \in \{0,1\}^5 \mid h(x) = 1 \ \& \ wt(x) < 3\}| = 16$ and one can immediately conclude (from the results in [19]) that neither h nor $1 + h$ has an annihilator of degree < 3. This is an example where after finding the affine transformation there is even no need for the solution step at all. For the function f, here $h(x) = f(Bx + b)$ such that $|\{x|h(x) = 1, wt(x) \leq d\}|$ is maximized.*

We also present an example for a sub optimal case. In this case we consider $B = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix}$, and b an all zero vector, then $|\{x \in \{0,1\}^5 \mid h(x) = 1 \ \& \ wt(x) < 3\}| = 14$. Thus the dimension of the matrix UA^r becomes 2×2 as $\nu_f = 2$ and $\sum_{i=0}^{d} \binom{n}{i} - \mu_f = 2$. Thus one needs to check just the rank of a 2×2 matrix.

The example clearly shows the efficiency, but the problem is how to find such an affine transformation (for the optimal or even for sub optimal cases) efficiently.

For exhaustive search to get the optimal affine transform one needs to check $f(Bx+b)$ for all $n \times n$ nonsingular binary matrices B and n bit vectors b. Since there are $\prod_{i=0}^{n-1}(2^n - 2^i)$ many nonsingular binary matrices and 2^n many n bit vectors, one needs to check $2^n \prod_{i=0}^{n-1}(2^n - 2^i)$ many cases for an exhaustive search. As weight of the input vectors are invariant under permutation of the arguments, checking for only one nonsingular matrix from the set of all nonsingular matrices whose rows are equivalent under certain permutation will suffice. Hence the exact number of search options is $\frac{1}{n!} 2^n \prod_{i=0}^{n-1}(2^n - 2^i)$. One can check for $n \times n$ nonsingular binary matrices B where $row_i < row_j$ for $i < j$ (row_i is the decimal value of binary pattern of ith row). It is clear that the search is infeasible for $n \geq 8$.

Now we present a heuristics towards this. Our aim is to find out an affine transformation which maximizes the value of μ_f. This means the weight of the most of the input vectors having weight $\leq d$ should be in $supp(f)$. So we attempt to get an affine transformation for a Boolean function f such that the transformation increases the probability that an input vector, having output 1, will be translated to a low weight input vector. Suppose $r_1, r_2, \ldots, r_n$ are the n-dimensional row vectors of the transformation B and $b = (b_1, b_2, \ldots, b_n)$ be an n dimensional vector. Then $B(x_1, x_2, \ldots, x_n)^{tr} + b = (y_1, y_2, \ldots, y_n)^{tr}$ where $(x_1, x_2, \ldots, x_n), (y_1, y_2, \ldots, y_n)$ are n dimensional input vectors. The chance of $(y_1, y_2, \ldots, y_n)$ getting low weight if the probability of $y_i = 0, 1 \leq i \leq n$ is increased. That means the probability of $r_i \cdot (x_1, x_2, \ldots, x_n) + b_i = 0$ for $1 \leq i \leq n$ needs to be increased. Hence we will like to choose a linearly independent set $r_i \in \{0,1\}^n$, $1 \leq i \leq n$ and $b \in \{0,1\}^n$ such that the probability $r_i \cdot (x_1, x_2, \ldots, x_n) + b_i = 0$, $1 \leq i \leq n$ is high when $(x_1, x_2, \ldots, x_n) \in supp(f)$. The heuristic is presented below. By $bin[i]$ we denote the n-bit binary representation of the integer i.

Heuristic 1

1. *loop* $= 0$*; max* $= |\{x | f(x) = 1, wt(x) \leq d\}|$*;*
2. *For* $(i = 1; i < 2^n; i++)$ {
   (a) $t = |\{x = (x_1, x_2, \ldots, x_n) \in supp(f) | bin[i] \cdot x = 0\}|$
   (b) *if* $t \geq \frac{wt(f)}{2}$, $val[i] = t$ *and* $a_i = 0$ *else* $val[i] = wt(f) - t$ *and* $a_i = 1$.
   }
3. *Arrange the triplets* $(bin[i], a_i, val[i])$ *in descending order of* $val[i]$.
4. *Choose suitable* n *many triplets* (r_j, b_j, k_j) *for* $1 \leq j \leq n$ *such that* r_j*s are linearly independent and* k_j*'s are high.*
5. *Construct the nonsingular matrix* B *taking* r_j, $1 \leq j \leq n$ *as* j*-th row and* $b = (b_1, b_2, \ldots, b_n)$.
6. *Increment loop by* 1*; while* $(loop < maxval)$
   (a) *if* $max < |\{x | f(Bx + b) = 1, wt(x) \leq d\}|$ *replace* $f(x)$ *by* $f(Bx + b)$ *and update max by* $|\{x | f(Bx + b) = 1, wt(x) \leq d\}|$.
   (b) *Go to step 2.*

One can check that time complexity of this algorithm is $(maxval \times n2^{2n})$. To find the annihilator having degree $\lfloor \frac{n-1}{2} \rfloor$ for a random n-variable balanced function, the expected size of the matrix is $2^{n-2} \times 2^{n-2}$ and hence the complexity to find the rank of this matrix is 2^{3n-6} by Gaussian elimination technique. Considering $maxval$ as a constant, $(maxval \times n2^{2n})$ is asymptotically smaller than 2^{3n-6}. Thus this heuristics will be much less costlier in time complexity and if it can really provide a good transformation according to our specification.

Experiments with this heuristic on different Boolean functions provide very positive results. First of all we have considered the functions which are random affine transformations $g(x)$ of the function [19]

$$\begin{aligned} f_s(x) &= 1 \text{ for } wt(x) \leq \lfloor \frac{n-1}{2} \rfloor, \\ &= 0 \text{ for } wt(x) \geq \lfloor \frac{n+1}{2} \rfloor, \end{aligned}$$

which has no annihilator having degree $\leq \lfloor \frac{n-1}{2} \rfloor$. This experimentation has been done for $n = 5$ to 16. For all the cases running Heuristic 1 on $g(x)$ we could go back to $f_s(x)$. Then we have randomly changed up to 2^{n-4} bits on the upper half of $f_s(x)$ to get $f'_s(x)$ and then put random transformations on $f'_s(x)$ to get $g(x)$. Running Heuristic 1, we could also go back to $f'_s(x)$ easily. For experiments we have taken $maxval = 20$.

Next we have run our heuristics on randomly chosen balanced functions. Note that the number of inputs up to weight d for a Boolean function is $\sum_{i=0}^{d} \binom{n}{i}$. Thus for a randomly chosen balanced function, it is expected that there will be $\frac{1}{2} \sum_{i=0}^{d} \binom{n}{i}$ many inputs up to weight d for which the outputs are 1. Below we present the improvement (on an average of 100 experiments in each case) we got after running Heuristic 1 with $maxval = 20$ for $n = 12$ to 16.

n	12			13			14			15			16		
d	3	4	5	4	5	6	4	5	6	5	6	7	5	6	7
$\sum_{i=0}^{d} \binom{n}{i}$	299	794	1586	1093	2380	4096	1471	3473	6476	4944	9949	16384	6885	14893	26333
$\lceil \frac{1}{2} \sum_{i=0}^{d} \binom{n}{i} \rceil$	149	397	793	541	1190	2048	735	1736	3238	2472	4974	8192	3442	7446	13166
Heuristic Value	228	535	964	717	1438	2322	957	2051	3648	2917	5525	8811	3995	8194	14114

Table 3: Efficiency of Heuristic 1 on random balanced functions.

We have implemented our strategy in Linux environment (Redhat 8) in C programming language. The machine is a personal computer with Pentium 4 processor and 1 GB RAM. We could analyse the annihilators (up to degree 8) of any random balanced Boolean functions on 16 variables in around 2 hours in such an environment. With proper storage of the matrix UA^r in the hard disk, we consider it is possible to go up to 20 variables. Further it may also be possible to analyse low degree annihilators (say up to degree 5) for functions up to degree 30 (without using Heuristic 1 for a good affine transformation to get h, but solving the matrix UA^r available directly from the function f). We are currently working in this direction.

It should be noted that after running our heuristic on random balanced functions, the improvement is not extremely significant. There are improvements as we find that the the values are significantly more than $\frac{1}{2} \sum_{i=0}^{d} \binom{n}{i}$ (making our algorithm efficient), but it is actually not very close to $\sum_{i=0}^{d} \binom{n}{i}$. It seems that it is not a problem with the efficiency of the heuristic, but with the inherent property of a random Boolean function that there may not be an affine transformation at all on $f(x)$ such that $|\{x|f(Bx + b) = 1, wt(x) \leq d\}|$ is very high. In fact we can show that for highly nonlinear functions $f(x)$, the increment from $|\{x|f(x) = 1, wt(x) \leq d\}|$ to $|\{x|f(Bx + b) = 1, wt(x) \leq d\}|$ may not be significant for any B, b. The reason for this is as follows.

Let $f \in B_n$ be a balanced function (n odd) having nonlinearity $nl(f) = 2^{n-1} - 2^{\frac{n-1}{2}}$ (see [17] for definition of nonlinearity) and consider that it has no annihilator at a degree $\leq \frac{n-1}{2}$. Let $g \in B_n$ be the function such that $g(x) = 1$ for $wt(x) \leq \frac{n-1}{2}$. By Theorem 3 in [19], $nl(g) = 2^{n-1} - \binom{n-1}{\frac{n-1}{2}}$. Now we like to find out a function $h(x) = f(Bx + b)$ such that $|\{x|h(x) = 1, wt(x) \leq \frac{n-1}{2}\}|$ is high. Consider the value $T = |supp(g) \cap supp(h)|$, i.e., $T = |\{x : h(x) = 1 \ \& \ wt(x) \leq \frac{n-1}{2}\}|$. Without loss of generality consider $T \geq 2^{n-2}$. Hence, $d(h, g) = 2(2^{n-1} - T) = 2^n - 2T$. Now, $nl(f) = nl(h) \leq nl(g) + d(h, g) = (2^{n-1} - \binom{n-1}{\frac{n-1}{2}}) + 2^n - 2T$. Thus, $2^{n-1} - 2^{\frac{n-1}{2}} \leq (2^{n-1} - \binom{n-1}{\frac{n-1}{2}}) + 2^n - 2T$, i.e., $T \leq 2^{n-1} - (\frac{1}{2}\binom{n-1}{\frac{n-1}{2}} - 2^{\frac{n-1}{2}-1})$. It is the problem that we can not get any h from f such that $|\{x|h(x) = 1, wt(x) \leq \frac{n-1}{2}\}|$ is arbitrarily close to 2^{n-1}.

Now consider the dimension of the matrix UA^r for the function h, which is $\nu_h \times (\sum_{i=0}^{\frac{n-1}{2}} \binom{n}{i} - \mu_h)$. Now $\sum_{i=0}^{\frac{n-1}{2}} \binom{n}{i} - \mu_h = 2^{n-1} - T \geq \frac{1}{2}\binom{n-1}{\frac{n-1}{2}} - 2^{\frac{n-1}{2}-1}$. Thus the dimension of the matrix UA^r is at least $(\frac{1}{2}\binom{n-1}{\frac{n-1}{2}} - 2^{\frac{n-1}{2}-1}) \times (\frac{1}{2}\binom{n-1}{\frac{n-1}{2}} - 2^{\frac{n-1}{2}-1})$.

In this direction we present the following technical result where the constraint of nonlinearity is removed.

Proposition 2 *Suppose $f \in B_n$ be a randomly chosen balanced function. Then the probability to get an affine transformation such that $|\{x|f(Bx+b) = 1, wt(x) \leq \lfloor \frac{n-1}{2} \rfloor\}| > \sum_{i=0}^{\lfloor \frac{n-1}{2} \rfloor} \binom{n}{i} - k$ is*

1. $< \frac{(n+1)2^n \sum_{i=0}^{k-1} \binom{2^{n-1}}{i}^2}{\binom{2^n}{2^{n-1}}}$ *for n odd.*

2. $< \frac{(n+1)2^n \sum_{i=0}^{k-1} \binom{\sum_{j=0}^{\frac{n}{2}-1}\binom{n}{j}}{i} \binom{2^n - \sum_{j=0}^{\frac{n}{2}-1}\binom{n}{j}}{i+\frac{1}{2}\binom{n}{\frac{n}{2}}}}{\binom{2^n}{2^{n-1}}}$ *for n even.*

Proof: First we prove it for n odd. The number of functions $h \in B_n$ such that $|\{x|h(x) = 1, wt(x) \leq \frac{n-1}{2}\}| > 2^{n-1} - k$ is $\sum_{i=0}^{k-1} \binom{2^{n-1}}{i}^2$ (consider the upper and lower half in the truth table of the function). So, there will be at most $\sum_{i=0}^{k-1} \binom{2^{n-1}}{i}^2$ many affinely invariant classes of such functions. Further the total number of balanced function is $\binom{2^n}{2^{n-1}}$. Hence the total number of affinely invariant classes of balanced function is $\geq \frac{\binom{2^n}{2^{n-1}}}{2^n(2^n-1)(2^n-2^1)\ldots(2^n-2^{n-1})} > \frac{\binom{2^n}{2^{n-1}}}{(n+1)2^n}$. Hence the probability of a randomly chosen balanced function will be function type h is bounded by $\frac{(n+1)2^n \sum_{i=0}^{k-1} \binom{2^{n-1}}{i}^2}{\binom{2^n}{2^{n-1}}}$. Similarly, the case for n even can be proved. ■

If one takes $k = 2^{n-3}$, then it can be checked easily that the probability decreases fast towards zero as n increases. Thus for a random balanced function f, the probability of getting an affine transformation (which generates the function h from f) such that $|\{x|f(Bx+b) = 1, wt(x) \leq \lfloor \frac{n-1}{2} \rfloor\}| > \sum_{i=0}^{\lfloor \frac{n-1}{2} \rfloor} \binom{n}{i} - 2^{n-3}$ is almost improbable.

Thus when one randomly chosen balanced function is considered, using the strategy of considering the function after affine transformation, one can indeed improve the performance of the algorithm, but the improvement may not be very significant in asymptotic terms when the annihilators at the degree of $\lfloor \frac{n-1}{2} \rfloor$ are considered. We are currently studying that given some n what is the range of d such that some heuristic (may be some changes over Heuristic 1) provides significant improvement.

5 Conclusion

In this paper we have studied how can we present an efficient algorithm to find annihilator of a Boolean function by solution of homogeneous linear equations. We have clearly shown asymptotic improvements over the existing algorithms for the construction of the system of homogeneous linear equations. We have also clearly identified how the dimension of the matrix (to be analysed) can be efficiently reduced. Towards the negative side, we point out that it may not be possible to reduce the dimension of the matrix for a randomly chosen balanced Boolean function to a great extent.

References

[1] F. Armknecht, C. Carlet, P. Gaborit, S. Kuenzli, W. Meier and O. Ruatta. Efficient computation of algebraic immunity for algebraic and fast algebraic attacks. Accepted in Eurocrypt 2006.

[2] F. Armknecht. Improving Fast Algebraic Attacks. In *FSE 2004*, number 3017 in Lecture Notes in Computer Science, pages 65–82. Springer Verlag, 2004.

[3] L. M. Batten. Algebraic Attacks over GF(q). In *Progress in Cryptology - INDOCRYPT 2004*, pages 84–91, number 3348, Lecture Notes in Computer Science, Springer-Verlag.

[4] A. Botev. On algebraic immunity of some recursively given sequence of correlation immune functions. In Proceedings of *XV international workshop on Synthesis and complexity of control systems*, Novosibirsk, October 18-23, 2004, pages 8-12 (in Russian).

[5] A. Botev. On algebraic immunity of new constructions of filters with high nonlinearity. In Proceedings of *VI international conference on Discrete models in the theory of control systems*, Moscow, December 7-11, 2004, pages 227-230 (in Russian).

[6] A. Botev and Y. Tarannikov. Lower bounds on algebraic immunity for recursive constructions of nonlinear filters. Preprint 2004.

[7] A. Braeken and B. Praneel. Probabilistic algebraic attacks. Accepted in 10th IMA international conference on cryptography and coding, 2005.

[8] A. Braeken and B. Praneel. On the Algebraic Immunity of Symmetric Boolean Functions. Accepted in Indocrypt 2005. Cryptology ePrint Archive, http://eprint.iacr.org/, No. 2005/245, 26 July, 2005.

[9] A. Canteaut. Open problems related to algebraic attacks on stream ciphers. In *WCC 2005*, pages 1–10, invited talk.

[10] C. Carlet. Improving the algebraic immunity of resilient and nonlinear functions and constructing bent functions. IACR ePrint server, http://eprint.iacr.org, 2004/276.

[11] J. H. Cheon and D. H. Lee. Resistance of S-boxes against Algebraic Attacks. In *FSE 2004*, number 3017 in Lecture Notes in Computer Science, pages 83–94. Springer Verlag, 2004.

[12] J. Y. Cho and J. Pieprizyk. Algebraic Attacks on SOBER-t32 and SOBER-128. In *FSE 2004*, number 3017 in Lecture Notes in Computer Science, pages 49–64. Springer Verlag, 2004.

[13] D. Coppersmith and S. Winograd. Matrix multiplication via arithmatic programming. *Journal of Symbolic Computation*, (9):251-280, 1990.

[14] N. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In *Advances in Cryptology - ASIACRYPT 2002*, number 2501 in Lecture Notes in Computer Science, pages 267–287. Springer Verlag, 2002.

[15] N. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology - EUROCRYPT 2003*, number 2656 in Lecture Notes in Computer Science, pages 345–359. Springer Verlag, 2003.

[16] N. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology - CRYPTO 2003*, number 2729 in Lecture Notes in Computer Science, pages 176–194. Springer Verlag, 2003.

[17] D. K. Dalai, K. C. Gupta and S. Maitra. Results on Algebraic Immunity for Cryptographically Significant Boolean Functions. In *INDOCRYPT 2004*, pages 92–106, number 3348, Lecture Notes in Computer Science, Springer-Verlag.

[18] D. K. Dalai, K. C. Gupta and S. Maitra. Cryptographically Significant Boolean functions: Construction and Analysis in terms of Algebraic Immunity. In *FSE 2005*, pages 98–111, number 3557, Lecture Notes in Computer Science, Springer-Verlag.

[19] D. K. Dalai, S. Maitra and S. Sarkar. Basic Theory in Construction of Boolean Functions with Maximum Possible Annihilator Immunity. Cryptology ePrint Archive, http://eprint.iacr.org/, No. 2005/229, 15 July, 2005.

[20] D. H. Lee, J. Kim, J. Hong, J. W. Han and D. Moon. Algebraic Attacks on Summation Generators. Lecture Notes in Computer Science, pages 34–48. Springer Verlag, 2004.

[21] W. Meier, E. Pasalic and C. Carlet. Algebraic attacks and decomposition of Boolean functions. In *Advances in Cryptology - EUROCRYPT 2004*, number 3027 in Lecture Notes in Computer Science, pages 474–491. Springer Verlag, 2004.