Computing the Rank of Incidence Matrix and the Algebraic Immunity of Boolean Functions

Deepak Kumar Dalai
School of Mathematical Sciences,
NISER, Bhubaneswar, INDIA-751005
Email: deepak@niser.ac.in

Abstract

The incidence matrix between a set of monomials and a set of vectors in $\mathbb{F}_2$ has a great importance in the study of coding theory, cryptography, linear algebra, combinatorics. The rank of these matrices are very useful while computing algebraic immunity(AI) of Boolean functions in cryptography literature [18, 7]. Moreover, these matrices are very sparse and well structured. Thus, for aesthetic reason finding the rank of these matrices is also very interesting in mathematics. In this paper, we have reviewed the existing algorithms with added techniques to speed up the algorithms and have proposed some new efficient algorithms for the computation of the rank of incidence matrix and solving the system of equations where the co-efficient matrix is an incidence matrix. Permuting the rows and columns of the incidence matrix with respect to an ordering, the incidence matrix can be converted to a lower block triangular matrix, which makes the computation in quadratic time complexity and linear space complexity. Same technique is used to check and computing low degree annihilators of an n-variable Boolean functions in faster time complexity than the usual algorithms. Moreover, same technique is also exploited on the Dalai-Maitra algorithm in [9] for faster computation. On the basis of experiments, we conjecture that the AI of n-variable inverse S-box is $\lfloor\sqrt{n}\rfloor + \lceil\frac{n}{\lfloor\sqrt{n}\rfloor}\rceil - 2$. We have also shown the skepticism on the existing fastest algorithm in [1] to find AI and lowest degree annihilators of a Boolean function.

Keywords: Boolean function, algebraic immunity, rank of matrix, LU-decomposition.

1 Notations

In this section, we introduce the basic notation and definitions which are useful to read the later part of the article.

$\mathbb{F}_2$: The finite field on two elements i.e., $GF(2)$.

V_n: The n dimensional vector space over $\mathbb{F}_2$. The vectors of V_n are represented in terms of its standard basis $(1, 0, \ldots, 0), \ldots, (0, \ldots, 0, 1)$.

$\mathsf{wt}(v)$: The weight of a vector $v = (v_1, \ldots, v_n) \in V_n$, is defined as $\mathsf{wt}(v) = |\{v_i : v_i = 1\}|$.

$u \subseteq v$: For two vectors $u = (u_1, \ldots, u_n)$ and $v = (v_1, \ldots, v_n)$, we define $u \subseteq v$ if $u_i = 1$ implies $v_i = 1$ for $1 \leq i \leq n$.

$\sum$ and $+$: The sum notation, are used as context based, whether it is over $\mathbb{F}_2$ or, over real field $\mathbb{R}$.

P_n : The binary quotient polynomial ring on n-variables $\mathbb{F}_2[x_1, x_2, \ldots, x_n]/\langle x_1^2 - x_1, x_2^2 - x_2, \ldots, x_n^2 - x_n\rangle$.

x^α: The polynomials of the form $x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$ for $\alpha_i \in \{0, 1\}, 1 \leq i \leq n$ are called monomials, which are represented as x^α where $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in V_n$. Monomials are also represented as $x_{i_1} x_{i_2} \ldots x_{i_k}$ where $\alpha_{i_1} = \alpha_{i_2} = \cdots = \alpha_{i_k} = 1$ and other α_i's are 0. Each polynomial from P_n can be represented as $\sum\limits_{\alpha \in V_n} a_\alpha x^\alpha$, where $a_\alpha \in \mathbb{F}_2$. P_n is a vector space over $\mathbb{F}_2$ with the monomial basis $\{x^\alpha : \alpha \in V_n\}$.

$\deg(p)$: The degree of a polynomial $p = \sum\limits_{\alpha \in V_n} a_\alpha x^\alpha \in P_n$ is defined by $\deg(p) = \max\{\mathsf{wt}(\alpha) : a_\alpha = 1\}$.

$P_{n,d}$: The subspace of polynomials of degree at most d with the monomial basis $B_{n,d} = \{x^\alpha : \mathsf{wt}(\alpha) \leq d\}$.

$\mathsf{log}(X)$: For a set of monomials X, $\mathsf{log}(X)$ denotes the set of exponent vectors of the monomials i.e., $\mathsf{log}(X) = \{\alpha : x^\alpha \in X\}$.

x^V: For a set of vectors V, x^V denotes the set of monomials with exponents from V i.e., $x^V = \{x^\alpha : \alpha \in V\}$.

Evaluation of Polynomial: A vector $v \in V_n$ satisfies a monomial x^α if $\alpha \subseteq v$. A polynomial $p = \sum\limits_{\alpha \in V_n} a_\alpha x^\alpha \in P_n$ is evaluated at a vector $v \in V_n$ as $p(v) = \sum\limits_{\alpha \in V_n, \alpha \subseteq v} a_\alpha \in \mathbb{F}_2$.

Ordering of monomials and vectors: A tricky ordering of vectors and monomials can speed up the computation. Here, we define some orderings of vectors in V_n. If $u, v \in V_n$, then

1. $u < v$ if $\mathsf{int}(u) < \mathsf{int}(v)$ where $\mathsf{int}(u)$ is the integer value of the binary string representation of u.

2. $u <_w v$ if $(\mathsf{wt}(u) < \mathsf{wt}(t))$ or, $(\mathsf{wt}(u) = \mathsf{wt}(v)$ and $\mathsf{int}(u) < \mathsf{int}(v))$.
3. Given a set of monomials X, $u <_X v$ if $(u, v \in \mathsf{log}(X)$ and $u < v)$ or, $(u, v \notin \mathsf{log}(X)$ and $u < v)$ or, $(u \in \mathsf{log}(X)$ and $v \notin \mathsf{log}(X))$.

Abusing the notation, we also use the same notation for the monomial ordering as

1. $x^u < x^v$ if $u < v$.
2. $x^u <_w x^v$ if $u <_w v$
3. Given a set of vectors V, $x^u <_V x^v$ if $(u, v \in V$ and $u < v)$ or, $(u, v \notin V$ and $u < v)$ or, $(u \in V$ and $v \notin V)$.

Incidence matrix (M_V^X): A vector $v \in V_n$ is incident on a polynomial $p \in P_n$ if $p(v) = 1$. The incidence matrix M_V^X for an ordered set of monomials $X \subseteq P_n$ and an ordered set of vectors $V \subseteq V_n$ is defined as $M_V^X[i, j] = X_j(v_i)$ where X_j and v_i are j-th and i-th element of the ordered sets X and U respectively. If $X = \{x^{\alpha_1}, \ldots, x^{\alpha_m}\}$ is an ordered set of monomials, then the incidence matrix can be defined as

$$M_V^X[i, j] = \begin{cases} 1 \text{ if } \alpha_j \subseteq v_i \\ 0 \text{ Otherwise.} \end{cases}$$

Incidence matrix (M_V^d): If $X = B_{n,d}$ with an ordering, then we denote the incidence matrix for a set of vectors V as M_V^d instead of lengthy notation $M_V^{B_{n,d}}$.

Boolean function: The polynomials from P_n are also called Boolean functions on n-variables. The form of the polynomial defined above is called *algebraic normal form* (ANF) of Boolean functions. The evaluations of the polynomial $p \in P_n$ at each vector of V_n with an order is known as *truth table* representation of p. The truth table representation of a polynomial can be viewed as a 2^n-tuple binary vector and its weight is defined as $\mathsf{wt}(p) = |\{v \in V_n : p(v) = 1\}|$. A polynomial $p \in P_n$ is called balanced if $\mathsf{wt}(p) = 2^{n-1}$. The support set of p is defined as $S(p) = \{v \in V_n : p(v) = 1\}$. One may refer to [6] for the standard cryptographic definitions related to Boolean functions. In this article, we use the term polynomial in place of Boolean function.

Annihilator: Given $p \in P_n$, a nonzero polynomial $q \in P_n$ is called an annihilator of p if $p * q = 0$, i.e., $p(v)q(v) = 0$ for all $v \in V_n$. The set of all annihilators of $p \in P_n$ is denoted by $An(p)$.

Algebraic immunity (AI): Algebraic immunity of a polynomial p is defined as $\mathsf{AI}(p) = min\{\deg(q) : q \in An(p) \cup An(1 + p)\}$. In some article, algebraic immunity is mentioned as annihilator immunity.

$\mathsf{wt}(M), \mathsf{den}(M)$: For a $m \times n$ binary matrix M, the weight and density of M are defined as $\mathsf{wt}(M) = |\{M[i][j] : M[i][j] = 1\}|$ and $\mathsf{den}(M) = \frac{\mathsf{wt}(M)}{mn}$ respectively.

2 Introduction

The incidence matrix M_V^X is an interesting tool in the study of combinatorics, coding theory, cryptography and polynomial interpolation. In coding theory, polynomials of degree at most d form a Reed-Muller code of order d of length 2^n. The matrix $M_{V_n}^d$ is the transpose of the generator matrix for the Reed-Muller code of length 2^n and order d [16]. Hence, the matrix M_V^d is the transpose of the restricted generator matrix for the Reed-Muller code of length 2^n and order d to the set V. A generalized version of the matrix M_V^d is used for polynomial interpolation in Guruswami-Sudan list decoding technique for Reed-Solomon code [17]. The matrix M_V^d can also be treated as a generalized Vandermonde matrix in the study of combinatorics.

Moreover, the incidence matrix M_V^d has a great importance in the study of algebraic cryptanalysis. It is related to algebraic immunity, for which the rank of this matrix is very important [18]. Some algorithms have been given in the literature to find the rank of M_V^d and finding solution of the system of equations $M_V^d \gamma = 0$ [18, 12, 13, 1, 9], which gives the annihilators of degree d of polynomials of support set V.

Algebraic attacks have received a lot of attention in studying the security of crypto systems [6]. For some keystream generators, algebraic attacks worked very well comparatively to all other known attacks. Particularly, algebraic attack using annihilators [5, 18] are highly effective on keystream generators like LFSR based nonlinear combiner and filter models.

From the point of view of algebraic cryptanalysis, a polynomial p should not be used to design a cryptosystem if $An(p) \cup An(1+p)$ contains low degree polynomials [5, 18]. The term algebraic immunity of a polynomial p, $\mathsf{AI}(p)$, is defined so. It is known that for any polynomial $p \in P_n$, $\mathsf{AI}(p) \leq \lceil \frac{n}{2} \rceil$ [18]. Thus, the target of a good design is to use a polynomial p such that neither p nor $1+p$ has an annihilator at a degree less than $\lceil \frac{n}{2} \rceil$. There is a need to construct such polynomials and the first one in this direction appeared in [8]. Later some more constructions with this property have been presented in [2, 3, 10, 15].

If $q \in P_n$ is an annihilator of $p \in P_n$ then $q(v) = 0$ for $v \in S(p)$. To find an annihilator $q \in P_{n,d}$, one has to solve the system linear equations

$$\sum_{\alpha \in V_n, \mathsf{wt}(\alpha) \leq d, \alpha \subseteq v} a_\alpha = 0 \text{ for } v \in S(p).$$

That is,

$$M_{S(p)}^d \gamma = 0 \tag{1}$$

where transpose of γ is the unknown row vector (a_α), for $\alpha \in V_n$ and $\mathsf{wt}(\alpha) \leq d$. To check the existence of d or lesser degree annihilator of p, one has to check whether the rank of matrix $M^d_{S(p)}$ is $|P_{n,d}| = \sum_{i=0}^{d} \binom{n}{i}$. In this article, we discuss the rank of the matrix M^X_V for an order sets of vectors V and monomials X, with more attention on the special case $M^d_{S(p)}$.

For an ordered set of vectors V and an ordered set of monomials X, the matrix M^X_V carries many structures compared to a random binary matrix of same dimension. Some of the structures are discussed as follows.

1. Each column of M^X_V 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 from X on the vectors from V. Hence, there is one-to-one correspondence from the vectors $v \in V_n$ to the row vectors of length $|X|$. All the information in each row of length $|X|$ can be algebraically retrieved by the corresponding vector of length n. This property can be used to find out the value at any positions instead of travelling all the entries of a rows/columns. In the case of $M^d_{S(p)}$, each row is an evaluation of a d or lesser degree monomial at a support vector of p. The information in $wt(p) \times |B_{n,d}|$ matrix $M^d_{S(p)}$ can be retrieved from the $wt(p) \times n$ matrix $M^1_{S(p)}$. If this algebraic property can be used, the algorithm may also take less than the quadratic time complexity on the number of monomials. The strategy of polynomial interpolation has been exploited to decrease the complexity in the paper [1], though some errors in this algorithm are detiled here.

2. Let $V \subset V_n$ and $X \subset B_{n,n}$ be randomly chosen subsets such that $|V| = |X| = 2^{n-1}$. Since a vector $v \in V$ of weight i is expected to be satisfied by 2^{i-1} monomials from X, the $\mathsf{wt}(M^X_V)$ is expected around $w = \frac{1}{2}\sum_{i=0}^{n}\binom{n}{i}2^{i-1} = \frac{1}{4}\sum_{i=0}^{n}\binom{n}{i}2^{i} = \frac{(1+2)^n}{4} = \frac{3^n}{4}$. So, the $\mathsf{den}(M^X_V) = \frac{w}{2^{2n-2}} = (\frac{3}{4})^n$ tends to zero as n tends to infinity, where as the density is expected to be $\frac{1}{2}$ for a random matrix. The matrix is very sparse. Hence, sparse matrix algorithms can be used for the purpose [21, 14, 11].

   A vector v of weight i is satisfied by $\sum_{i=0}^{d}\binom{n}{i}$ monomials of degree d or less. Hence, for a randomly chosen balanced $p \in P_n$, the $\mathsf{wt}(M^d_{S(p)})$ is expected around $w = \frac{1}{2}\sum_{i=0}^{n}(\binom{n}{i}\sum_{j=0}^{d}\binom{i}{j})$, where as the number of entries in $M^d_{S(p)}$ is $e = \frac{1}{2}\sum_{i=0}^{n}\binom{n}{i}\sum_{j=0}^{d}\binom{n}{j}$. The density $\mathsf{den}(M^d_{S(p)})$

(i.e., $\frac{w}{e}$) tends to zero for $d > 1$ and large n. In the following table we have put the values of the $\mathsf{den}(M^d_{S(p)})$ for some n and $d = \lfloor \frac{n}{2} \rfloor$.

Table 1: Sparseness of $M^d_{S(p)}$

n	11	12	13	14	15	16
d	5	5	6	6	7	7
$\mathsf{den}(M^d_{S(p)})$	.0742	.0673	.0426	.0383	.0244	.0218

3. If the degree of a monomial is higher, then the evaluation of the monomial at a vector has low chance to be non-zero. If the monomials are ordered by $<_w$, then it can be seen that each row gets sparser towards the right end because the degree of monomials increases as we move towards right end of the matrix. Therefore, the upper triangular part of M^X_V would be very sparse. M^X_V looks like a lower triangular matrix except a few non-zero entries at the upper triangular part. This sparseness can be exploited in a fast implementation. Similar sparseness can also be observed if the monomials and support vectors are ordered by $<$. We will use such sparseness structure in our algorithms in Section 5.

Therefore, solving Equation 1 can be faster as compared to solving an arbitrary system of equations of same dimension if the algebraic structures of M^X_V are carefully exploited. For example, in [9], some more structures have been exploited to make it constant time faster in average case.

In Section 3, we have studied some existing algorithms and proposed how the sparseness can be exploited to make them faster. In Section 4, we have shown the incorrectness of the ACGKMR algorithm proposed in [1]. In Section 5, we have proposed a technique on the ordering of vectors and monomials which makes the matrix $M^d_{S(p)}$ a lower block triangular. The Section 5.2 contains the main results of this article to reduce the computation time. Further, in Section 5.3, we use the same technique on the Dalai and Maitra's algorithm presented in [9] to make it even faster. Experimental results of some important exponent S-boxes are presented in Section 6. On the basis of experiments, we conjecture that the AI of n-variable inverse S-box is $\lfloor \sqrt{n} \rfloor + \lceil \frac{n}{\lfloor \sqrt{n} \rfloor} \rceil - 2$.

3 Basic Algorithms

In this section, we study the basic algorithms to find the rank of matrix M^X_V or, solving $M^X_V \gamma = 0$.

3.1 Technique 1

The most basic algorithm for finding the rank of M_V^X and solving $M_V^X\gamma = 0$ is by using the standard algorithms like Gaussian elimination, Strassen's method etc. This is equivalent to the algorithm mentioned in [18, Algorithm 1]. The theoretical bound of time complexity is $O(2^{\omega(n-1)})$. Using the well known Gaussian elimination technique we have $\omega = 3$; Strassen's algorithm [20] takes $\omega = \log_2 7 \approx 2.807$ and also the one by Coppersmith and Winogard in [4] takes $\omega = 2.376$. Since the matrix $M^d_{S(p)}$ is very sparse, in practice, it is more efficient than a random matrix of same size. To make it faster, one can also use some suitable sparse algorithms [21, 14, 11].

3.2 Technique 2

The evaluation of x^α at α is 1 (i.e., $x^\alpha(\alpha) = 1$) for $\alpha \in V_n$. While working for the matrix M_V^X, one can eliminate a_α for $\alpha \in V \cap \log(X)$ easily during Gaussian elimination process to increase the efficiency. This technique is used for the matrix $M^d_{S(p)}$ in [18, Algorithm 2] to find out annihilators. Here, we describe for a faster implementation of this technique exploiting the triangular nature and sparseness of some part of matrix M_V^X. Let the ordering of the monomials of $U = \log(X)$ and vectors of V be $<_V$ and $<_U$ respectively. Then, the form of matrix M_V^X is

$$M_V^X = \begin{pmatrix} A & B \\ C & D \end{pmatrix}.$$

Here the sub matrices are incidence matrices $A = M_W^W$, $B = M_W^Z$, $C = M_Y^W$ and $D = M_Y^Z$ where $W = V \cap U$, $Y = V \setminus W$ and $Z = U \setminus W$. The matrix A is lower triangular with diagonal entries are non-zero. This property helps to speed up the row operations for the rows of M_V^X associated with the sub-matrix A.

Now we will give attention on the matrix $M^d_{S(p)}$. The form of matrix $M^d_{S(p)}$ is

$$M^d_{S(p)} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}. \qquad (2)$$

Here the sub matrices are incidence matrices $A = M_W^W$, $B = M_W^Z$, $C = M_Y^W$ and $D = M_Y^Z$ where $W = S(p) \cap \mathsf{log}(B_{n,d})$, $Y = S(p) \setminus W$ and $Z = \mathsf{log}(B_{n,d}) \setminus W$. It is clear that the matrix A is lower triangular with nonzero diagonal entries. This property helps to speed up the row operations for the rows of $M^d_{S(p)}$ associated with the sub-matrix A. Since the incident vectors in W are of low weight (i.e., up to d), the sub-matrices A and B are very sparse. This nature, in addition to the lower triangularity of A, makes more efficient to find the rank of $M^d_{S(p)}$. For a random $p \in P_n$, $|W|$ and $|Z|$ are approximately $\frac{1}{2}\sum_{i=0}^{d}\binom{n}{i}$. So, $\mathsf{wt}(A)$ and

$\mathsf{wt}(B)$ are approximately $w = \frac{1}{4}\sum_{i=0}^{d}(\binom{n}{i}\sum_{j=0}^{d}\binom{i}{j}) \leq \frac{1}{4}\sum_{i=0}^{d}(\binom{n}{i}\sum_{j=0}^{d}\binom{d}{j})$ $= 2^{d-2}\sum_{i=0}^{d}\binom{n}{i}$. Then, the $\mathsf{den}(A)$ and $\mathsf{den}(B)$ are bounded by $\frac{2^{d-2}\sum_{i=0}^{d}\binom{n}{i}}{(\frac{1}{2}\sum_{i=0}^{d}\binom{n}{i})^2}$ $= \frac{2^d}{\sum_{i=0}^{d}\binom{n}{i}}$.

For $d = \lfloor\frac{n-1}{2}\rfloor$, we have around $2^{d-2} \times 2^{n-2}$ nonzero entries for the matrices A and B of size $2^{n-2} \times 2^{n-2}$. So $\mathsf{den}(A)$ and $\mathsf{den}(B)$ are bounded by $2^{d-n+1} = O(2^{-\frac{n}{2}}) = O(2^{-d})$. This sparseness in A and B can be further exploited to speed up the process. The algorithm can be implemented in two parts. At first, elementary row reduction can be done for the upper half of the matrix $M^d_{S(p)}$ and then rest (updated) part can be done using any usual technique. While doing elementary row operations in the upper part, one can explore only the positions of non-zero entries in the row instead of exploring all elements of the matrix. Hence, the reduction process can be made faster in the order of $O(2^{\frac{n}{2}})$ for the upper part of the matrix $M^d_{S(p)}$.

3.3 Technique 3

The discussion in this section refers to the algorithm for checking the rank of $M^d_{S(p)}$ described in [9]. Since the sub-matrix A in Equation 2 is a nonsingular lower triangular matrix, we solve for $M^d_{S(p)}$ in two steps in technique 3.2. At first step, the row reduction is done on the rows associated with W and, in the next step, the reduction is done on the updated rows associated with Y. The strategy described in [9] avoids the first step and directly works with the modified version of the matrix D. Here, we need to find the rank of a $|Y| \times |Z|$ matrix D' rather than to find the rank of a $(|W| + |Y|) \times (|W| + |Z|) = wt(p) \times |B_{n,d}|$ matrix $M^d_{S(p)}$. As [9], the matrix D' is computed as $D' = D'[u, \alpha]$ for $u \in Y$ and $\alpha \in Z$ such that

$$D'[u,\alpha] = \begin{cases} \displaystyle\sum_{i=0}^{d-\mathsf{wt}(\alpha)} \binom{\mathsf{wt}(u) - \mathsf{wt}(\alpha)}{i} \bmod 2 \text{ if } \alpha \subseteq u \\ 0 \text{ other wise.} \end{cases}$$

Given u, α with $\mathsf{wt}(u) > d$ and $\mathsf{wt}(\alpha) = l \leq d$, the probability that $\alpha \subseteq u$ is $\frac{\sum_{i=d+1}^{n}\binom{n-l}{i-l}}{\sum_{i=d+1}^{n}\binom{n}{i}}$ which is very less than .5 for $l > 0$. Further, even $v \subseteq u$, there is 50% chance that $D'[u, v] = 1$. Therefore, the matrix D' is very sparse and algorithms to find the rank of sparse matrix [21, 14, 11] can be used for the purpose.

4 *LU* decomposition and Algorithm in [1]

Let M be a square matrix. An LU decomposition is a factorization of M of the form $M = LU$, where L and U are lower and upper triangular matrices of the same size respectively. LU decomposition is a handy tool in several fundamental algorithms in linear algebra such as solving a system of linear equations, inverting a matrix, or computing the determinant of a matrix.

In this section, we discuss about the algorithm presented in [1] to find the rank of $M^d_{S(p)}$ and to find the solutions of $M^d_{S(p)}\gamma = 0$. For the reference, we call this algorithm as ACGKMR algorithm. The algorithm exploits the LU decomposition of $M^d_{S(p)}$ for the purpose and is claimed as a quadratic time complexity on the number of columns (i.e, the number of monomials). Here, we have shown that the algorithm is wrong for different reasons.

4.1 Equivalence between Solving a System of Linear Equations and Finding Affine Annihilators

The problems of finding the solutions of a system of linear equations, finding the rank of a matrix, inverting of a nonsingular matrix are considered as equivalent problems in linear algebra. These problems on $m \times m$ matrix can be solved in $O(n^\omega)$ time complexity, where the known lowest value of ω is 2.376. Since a general matrix needs $O(n^2)$ memory for its representation i.e., the space complexity, by any strategy the value of ω can not be less than 2. Consider P is an another problem which takes at most quadratic time complexity on the size of problem. If the problem of solving system of linear equations can be reduced to the problem P in $O(m^2)$ time complexity then one can solve a system of linear equations in $O(m^2)$ time complexity. In the following part, we have shown that finding the solutions of a system of linear equations on $\mathbb{F}_2$ is not harder than finding the affine annihilators of a Boolean function.

Theorem 1. *The problem of finding the solutions of a system of m linear homogeneous equations on m variables on $\mathbb{F}_2$ can be reduced to the problem of finding the affine annihilators of a polynomial in P_{m-1} in $O(m^2)$ time complexity.*

Proof. Consider M is the $m \times m$ coefficient matrix of the system of linear equations. Without loss of generality, we consider that the first column of M is not all zero column. Since the first column is nonzero, there must be a row (say, k-th row) whose 1st entry is 1. Now entry wise adding (over $\mathbb{F}_2$) kth row with all other rows whose 1st entry is 0, we can make the first column all 1's. This operation takes $O(m^2)$ complexity. We keep the same name M for the updated matrix after these row operations. Now we construct $p_M \in P_{m-1}$ where $S(p_M) = \{(M_{i,2}, M_{i,3}, \ldots, M_{i,m}), 1 \leq i \leq m\}$ i.e., the vectors formed by last $(m-1)$ entries of each row. Here $wt(p_M) = m$.

The matrix $M^1_{S(p_M)}$ is same as M. Therefore, the coefficients of 1-degree annihilators of p_M give the solutions of M. □

Example 1. *Consider* $M = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$*, a* 5×5 *coefficient matrix of a system of homogeneous linear equations. To make its first column all* 1*, we add either the* 1*st row or the* 3*rd row with the* 2*nd,* 4*th and* 5*th rows. Adding the* 1*st row, we have the updated matrix*

$$M = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

We construct a polynomial $p_M \in p_4$ *where* $S(p_M) = \{(0,1,1,0),(0,1,0,1),(0,0,0,1),(1,1,0,1),(0,0,1,0)\}$*. It can be easily verified that the matrix* $M^1_{S(p_M)}$ *i.e., the coefficient matrix of the system of equations* $a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = 0$ *for* $(x_1, x_2, x_3, x_4) \in S(p_M)$ *is same as* M*.*

Since finding annihilators of a polynomial is reduced to find the solutions of a system of linear equations, we have the following proposition.

Proposition 1. *The problem of finding the solutions of a system of linear equations on* $\mathbb{F}_2$ *is equivalent to the problem of finding the affine annihilators of a Boolean function.*

For an random $p \in P_n$, the matrix $M^1_{S(p)}$ can be realized as a random matrix except the first column. The reason is that each column evaluated by the linear monomials x_i are independent to each other. Hence both problems reduce to each other. When $d > 1$, i.e., searching for the d-degree annihilators, the matrix $M^d_{S(p)}$ seems different than an arbitrary matrix. The columns corresponding to the nonlinear monomials are algebraically dependent on the columns of lower degree monomials. Hence, it does not seem the problem of finding solution of system linear equations can be reduced to searching for non-affine annihilators. But it is not proved yet. It is an open problem.

Therefore, if there is an algorithm to find 1-degree annihilator of $p \in P_n$ in $O(n^2)$ time complexity, then a binary system of linear equations can be solved in $O(n^2)$ time complexity. Note that the quadratic complexity is the least complexity as one needs $O(n^2)$ space to represent the matrix. The ACGKMR algorithm proposed in Eurocrypt 2006 [1], which requires quadratic time complexity on the number of monomials to find out the lowest degree annihilator of a polynomial. Therefore, *"solving a system of*

n linear equations on n-variables requires $O(n^2)$ running time complexity". This result would be a great contribution to the study of linear algebra. Therefore, we got a big doubt on the correctness of ACGKMR algorithm and the mistake is described in the following subsection to stand with our doubt.

4.2 LU Decomposition

If M is a nonsingular matrix then there is a permutation matrix P such that $PM = LU$ where L and U are nonsingular lower and upper diagonal matrices respectively. If $p \in P_n$ having no annihilator of degree d then the vectors of $S(p)$ can be ordered in such a way that $M^d_{S(p)} = LU$ where L and U are nonsingular lower and upper diagonal matrices respectively. Once $M^d_{S(p)}$ is factorized into LU, solving $M^d_{S(p)}\gamma = LU\gamma = 0$ can be solved in quadratic time complexity. This technique is exploited in ACGKMR algorithm to find the annihilators of $M^d_{S(p)}$ which is briefly described in the following paragraphs.

Let $\{\alpha_1, \alpha_2, \ldots, \alpha_D\}$ be the set of exponent vectors of weight at most d with an ordering and $S = S(p)$. ACGKMR algorithm is iterative in nature and starts with a matrix $M_1 = (v_1^{\alpha_1})$, where $v_1 \in S$ is chosen in such a way that $v_j^{\alpha_1} = 1$. Then the LU decomposition is done as $M_1 = (1)(1)$ and $S = S \setminus \{v_1\}$. Let the iteration for LU-decomposition be done till i-th step. For the $i+1$-th step we have the following processing.

$$\begin{aligned} M_{i+1} &= \begin{pmatrix} M_i & C_i \\ R_i & v_{i+1}^{\alpha_{i+1}} \end{pmatrix} \\ &= \begin{pmatrix} L_i & 0 \\ R_iU_i^{-1} & 1 \end{pmatrix} \begin{pmatrix} U_i & L_i^{-1}C_i \\ 0 & v_{i+1}^{\alpha_{i+1}} + R_iU_i^{-1}L_i^{-1}C_i \end{pmatrix} \\ &= L_{i+1}U_{i+1} \end{aligned}$$

where $C_i = (v_1^{\alpha_{i+1}} \ldots v_i^{\alpha_{i+1}})^t$, $R_i = (v_{i+1}^{\alpha_1} \ldots v_{i+}^{\alpha_i})$ and $v_{i+1} \in S$ is chosen in such a way that the rank of $M_{i+1} = i+1$ i.e., $v_{i+1}^{\alpha_{i+1}} + R_iU_i^{-1}L_i^{-1}C_i = 1$. Considering the output of $R_iU_i^{-1}L_i^{-1}C_i = 1$ is uniform, the probability of getting such $v_{i+1} = 1$ is $\frac{1}{2}$.

In the ACGKMR algorithm, the term $v_{i+1}^{\alpha_{i+1}} + R_iU_i^{-1}L_i^{-1}C_i$ is computed (at the last paragraph of page no. 153 [1]) in a strange way without any proper explanation, i.e., $v_{i+1}^{\alpha_{i+1}} - \sum_{j=1}^{i} v_{i+1}^{\alpha_1}.P_{i+1,j}$ where $P_{i+1,j}$ is the jth coordinate of $P_{i+1} = (L_i^{-1}C_i)^t$. Hence $R_iU_i^{-1}$ should be same as $(v_{i+1}^{\alpha_1}, \ldots, v_{i+1}^{\alpha_1})$, which can be easily verified that it can not be true. Even if we consider that there is a typing mistake, we believe that $R_iU_i^{-1}$ can not be written as a so simple expression. Hence it is another reason for not trusting the algorithm.

Now, we shall discuss about the obstacles present in the faster computation for LU decomposition of $M^d_{S(p)}$. Here, faster computation we mean quadratic time complexity computation i.e., $O(2^{2n})$. During the process we face 3 computations, i.e., $R_iU_i^{-1}$, $L_i^{-1}C_i$ and $R_iU_i^{-1}L_i^{-1}C_i$ to be made faster. If $R_iU_i^{-1}$ and $L_i^{-1}C_i$ are available, then $R_iU_i^{-1}L_i^{-1}C_i$ can be computed in $O(2^n)$ time. Further, if U_i^{-1} and L_i^{-1} are known priorly then $R_iU_i^{-1}$ and $L_i^{-1}C_i$ can be computed in $O(2^{n+wt(v_i)})$ and $O(2^{2n-wt(\alpha_i)})$ as weight of R_i and C_i are at most $2^{wt(v_i)}$ and $2^{n-wt(\alpha_i)}$ respectively. The computation of L^{-1} and U^{-1} can be computed recursively. But we do not find any technique to make this computation less than the quadratic time complexity. There may exist some other hidden way, but computing U_i^{-1} and L_i^{-1} efficiently is still an unsolved task for the purpose.

The Table 1 in [1] also contains some errors. It is known that the Kasami function in n-variables have exponents of the form $2^{2k} - 2^k + 1$ with $\gcd(n, k) = 1$. Therefore, the degree of Kasami function is $k+1$. In [1, Table 1], the exponent of Kasami function on 14 and 15 variables is written as $4033 = 2^{2*6} - 2 * 6 + 1$ where as $\gcd(14, 6) \neq 1 \neq \gcd(15, 6)$. Moreover, the degree of Kasami exponent on $12, 16$ and 20 variables are supposed to be $6, 8$ and 10 as $k = 5, 7$ and 9 respectively.

5 Lower-block triangularity of M_V^X

An $n \times m$ matrix M is called a lower-block triangular matrix if the structure of M is as follows.

$$M = \begin{pmatrix} M_{11} & M_{12} & \dots & M_{1l} \\ M_{21} & M_{22} & \dots & M_{2l} \\ \dots & \dots & \ddots & \dots \\ M_{l1} & M_{l2} & \dots & M_{ll} \end{pmatrix} \tag{3}$$

where M_{ij} are $n_i \times m_j$ sub-matrices for $1 \leq i, j \leq l$ with $\sum_{i=0}^{l} n_i = n$ and $\sum_{j=0}^{l} m_j = m$ and $M_{i,j}$ are zero sub-matrices for $j > i$.

5.1 Ordering $<_w$

Let the monomials in $B_{n,n}$ and vectors in V_n be ordered by $<_w$. Consider a set of monomials $X \subseteq B_{n,n}$ and a set of vectors $V \subseteq V_n$. Let $X^0, X^1, \dots, X^n$ be disjoint subsets of X, partitioned on the degree of monomials. The set X^i contains all the monomials of degree i from X. If $x^\alpha \in X^i$, $x^\beta \in X^j$ and $i < j$ then $x^\alpha <_w x^\beta$ and $\alpha <_w \beta$.

Similarly, the vector set V is partitioned by the weight of vectors and are denoted by $V^0, V^1, \dots, V^n$. If $v \in V^i$, $x^\alpha \in X^j$ and $i < j$, it is clear that $v <_w \alpha$ and $\alpha \nsubseteq v$. Hence, we have the following theorem.

Theorem 2. *The incidence matrix M_V^X is a lower block triangular matrix with $M_{ij} = M_{V^i}^{X^j}$ on the ordering $<_w$ of elements of V and X.*

Example 2. *Let X be the set of monomials on 4-variables such that* $\mathsf{log}(X) = \{0, 2, 3, 4, 8, 9, 14, 15\}$ *and set of vectors* $V = \{0, 3, 4, 5, 7, 9, 12, 15\}$*. The vectors are shown in their integer form. Then,* $\mathsf{log}(X^0) = \{0\}, \mathsf{log}(X^1) = \{2, 4, 8\}, \mathsf{log}(X^2) = \{3, 9\}, \mathsf{log}(X^3) = \{14\}, \mathsf{log}(X^4) = \{15\}$ *and* $V^0 = \{0\}, V^1 = \{4\}, V^2 = \{3, 5, 9, 12\}, V^3 = \{7\}, V^4 = \{15\}$*. Then the matrix*

$$M_V^X = \left(\begin{array}{c|ccc|cc|c|c}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline
1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ \hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}\right)$$

is a lower block triangular matrix.

Since M_V^X is lower block triangular, one can implement block wise Gaussian row elimination from the down to top i.e., doing block wise Gaussian row reduction of transpose of M_V^X to reduce the time complexity for computing the rank of M_V^X.

Consider V, X are chosen randomly such that $|V| = |X| = 2^{n-1}$. Here $|X^i|$ and $|V^i|$ are approximately $\frac{1}{2}\binom{n}{i}$ for $0 \le i \le n$. The time complexity for ith block wise row elimination of is $O(\binom{n}{n-i}^3) = O(\binom{n}{i}^3)$. Hence, the time complexity for finding the rank of M_V^X is $O(\sum_{i=0}^{n}\binom{n}{i}^3)$.

Now we will discuss about the rank of $M_{S(p)}^d$. In this case, $X = B_{n,d}$ and $V = S(p)$. So, $|X^i| = \binom{n}{i}$ for $0 \le i \le d$ and $|X^i| = 0$ for $d + 1 \le i \le n$. If $p \in P_n$ is a random polynomial, then we have $|V^i| \approx \frac{1}{2}\binom{n}{i}$, for $0 \le i \le n$. During the block wise row operation of matrix $M_{S(p)}^d$ from down to top, every time all columns (monomials) should be eliminated to have the rank equal to number of columns. So, same number of rows are eliminated and rest of the rows augmented to the next block of rows. Since $|X^{n-j}| = 0, 0 \le j < n - d$, there is no computation needed for the jth block wise row elimination. For jth block operation, $n - d \le j \le n$, the number of rows is $r_j = |V^{n-j}| + \displaystyle\sum_{i=0}^{j-1}|V^{n-i}| - \sum_{i=n-d}^{j-1}\binom{n}{n-i} \approx \frac{1}{2}\sum_{i=0}^{j}\binom{n}{i} - \sum_{i=n-d}^{j-1}\binom{n}{i}$. For $d < \frac{n}{2}$, $r_j = \displaystyle\frac{1}{2}(\binom{n}{j} + \sum_{i=n-d}^{j-1}\binom{n}{i} + \sum_{i=d+1}^{n-d-1}\binom{n}{i} + \sum_{i=n-j+1}^{d}\binom{n}{n} + \sum_{i=0}^{n-j}\binom{n}{i}) - \sum_{i=n-d}^{j-1}\binom{n}{i} = \frac{1}{2}(\binom{n}{j} + \sum_{i=d+1}^{n-d-1}\binom{n}{i} + \sum_{i=0}^{n-j}\binom{n}{i}) = O(2^n)$. At the jth block wise operation, the sub matrix has r_j many rows, $\displaystyle\sum_{i=0}^{n-j}\binom{n}{i}$ many columns and $\binom{n}{n-j}$ many columns to be eliminated. Therefore, the time complexity for the jth block wise row elimination is

$$O(r_j\binom{n}{n-j}(\sum_{i=0}^{n-j}\binom{n}{i})) = O(r_j\binom{n}{n-j}^2) = O(r_j\binom{n}{j}^2)$$

and hence, finding the rank of $M^d_{S(p)}$ is $O(\sum_{j=n-d}^{n}(r_j\binom{n}{j}^2)) = O(2^n\sum_{j=n-d}^{n}\binom{n}{j}^2)$. However, as discussed in Subsection 3.2, each sub-matrix is sparser by $O(2^d)$, which can be exploited for block wise elimination to speed up the process by $O(2^d)$. Thus, the time complexity is better than the time complexity of usual algorthims described in Section 3. Moreover, we have advantage in space complexity as we need only the sub-matrix of size $r_j \times \binom{n}{j} = O(2^n\binom{n}{j})$ at the jth block operation instead of the whole $2^{n-1} \times 2^{n-1}$ matrix.

The following section contains the main result, by changing the ordering to $<$, we gain better time and space complexity.

5.2 Ordering $<$

Let the monomials of $B_{n,n}$ and vectors of V_n be ordered by $<$. Consider a set of monomials $X \subseteq B_{n,n}$ and a set of vectors $V \subseteq V_n$. Let $X^0, X^1, \ldots, X^{2^k-1}, k \leq n$, be disjoint subsets of X, partitioned on the value of last k coordinates of the exponent vector α of monomials x^α. The superscript i of X^i denotes the integer value of last k-coordinates of exponent vector α. If $x^\alpha \in X^i$, $x^\beta \in X^j$ and $i < j$ then $x^\alpha < x^\beta$ and $\alpha < \beta$.

Example 3. *Consider $X = B_{4,2}$. Then $\mathsf{log}(B_{4,2}) = \{0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 12\}$ with the ordering $<$. Here the vectors are represented in their integer form. Fixing the last two coordinates of α, we have $\mathsf{log}(B^0_{4,2}) = \{0, 1, 2, 3\}$,$\mathsf{log}(B^1_{4,2}) = \{4, 5, 6\}$, $\mathsf{log}(B^2_{4,2}) = \{8, 9, 10\}$ and $\mathsf{log}(B^3_{4,2}) = \{12\}$.*

Similarly, the vector set V is partitioned by the value of last k coordinates of vectors of V and are denoted by $V^0, V^1, \ldots, V^{2^{k-1}}$. If $v \in V^i$, $x^\alpha \in X^j$ and $i < j$, it is clear form the ordering $<$ of vectors that $v < \alpha$ and $\alpha \nsubseteq v$. Let denote $\mathsf{vect}(i)$ is the vector form of binary representation of i. Hence, we have the following lemma.

Lemma 1. *The incidence matrix $M^{X^j}_{V^i}$ is a zero matrix if $\mathsf{vect}(j) \nsubseteq \mathsf{vect}(i)$ for $0 \leq i, j \leq 2^k - 1$.*

Example 4. *Let X be the set of monomials on 4-variables such that $\mathsf{log}(X) = \{1, 2, 3, 4, 8, 9, 10, 14\}$ and set of vectors $V = \{0, 3, 4, 5, 7, 9, 12, 15\}$. The vectors are shown in their integer form. If we fix last two coordinates, then $\mathsf{log}(X^0) = \{1, 2, 3\}, \mathsf{log}(X^1) = \{4\}, \mathsf{log}(X^2) = \{8, 9, 10\}, \mathsf{log}(X^3) = \{14\}$ and*

$V^0 = \{0,3\}, V^1 = \{4,5,7\}, V^2 = \{9\}, V^3 = \{12,15\}$. Then the matrix

$$M_V^X = \left(\begin{array}{ccc|c|ccc|c} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right)$$

Here, $M_{V^0}^{X^1}, M_{V^0}^{X^2}, M_{V^0}^{X^3}, M_{V^1}^{X^2}, M_{V^1}^{X^3}, M_{V^2}^{X^1}, M_{V^2}^{X^3}$ are zero sub-matrices of M_V^X.

Since $\mathsf{vect}(j) \not\subseteq \mathsf{vect}(i)$ for $j > i$, $M_{V^i}^{X^j}$ is zero matrix for $j > i$. So, we have the following theorem.

Theorem 3. *The incidence matrix M_V^X is a lower block triangular matrix with $M_{ij} = M_{V^i}^{X^j}$ on the ordering $<$ of elements of V and X.*

Since M_V^X is lower block triangular, one can implement block wise Gaussian row elimination from down to top for reducing the time complexity of computing the rank of M_V^X. Hence we have the following result on the rank of M_V^X.

Corollary 1. *The $rank(M_V^X) < |X|$ iff $rank(M_{\overline{V}}^{\overline{X}}) < |\overline{X}|$ where $\overline{V} = \cup_{i=0}^{p} V^{2^k-1-i}$ and $\overline{X} = \cup_{i=0}^{p} X^{2^k-1-i}$ for some $0 \leq p \leq 2^k - 1$.*

Therefore, we have the following necessary condition on the rank of M_V^X.

Corollary 2. *If $\sum_{i=0}^{p} |V^{2^k-1-i}| < \sum_{i=0}^{p} |X^{2^k-1-i}|$ for some $0 \leq p \leq 2^k - 1$, then $rank(M_V^X) < |X|$. Moreover, if $|V| = |X|$ and $\sum_{i=0}^{p} |V^i| > \sum_{i=0}^{p} |X^i|$ for some $0 \leq p \leq 2^k - 1$, then $rank(M_V^X) < |X|$.*

In Example 4, we have $|V^3 \cup V^2| = 3$ and $|X^3 \cup X^2| = 4$. Hence, $rank(M_V^X) < |X|$. There is an annihilator on the monomials from X of the polynomial having support set V. The inequality in Corollary 2 classifies some polynomials of having low AI.

For a random set of 2^{n-1} vectors, V, and a random set of 2^{n-1} monomials, X, $|V^i| \approx 2^{n-k-1}$ and $|X^i| \approx 2^{n-k-1}$. Now one can use the Corollary 1 and Corollary 2, to check the rank of M_V^X. One can use corollary 2 in better way by finding a proper permutation on the variables $x_1, x_2, \ldots, x_n$, such that $\sum_{i=0}^{p} |V^{2^k-1-i}| < \sum_{i=0}^{p} |X^{2^k-1-i}|$ for a some p. Then, one can compute the rank of M_V^X faster.

Corollary 3. *If $rank(M_V^X) = |X|$ then for every permutation on variables $x_1, x_2, \ldots, x_n$ and every k, p, $0 \leq k \leq n, 0 \leq p < 2^k$, $\sum_{i=0}^{p} |V^{2^k-1-i}| \geq \sum_{i=0}^{p} |X^{2^k-1-i}|$.*

Hence, using Corollary 1, one has to perform block wise row elimination operation from down to top of the matrix, to compute the rank of M_V^X. During the operation, the un-eliminated rows in a block are augmented with the next block.

Example 5. *Consider the sets X and V in Example 4. Then the block wise Gaussian row reduction can be done as following. The block of rows enclosed by double lines are to be reduced.*

$$M_V^X = \left(\begin{array}{ccc|c|ccc|c}
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
\hline\hline
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline\hline
\end{array}\right)$$

$$\to \left(\begin{array}{ccc|c|ccc|c}
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
\hline\hline
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
\hline\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
\end{array}\right) \to \left(\begin{array}{ccc|c|ccc|c}
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\hline\hline
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
\hline\hline
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
\end{array}\right)$$

$$\to \left(\begin{array}{ccc|c|ccc|c}
\hline\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\hline\hline
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
\end{array}\right) \to \left(\begin{array}{ccc|c|ccc|c}
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\hline
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
\end{array}\right)$$

After the row reduction, we got $rank(M_V^X) = 6$ i.e., there are two free monomials x_2 and x_2x_4. So, there are 2 linearly independent annihilators on the monomials from X of the polynomial having support set V.

Now consider V, X are chosen randomly such that $|V| = |X| = \eta$. Since k variables are fixed, there are 2^k blocks of rows of size approximately $\frac{\eta}{2^k}$. The time complexity for row elimination of each block is $O(\eta \times (\frac{\eta}{2^k})^2) = O(\eta^3 2^{-2k})$. Hence, the time complexity for finding the rank of M_V^X is $O(2^k \times \eta^3 2^{-2k}) = O(\eta^3 2^{-k})$. If $|V| = |X| = 2^{n-1}$, the time complexity for finding the rank of M_V^X is $O(2^{3n-k})$. If one fixes all n variables, theoretical time complexity becomes $O(2^{2n})$, i.e., quadratic time complexity. Moreover, the space complexity for the computation is $O(2^n)$ as one needs only the block of rows during the computation.

Theorem 4. *For a randomly chosen $V \subset V_n$ and $X \subset B_{n,n}$ such that $|V| = |X| = 2^{n-1}$, the expected time complexity and space complexity to compute the rank of the $2^{n-1} \times 2^{n-1}$ matrix M_V^X is $O(2^{2n})$ and $O(2^n)$ respectively i.e., quadratic time complexity and linear space complexity on the $|X|$ respectively.*

Now we will discuss about the rank of $M^d_{S(p)}$, which is important to compute $\mathsf{AI}(p)$ for $p \in P_n$. In this case, $X = B_{n,d}$ and $V = S(p)$. Since the monomial set X is not randomly chosen, the time complexity differs than the described one in Theorem 4. Fixing the last k coordinates, we have $|B^i_{n,d}| = b_i = \sum_{j=0}^{d-\mathsf{wt}(i)} \binom{n-k}{j}$ for $0 \le i < 2^k, 0 \le k \le n$. If $p \in P_n$ is a random polynomial, then we have $|V^i| \approx 2^{n-k-1}$, $0 \le i < 2^k$. Here onwards, we follow the notation $K = 2^k - 1$ and $N = 2^n - 1$. During the block wise row operation (from down to top) of matrix $M^d_{S(p)}$, every time all columns (monomials) in the block should be eliminated to have the rank equal to number of columns. So, the same number of rows also are eliminated and rest of the rows are augmented to the next block of rows. Hence, during the jth block wise row operation, for $0 \le j \le K$, the number of rows is $r_j = |V^{K-j}| + \sum_{i=0}^{j-1}(|V^{K-i}| - b_{K-i}) \approx (j+1)2^{n-k-1} - \sum_{i=0}^{j-1} b_{K-i}$. At the jth operation, the sub-matrix contains r_j rows and $c_j = \sum_{i=0}^{K-j} b_i$ columns, b_{K-j} columns from these c_j columns to be eliminated. So, the time complexity for the row elimination of jth block is $O(r_j c_j b_{K-j})$ and hence, time complexity to find the rank of $M^d_{S(p)}$ is $O(\sum_{j=0}^{K} r_j c_j b_{K-j})$.

For $k = n$, the time complexity to compute the rank of $M^d_{S(p)}$ is $O(\sum_{j=0}^{N} r_j c_j b_{N-j})$. Here,

$$b_i = \sum_{i=0}^{d-\mathsf{wt}(i)} \binom{0}{i} = \begin{cases} 1 \text{ if } \mathsf{wt}(i) \le d \\ 0 \text{ if } \mathsf{wt}(i) > d. \end{cases}$$

So, $b_{N-j} = \begin{cases} 1 \text{ if } \mathsf{wt}(j) \ge n-d \\ 0 \text{ if } \mathsf{wt}(j) < n-d, \end{cases}$

$c_j = \sum_{i=0}^{N-j} b_i = \sum_{\substack{0 \le i \le N-j \\ \mathsf{wt}(i) \le d}} 1 = \sum_{i=0}^{d} \binom{n}{i} - \sum_{\substack{0 \le i \le j-1 \\ \mathsf{wt}(i) \ge n-d}} 1$

and $r_j \approx \frac{j+1}{2} - \sum_{i=0}^{j-1} b_{N-i} = \frac{j+1}{2} - \sum_{\substack{0 \le i \le j-1 \\ \mathsf{wt}(i) \ge n-d}} 1.$

When $\mathsf{wt}(j) < n-d$ i.e., $b_{N-j} = 0$, there is no column to eliminate and hence no operation needed for the block operation. When $\mathsf{wt}(j) \ge n-d$, i.e., $b_{N-j} = 1$, there is only one column (monomial) to eliminate. So, the time complexity for j-th block operation is $O(r_j c_j)$. Therefore, the time complexity to find the rank of $M^d_{S(p)}$ is $O(\sum_{\substack{0\le j\le 2^n-1\\ \mathsf{wt}(j)\ge n-d}} r_jc_j)$. Now we will simplify it.

$$\sum_{\substack{0\le j\le N\\ \mathsf{wt}(j)\ge n-d}} r_jc_j = \sum_{\substack{0\le j\le N\\ \mathsf{wt}(j)\ge n-d}} (\frac{j+1}{2} - \sum_{\substack{0\le i\le j-1\\ \mathsf{wt}(i)\ge n-d}} 1)(\sum_{i=0}^{d}\binom{n}{i} - \sum_{\substack{0\le i\le j-1\\ \mathsf{wt}(i)\ge n-d}} 1)$$

$$\le \sum_{\substack{0\le j\le N\\ \mathsf{wt}(j)\ge n-d}} (\frac{j+1}{2} - \sum_{\substack{0\le i\le j-1\\ \mathsf{wt}(i)\ge n-d}} 1)(\sum_{i=0}^{d}\binom{n}{i})$$

There are $\sum_{i=n-d}^{n}\binom{n}{i}$ many terms in the summation $\sum_{\substack{0\le j\le N\\ \mathsf{wt}(j)\ge n-d}} j$. The integer i has $\mathsf{wt}(i)$ many non-zero positions in binary expansion and each non-zero position k contributes the value 2^k to the summation. In the summation, each position, k, for $0\le k<n$ contributes the value $\frac{1}{n}\sum_{i=n-d}^{n} i\binom{n}{i}$ $= \sum_{i=n-d}^{n}\binom{n-1}{i-1}$ many times.

So, $\sum_{\substack{0\le j\le N\\ \mathsf{wt}(j)\ge n-d}} \frac{j+1}{2} = \frac{1}{2}(\sum_{\substack{0\le j\le N\\ \mathsf{wt}(j)\ge n-d}} j + \sum_{i=n-d}^{n}\binom{n}{i})$

$$= \frac{1}{2}(\sum_{i=n-d}^{n}\binom{n-1}{i-1}(2^0+2^1+\cdots+2^{n-1}) + \sum_{i=n-d}^{n}\binom{n}{i})$$

$$= \frac{1}{2}(\sum_{i=n-d}^{n}\binom{n-1}{i-1}(2^n-1) + \sum_{i=n-d}^{n}\binom{n}{i}) = 2^{n-1}\sum_{i=n-d}^{n}\binom{n-1}{i-1} + \frac{1}{2}\sum_{i=n-d}^{n}\binom{n-1}{i})$$

Now, in the summation $\sum_{\substack{0\le j\le N\\ \mathsf{wt}(j)\ge n-d}}\sum_{\substack{0\le i\le j-1\\ \mathsf{wt}(i)\ge n-d}} 1$, an integer k with $\mathsf{wt}(k)\ge n-d$, is counted l of times, where $l = |\{s : k < s \le N, \mathsf{wt}(s)\ge n-d\}|$. Let, $i_1, i_2, \ldots, N$ are integers with weight at least $n-d$, then i_1 is counted $\sum_{i=n-d}^{n}\binom{n}{i} - 1$ times, i_2 is counted $\sum_{i=n-d}^{n}\binom{n}{i} - 2$ times and so on.

So, $\sum_{\substack{0\le j\le N\\ \mathsf{wt}(j)\ge n-d}}\sum_{\substack{0\le i\le j-1\\ \mathsf{wt}(i)\ge n-d}} 1 = (\sum_{i=n-d}^{n}\binom{n}{i} - 1) + (\sum_{i=n-d}^{n}\binom{n}{i} - 2) + \cdots + 0$

$$= \frac{1}{2}\sum_{i=n-d}^{n}\binom{n}{i}(\sum_{i=n-d}^{n}\binom{n}{i} - 1)$$

Hence, $\sum_{\substack{0\le j\le N \\ \mathsf{wt}(j)\ge n-d}} r_j c_j \le (2^n \sum_{i=n-d}^{n} \binom{n}{i} - (\sum_{i=n-d}^{n} \binom{n}{i})^2) \sum_{i=0}^{d} \binom{n}{i}$
$= (\sum_{i=0}^{d} \binom{n}{i})^2 \sum_{i=d+1}^{n} \binom{n}{i}$.

Theorem 5. *For a randomly chosen polynomial $p \in P_n$, the expected time complexity and space complexity to compute the rank of the matrix $M^d_{S(p)}$ is $O((\sum_{i=0}^{d} \binom{n}{i})^2 \sum_{i=d+1}^{n} \binom{n}{i})$ and $O(\max_{0\le j\le N} r_j c_j)$ respectively.*

Since simplifying the above expression is not very easy, the time complexity bound given in the Theorem 5 is not a tight upper bound. Hence the theoretical time complexity is deduced in Theorem 5 do not have significant advantage over the general algorithm. However, in practice, it is very fast and can be used to compute for $n = 20$. Moreover, exploiting the sparseness of the sub-matrices, the computation speed can be further improved.

5.3 Ordering < and Dalai-Maitra Algorithm

In the sub-section 3.3, we discussed Dalai-Maitra algorithm in [9] to exploit the sparseness of the matrix D' for finding the rank of $M^d_{S(p)}$. In this section, we shall further use the ordering $<$ for faster computation. Now we shall follow the notation in sub-section 3.3. Now order the set of monomials Y and set of vectors Z by $<$. For k, $0 \le k \le n$, make partition of Y and Z on their last k coordinates as $Y^0, \ldots, Y^{2^k-1}$ and $Z^0, \ldots, Z^{2^k-1}$ respectively. Now, denoting $D'[Y^i, Z^j]$ is the sub-matrix in D' corresponding to the vector set Y^i and monomial set Z^j, we have the following theorem.

Theorem 6. *The matrix D' is a lower block triangular matrix with $D'_{ij} = D'[Y^i, Z^j]$, $0 \le i, j \le 2^k - 1$ on the ordering $<$ of elements of Y and Z.*

Comparing to the partitions in subsection 5.2, we have $|Y^i| \approx \frac{|V^i|}{2}$, $|Z^i| \approx \frac{|X^i|}{2}$. Therefore, the computation in this technique is expected to be 8 times faster than the computation in subsection 5.2. However, the theoretical complexity is same as in Theorem 5.

6 Experiment

Using the method described in Section 5.3, it is possible to check AI of a polynomial from P_{20} with less memory. In this section we present some experimental results on some important power S-boxes (i.e., multi out put polynomials from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^n}$) as presented in [1]. The AI of an S-box is the minimum of the AI of the non-trivial linear combination of the component functions of the S-box. The AI of n-variable inverse function is bounded by $\lfloor\sqrt{n}\rfloor + \lceil\frac{n}{\lfloor\sqrt{n}\rfloor}\rceil - 2$, Kasami and Niho exponents are bounded by $\lfloor\sqrt{n}\rfloor + \lceil\frac{n}{\lfloor\sqrt{n}\rfloor}\rceil$ [19].

Experimentally, we check that the AI of inverse S-box is $\lfloor\sqrt{n}\rfloor+\lceil\frac{n}{\lfloor\sqrt{n}\rfloor}\rceil-2$ for $n \leq 21$. Moreover, we found that the number of annihilators of the component functions and its complement functions at $\lfloor\sqrt{n}\rfloor+\lceil\frac{n}{\lfloor\sqrt{n}\rfloor}\rceil-2$ are same. Therefore, we have the following conjecture.

Conjecture 1. *Let $INV : \mathbb{F}_{2^n} \mapsto \mathbb{F}_{2^n}$ be the inverse mapping i.e., $INV(x) = x^{-1} = x^{2^n-2}$ for $x \in \mathbb{F}_{2^n}$. Then $\mathsf{AI}(INV) = \lfloor\sqrt{n}\rfloor+\lceil\frac{n}{\lfloor\sqrt{n}\rfloor}\rceil-2$. The number of $(\lfloor\sqrt{n}\rfloor + \lceil\frac{n}{\lfloor\sqrt{n}\rfloor}\rceil - 2)$-degree annihilators of $\alpha_0 + \sum_{i=1}^{n} a_i INV^i$ are same, where INV^i is the ith component function of INV and $\alpha_0, a_i \in \{0,1\}$ and not all a_i are 0.*

A Kasami exponent $K : \mathbb{F}_{2^n} \mapsto \mathbb{F}_{2^n}$ is of the form $x^{2^{2k}-2^k+1}$ for $k \leq \frac{n}{2}$ and $\gcd(n,k) = 1$. The degree of Kasami exponent is $k+1$. Therefore, $\mathsf{AI}(K) \leq min\{k+1, \lfloor\sqrt{n}\rfloor + \lceil\frac{n}{\lfloor\sqrt{n}\rfloor}\rceil\}$. The following table presents the experimental result of $\mathsf{AI}(K)$ for the largest $k \leq \frac{n}{2}$ and $\gcd(n,k) = 1$.

n	k	$\deg(K)$	$\lfloor\sqrt{n}\rfloor + \lceil\frac{n}{\lfloor\sqrt{n}\rfloor}\rceil$	$\mathsf{AI}(K)$
10	3	4	7	4
11	5	6	7	5
12	5	6	7	5
13	6	7	8	6
14	5	6	8	6
15	7	8	8	7
16	7	8	8	7
17	8	9	9	8

For odd $n = 2s+1$, a Niho exponent $N : \mathbb{F}_{2^n} \mapsto \mathbb{F}_{2^n}$ is of the form $x^{2^s+2^{\frac{s}{2}}-1}$ if s is even and $x^{2^{\frac{3s+1}{2}}+2^s-1}$ if s is odd. The degree of Niho exponent is $d = \frac{n+3}{4}$ if $n \equiv 1 \mod 4$ and $d = \frac{n+1}{2}$ if $n \equiv 3 \mod 4$. Therefore, $\mathsf{AI}(N) \leq min\{d, \lfloor\sqrt{n}\rfloor + \lceil\frac{n}{\lfloor\sqrt{n}\rfloor}\rceil\}$. The following table presents the experimental result of $\mathsf{AI}(N)$.

n	$\deg(N)$	$\lfloor\sqrt{n}\rfloor + \lceil\frac{n}{\lfloor\sqrt{n}\rfloor}\rceil$	$\mathsf{AI}(N)$
9	3	7	3
11	6	7	5
13	4	8	4
15	8	8	7
17	5	9	5
19	10	9	9

References

[1] F. Armknecht, C. Carlet, P. Gaborit, S. Künzli, W. Meier, and O. Ruatta. Efficient computation of algebraic immunity for algebraic and fast algebraic attacks. In *Advances in Cryptology - Eurocrypt 2006*, number 4004 in Lecture Notes in Computer Science, pages 147–164. Springer-Verlag, 2006.

[2] C. Li C. Carlet, X. Zeng and L. Hu. Further properties of several classes of boolean functions with optimum algebraic immunity. *Design, Codes and Cryptography*, 52(3):303–338, 2006.

[3] C. Carlet and K. Feng. An infinite class of balanced functions with optimal algebraic immunity, good immunity to fast algebraic attacks and good nonlinearity. In *Advances in Cryptology - Asiacrypt 2008*, number 5350 in Lecture Notes in Computer Science, pages 425–440. Springer-Verlag, 2008.

[4] D. Coppersmith and S. Winograd. Matrix multiplication via arithmatic programming. *Journal of Symbolic Computation*, 9(3):251–280, 1990.

[5] 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.

[6] D. K. Dalai. *On Some Necessary Conditions of Boolean Functions to Resist Algebraic Attacks.* PhD thesis, Indian Statistical Instituite, 2006.

[7] D. K. Dalai, K. C. Gupta, and S. Maitra. Results on algebraic immunity for cryptographically significant boolean functions. In *INDOCRYPT-2004*, number 3348 in Lecture Notes in Computer Science, pages 92–106. Springer-Verlag, 2004.

[8] D. K. Dalai, K. C. Gupta, and S. Maitra. Cryptographically significant boolean functions: Construction and analysis in terms of algebraic immunity. In *Fast Software Encryption, FSE-2005*, number 3557 in Lecture Notes in Computer Science, pages 98–111. Springer-Verlag, 2005.

[9] D. K. Dalai and S. Maitra. Reducing the number of homogeneous linear equations in finding annihilators. In *SETA-2006*, number 4086 in Lecture Notes in Computer Science, pages 376–390. Springer-Verlag, 2006.

[10] D. K. Dalai, S. Maitra, and S. Sarkar. Basic theory in construction of boolean functions with maximum possible annihilator immunity. *Design, Codes and Cryptography*, 40(1):41–58, 2006.

[11] T. A. Davis. *Direct Methods for Sparse Linear Systems.* SIAM, 2006.

[12] F. Didier. Using wiedemann's algorithm to compute the immunity against algebraic and fast algebraic attacks. In *INDOCRYPT-2006*, number 4329 in Lecture Notes in Computer Science, pages 236–250. Springer-Verlag, 2006.

[13] F. Didier and J. Tillich. Computing the algebraic immunity efficiently. In *Fast Software Encryption, FSE-2006*, number 4047 in Lecture Notes in Computer Science, pages 359–374. Springer-Verlag, 2006.

[14] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices.* Oxford Science Publications, 1989.

[15] N. Li, L. Qu, W. Qi, G. Feng, C. Li, and D. Xie. On the construction of boolean functions with optimal algebraic immunity. *IEEE Transactions on Information Theory*, 54(3):1330–1334, 2008.

[16] F. J. MacWilliams and N. J. A. Sloane. *The theory of error-correcting codes.* Elsevier/North-Holland, Amsterdam, 1981.

[17] R. J. McEliece. The guruswami-sudan decoding algorithm for reed-solomon codes. [online] Available: www.systems.caltech.edu/EE/Faculty/rjm/papers/RSD-JPL.pdf, 2003.

[18] W. Meier, E. Pasalic, and C. Carlet. Algebraic attacks and decomposition of boolean functions. In *Advances in Cryptology - Eurocrypt 2004*, number 3207 in Lecture Notes in Computer Science, pages 474–491. Springer-Verlag, 2004.

[19] Y. Nawaz, G. Gong, and K. C. Gupta. Upper bounds on algebraic immunity of boolean power functions. In *Fast Software Encryption, FSE-2006*, number 4047 in Lecture Notes in Computer Science, pages 375–389. Springer-Verlag, 2006.

[20] V. Strassen. Guassian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.

[21] D. H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, 32(1):54–62, 1986.