Algebraic Attacks and Decomposition of Boolean Functions

Willi Meier[1], Enes Pasalic[2], and Claude Carlet[2]

[1] FH Aargau, CH-5210 Windisch, Switzerland
`meierw@fh-aargau.ch`
[2] INRIA, projet CODES, Domaine de Voluceau, Rocquencourt
BP 105, 78153 Le Chesnay Cedex, France
`Enes.Pasalic,Claude.Carlet@inria.fr.`

Abstract. Algebraic attacks on LFSR-based stream ciphers recover the secret key by solving an overdefined system of multivariate algebraic equations. They exploit multivariate relations involving key bits and output bits and become very efficient if such relations of low degrees may be found. Low degree relations have been shown to exist for several well known constructions of stream ciphers immune to all previously known attacks. Such relations may be derived by multiplying the output function of a stream cipher by a well chosen low degree function such that the product function is again of low degree. In view of algebraic attacks, low degree multiples of Boolean functions are a basic concern in the design of stream ciphers as well as of block ciphers.
This paper investigates the existence of low degree multiples of Boolean functions in several directions: The known scenarios under which low degree multiples exist are reduced and simplified to two scenarios, that are treated differently in algebraic attacks. A new algorithm is proposed that allows to successfully decide whether a Boolean function has low degree multiples. This represents a significant step towards provable security against algebraic attacks. Furthermore, it is shown that a recently introduced class of degree optimized Maiorana-McFarland functions immanently has low degree multiples. Finally, the probability that a random Boolean function has a low degree multiple is estimated.

Keywords : Algebraic attacks, Stream ciphers, Boolean functions, Algebraic degree, Annihilator, Low degree multiple, Resiliency.

1 Introduction

Algebraic attacks on stream ciphers based on linear feedback shift registers (LFSR's) have been proposed in [8]. Many stream ciphers consist of a linear part, producing a sequence with a large period, usually composed of one or several LFSR's, and a nonlinear combining function f that produces the output, given the state of the linear part. Algebraic attacks recover the secret key by solving an overdefined system of multivariate algebraic equations. These attacks exploit multivariate relations involving key/state bits and output bits of f. If one such relation is found that is of low degree in the key/state bits, algebraic attacks are very efficient, [6].

In [8] it is demonstrated that low degree relations and thus successful algebraic attacks exist for several well known constructions of stream ciphers that are immune to all previously known attacks. In particular, low degree relations are proven to exist for ciphers using a combining function f with a small number of inputs. These low degree relations are obtained by producing low degree polynomial multiples of f, i.e., by multiplying the Boolean function f by a well chosen low degree function g such that the product function $f * g$ is again of low degree.

There have become known alternative methods to attack stream ciphers by solving overdefined systems of equations using Gröbner bases, [11]. In order to be efficient, these methods rest on the existence of low degree multiples as well.

To counter algebraic attacks, it is recommended in [8], that the combining function f should have at least 32 inputs. But even then, by now it cannot be excluded for certain, that f has low degree multiples that would then make a fielded or a new design vulnerable to algebraic attacks. This is in strong contrast to other attacks on stream ciphers: A variety of proposed stream ciphers have been shown to be provably resistant, e.g., against the Berlekamp-Massey shift register synthesis algorithm.

In a different direction, in view of algebraic attacks on block ciphers, [7], it may be desirable to know for certain, e.g., that there are no low degree equations, relating output bits of a (reduced round) block cipher, plaintext bits and key bits. We mention also that recently the framework of algebraic attacks has been extended to combiners with memory [6, 1].

As a consequence, investigation of Boolean functions with regard to existence of low degree multiples is of both, theoretical and practical interest.

The results of this paper contribute to this problem in four directions: We reduce and simplify the scenarios found in [8], under which low degree multiples may exist. As a significant step towards provable resistance against algebraic attacks we propose an algorithm that allows to successfully decide whether a Boolean function has low degree multiples. This new algorithm can be efficient for input sizes of f of 32 bits or larger. Furthermore, we show that for a recently proposed class of Boolean functions, the degree optimized Maiorana-McFarland class [18], relatively low degree multiples are immanent. Finally we derive upper bounds on the probability that a random Boolean function has a low degree multiple. This is partly done by using results from coding theory. These bounds are shown to give strong estimates for input sizes of practical interest.

To further explain some of our results, recall that the main cryptographic criteria for Boolean functions f used for stream cipher applications had previously been a high algebraic degree, to counter linear synthesis by Berlekamp-Massey algorithm, some order of correlation immunity (resiliency), and large distance to affine functions (high nonlinearity), to withstand different types of correlation and linear attacks [17, 13, 3]. There are some known tradeoffs between the criteria, e.g., there is the bound by Siegenthaler [19], that the algebraic degree of f is upper bounded by $n-t-1$, where n is the number of inputs of f and $t<n-1$ is its order of resiliency.

The more recent algebraic attacks impose a new restriction on the combining function f chosen: f shouldn't have low degree multiples. In [8], essentially three different scenarios are described which lead to low degree multiples of a Boolean function which can be exploited in algebraic attacks. We show that these scenarios can be reduced to two, to be treated differently in algebraic attacks. This simplified description of scenarios leads to a precise measure of algebraic immunity of a Boolean function f: *The algebraic immunity $AI(f)$ is the minimum value of d such that f or $f+1$ admits an annihilating function of degree d.* Recall that an annihilator of f is a non-zero function g such that $f*g=0$.

The new criterion that f shouldn't have a low algebraic immunity, may be in conflict with some established criteria. This is exemplified for the Maiorana-McFarland class. These functions can have high resiliency, high nonlinearity, and optimum algebraic degree [10, 2, 4, 18]. Nevertheless it is shown in this paper that such functions can have relatively low algebraic immunity (Example 1). This is done by deriving a useful representation for the complete set of annihilators for a given function f. Any annihilator can be viewed as a concatenation of annihilators from some smaller variable space. This method when applied to a function in the standard Maiorana-McFarland class [10, 2] only yields annihilators of degree larger than the degree of the function itself. However, this method may be successfully applied to the degree optimized Maiorana-McFarland class [18], showing that relatively low degree annihilators are immanent for this class.

In the design of stream ciphers, this property needs to be avoided. Therefore, it is desirable to have an efficient algorithm for deciding whether a given Boolean function has no low degree annihilator. Such an algorithm is derived in this paper (Algorithm 2). A refined version allows to decide whether a Boolean function with n inputs has no annihilator of degree d at most 5, in about $\binom{n}{d-1}^3$ operations, which e.g. for $n=32$ is certainly feasible. If for a stream cipher a degree d annihilator with $d=4$ (say) of its combining function f (or $f+1$) is found by our algorithm, we can break this cipher. On the other hand, if f and $f+1$ are shown to have no annihilator of degree $d \leq 5$, this cipher has some amount of immunity against algebraic attacks, as for $d=6$ and for a size of the initial state of 128 bits, the computational complexity of the basic algebraic attack in [8] is already about 2^{96}.

The paper is organized as follows. In Section 2 the basic definitions and notions regarding Boolean functions are introduced. Section 3 recalls and simplifies the various scenarios of algebraic attacks. Algebraic properties of annihilators for an arbitrary function f are addressed in Section 4, where an alternative representation of annihilators is given which is useful for the analysis of some well known classes of Boolean functions. Section 5 deals with the fundamental problem of efficiently deciding whether the combining function in a stream cipher has annihilators of low degrees. In Section 6 we estimate an upper bound on the probability that a random function has annihilators of certain degree.

2 Preliminaries

A Boolean function on n variables may be viewed as a mapping from $\{0,1\}^n$ into $\{0,1\}$. A Boolean function $f(x_1,\ldots,x_n)$ is also interpreted as the output column of its *truth table* f, i.e., a binary string of length 2^n,

$$\overline{f} = [f(0,0,\cdots,0), f(1,0,\cdots,0), f(0,1,\cdots,0),\ldots, f(1,1,\cdots,1)].$$

The *Hamming distance* between n-variable functions f,g, denoted by $d(f,g)$, is

$$d(f,g) = \#\{x \in \mathbb{F}_2^n \mid f(x) \neq g(x)\}.$$

Also the *Hamming weight* or simply the weight of f is the number of ones in $\overline{f}$. This is denoted by $wt(f)$. An n-variable function f is said to be *balanced* if its output column in the truth table contains equal number of 0's and 1's (i.e., $wt(f) = 2^{n-1}$).

The Galois field of order 2^n will be denoted by $\mathbb{F}_{2^n}$ and the corresponding vector space by $\mathbb{F}_2^n$. Addition operator over $\mathbb{F}_2$ is denoted by $\oplus$, and if no confusion is to arise we use the usual addition operator $+$. An n-variable Boolean function $f(x_1,\ldots,x_n)$ can be considered to be a multivariate polynomial over $\mathbb{F}_2$. This polynomial can be expressed as a sum of products representation of all distinct r-th order products $(0 \leq r \leq n)$ of the variables. More precisely, $f(x_1,\ldots,x_n)$ can be written as

$$f(x_1,\ldots,x_n) = \sum_{u\in\mathbb{F}_2^n} \lambda_u \left(\prod_{i=1}^{n} x_i^{u_i}\right) \ , \quad \lambda_u \in \mathbb{F}_2 \ , u = (u_1,\ldots,u_n). \tag{1}$$

This representation of f is called the *algebraic normal form* (ANF) of f. The *algebraic degree* of f, denoted by $deg(f)$ or sometimes simply d, is the maximal value of the Hamming weight of u such that $\lambda_u \neq 0$. There is a one-to-one correspondence between the truth table and the ANF via so called inversion formulae. The set of x values for which $f(x) = 1$ respectively $f(x) = 0$ is called the on-set respectively the off-set, denoted by $S_1(f)$ and $S_0(f)$. The ANF of f is fully specified by its on-set using the following expansion,

$$f(x_1,\ldots,x_n) = \sum_{\tau\in S_1(f)} \left(\prod_{i=1}^{n}(x_i+\tau_i+1)\right), \quad \tau = (\tau_1,\ldots,\tau_n). \tag{2}$$

The set of all Boolean functions in n variables is denoted by $\mathcal{R}_n$. For any $0 \leq b \leq n$ an n-variable function is called non degenerate on b variables if its ANF contains exactly b distinct input variables. Functions of degree at most one are called *affine* functions. An affine function with constant term equal to zero is called a *linear* function. The set of all n-variable affine (respectively linear) functions is denoted by $\mathcal{A}_n$ (respectively $\mathcal{L}_n$). The *concatenation*, denoted by $''||''$ simply means that the truth tables of the functions are merged. For instance, for $f_1, f_2 \in \mathcal{R}_{n-1}$ one may construct $\overline{f} = \overline{f_1}||\overline{f_2}$ (where $f \in \mathcal{R}_n$), meaning that the upper half part of the truth table of f correspond to f_1 and the lower part to f_2. The ANF of f is then given by $f(x_1,\ldots,x_n) = (1+x_n)f_1(x_1,\ldots,x_{n-1}) + x_n f_2(x_1,\ldots,x_{n-1})$.

3 Algebraic attacks: Scenarios revisited

In [8], three different scenarios (S3a, S3b, S3c) are described under which low degree relations (that hold with probability 1) may exist and how they can be exploited in algebraic attacks. The aim of this section is to show that these can be reduced to essentially two scenarios, and to clarify how to use them in an attack.
To recall the scenarios in [8], let the Boolean function f have high degree.

- S3a Assume that there exists a function g of *low* degree such that the product function is of low degree, *i.e.*, $f * g = h$, where h is a nonzero function of low degree.
- S3b Assume there exists a function g of low degree such that $f * g = 0$.
- S3c Assume there exists a function g of *high* degree such that $f * g = h$ where h is nonzero and of low degree.

Consider scenario S3c. Then $f*g = h \neq 0$. Multiply this equation by f. As $f^2 = f$ does hold over $\mathbb{F}_2$, we get $f^2*g = f*h = f*g = h$. Hence $f*h = h$. As h is of low degree, we are in scenario S3a. Therefore, scenario S3c is redundant. Further, one might consider another scenario (not contained in [8]): Factorizations of the form $f = g * h$, where g and/or h are of low degree. However, $g * (1 + g) = 0$ over $\mathbb{F}_2$. Hence by multiplying $f = g * h$ by $1 + g$, we get $f * (1 + g) = 0$, i.e., we are back in scenario S3b. These considerations suggest that in algebraic attacks one can always restrict to scenarios S3a and S3b. There is an interesting relation between the two:

Proposition 1 *Assume that $f * g = h \neq 0$, does hold for some functions g and h of degrees at most d (scenario S3a). Suppose in addition that $g \neq h$. Then there is a function $g^{'}$ of degree at most d such that $f * g' = 0$ (scenario S3b).*

Proof. As above, we have $f^2 * g = f * g = f * h = h$. Hence $f * (g + h) = 0$. ∎

The argument just given shows that we can reduce ourselves to scenario S3a in case where $g = h$, and scenario S3b. However, S3a with $g = h$ is equivalent to scenario S3b for the function $f + 1$.

The existence of algebraic attacks will impose that neither f nor $f + 1$ does admit an annihilating function of low degree. This motivates the notion “algebraic immunity” of f, denoted by $AI(f)$, which is the minimum value of d such that f or $f + 1$ admits an annihilating function of degree d.

In [8], low degree relations according to scenarios S3a or S3b are proven to exist for any Boolean function f with a small number of inputs:

Theorem 6.0.1 [8, 9] *Let f be any Boolean function with n inputs. Then there is a Boolean function $g \neq 0$ of degree at most $\lceil n/2 \rceil$ such that $f * g$ is of degree at most $\lceil n/2 \rceil$.*

Remark 1 *Without restricting the form of the function, the upper bound given above cannot be improved for the case of annihilators, i.e. $f * g = 0$. For instance one example of a function not admitting annihilators of degree lower than $n/2$ is given in [11]. Namely the function in 6 variables, denoted there CanFil 8, has annihilators of degree $d \geq 3$ only. Moreover, [9, Table 3] gives experimental evidence that a random function with 10 variables is not likely to have an annhilator of degree lower than 5.*

To exploit low degree relations as in scenarios S3a and S3b, assume that N_d linearly independent functions h with $f * g = h$ have been found, where h and g have low degree d. Similarly, assume that $N_d^{'}$ linearly independent functions g of low degree d have been found such that $f * g = 0$.

In an algebraic attack on an LFSR-based stream cipher, it is assumed that the feedback connections are known. Let $(s_0, ..., s_{k-1})$ be the initial state of the driving LFSR's. Then the output of the cipher is given by:

$$\begin{cases} b_0 & = & f(s_0, ..., s_{k-1}) \\ b_1 & = & f(L(s_0, ..., s_{k-1})) \\ b_2 & = & f(L^2(s_0, ...s_{k-1})) \\ ... & = & ... \end{cases}$$

Here L denotes the linear update function to the next state of the LFSR's involved. The problem is to recover the k-bit key $(s_0, ..., s_{k-1})$. Let x equal $L^i(s_0, ..., s_{k-1})$.

If the output bit $b_i = 1$, we use scenario S3b, i.e., $f * g = 0$, and get an equation $g(x) = 0$. Alternatively, we can use scenario S3a, $f * g = h$, and take $g(x) = h(x)$. However, either $g = h$, which gives nothing, or $g \neq h$, which gives $g + h = 0$, i.e. we are back in scenario S3b.

If $b_i = 0$, use scenario S3a: $h(x) = 0$. Hence for any known output bit b_i we get N_d equations, if $b_i = 0$, and $N_d^{'}$ equations, if $b_i = 1$.

If we get at least one such equation for each of sufficiently many output bits, we obtain a very overdefined system of multivariate equations of low degree d, that can be solved efficiently: There are about $T \approx \binom{n}{d}$ monomials of degree at most d in the k variables s_i, $i = 0, \ldots, k-1$ (assuming $d \ll n/2$). Consider each of these monomials as a new variable V_j. Given $R \geq \binom{n}{d}$ equations, we get a system of $R \geq T$ linear equations in the V_j's that can be solved by Gaussian elimination. If more than one equation holds per output bit, the output stream needed reduces accordingly.

4 Properties of the annihilator set

As set out in the introductory part, in the realm of algebraic attacks there is one major concern: Given a Boolean function f used in a stream cipher, the task is to determine whether this function has low algebraic immunity, i.e., whether f or $f + 1$ has a low degree annihilator. In this section we specify the structure of the set of annihilators for a given f, and also give an alternative representation of their ANF. Let $An(f) = \{g \mid f * g = 0\}$ denotes the annihilator set for the function f in the Boolean ring $\mathcal{R}_n = \mathbb{F}_2[x_1, \dots, x_n]/I$, I being an ideal generated by the polynomials $x_i^2 - x_i$, $i = 1, \dots, n$. Since in this ring $f(1 + f) = 0$ for any $f \in \mathcal{R}_n$ the set $An(f)$ is nonempty.

Theorem 1. *Let f be any Boolean function in $\mathcal{R}_n$. Then $An(f)$ is a principal ideal in $\mathcal{R}_n$ generated by $(1 + f)$, i.e. $An(f) = \{(1 + f)r \mid r \in \mathcal{R}_n\} =< 1 + f >$. Its cardinality equals to $|An(f)| = 2^{2^n - |S_1(f)|}$. In particular when f is balanced $|An(f)| = 2^{2^{n-1}}$.*

Proof. In order to show that $An(f)$ is a principal ideal in the Boolean ring $\mathcal{R}_n$ generated by $(1 + f)$, we prove firstly that $An(f)$ is a subring of $\mathcal{R}_n$, then an ideal which is principal.

To prove that $An(f)$ is a subring of $\mathcal{R}_n$ it is enough to demonstrate that $An(f)$ is closed under the operations $'+'$ and $'*'$. Clearly $An(f)$ is nonempty since $(1 + f) \in An(f)$. Let $g, h \in An(f)$. Then $f * (g + h) = f * g + f * h = 0$, and $f * (g * h) = (f * g) * h = 0$. Hence $An(f)$ is closed under $'+'$ and $'*'$ and therefore a subring of $\mathcal{R}_n$.

Obviously for any $r \in \mathcal{R}_n$, $g \in An(f)$, we have $r * g \in An(f)$. Thus $An(f)$ is an ideal. Let us prove that $An(f)$ is a principal ideal. For if $h \in An(f)$ and $h \notin < 1 + f >$, then $f * h = 0$ implying $h * (1 + f) = h$, so $h \in < 1 + f >$.

Next we prove the assertion on the cardinality of $An(f)$. Note that the condition $f(x) * g(x) = 0$ implies that

$$f(x) = 1 \Rightarrow g(x) = 0 \;\; \forall x \in \mathbb{F}_2^n.$$

Then at any position $\tau \in \mathbb{F}_2^n$ for which $f(\tau) = 0$, $g(\tau)$ may be selected arbitrary, i.e. there are $2^{2^n - |S_1(f)|}$ possibilities for g. Hence $|An(f)| = 2^{2^n - |S_1(f)|}$. In particular if f is balanced then $|An(f)| = 2^{2^{n-1}}$. ∎

Henceforth we restrict our discussion to balanced functions having much wider cryptographic applications (at least in the case of stream ciphers). For a balanced function f the quotient ring $\mathcal{R}_n/An(f)$ has $2^{2^{n-1}}$ elements. As noticed, there is a strong symmetry between the two different attacks based on the annihilators $f * g = 0$ and the multiples of low degree $f * r = h$. Indeed, the cardinality of nonzero annihilators $\#\{An(f) \setminus 0\} = 2^{2^{n-1}} - 1$ is the same as the number of distinct h when considering $f * r = h$. This is confirmed by noting that any function r in the coset $a + An(f)$ gives $f * r = f * a = h$, and there are $2^{2^{n-1}} - 1$ such cosets for $a \neq 0$. In other words, finding low degree annihilators is equivalent to designing a low degree function g defined on some subset of $S_0(f)$. Similarly, as any g defined on the subset of $S_0(f)$ gives $f * g = 0$, the existence of low degree multiples of the form $f * r = h$ may always be viewed as design of the low degree h on the subset of $S_1(f)$ due to the deccomposition of the form $r = g + h$. We attempt to deduce some properties of the cosets of $An(f)$ regarding its minimum degree.

Proposition 2 *Let $f \in \mathcal{R}_n$ be a nonaffine balanced function. Then $An(f)$ contains exactly one balanced function, namely the function $1+f$.*
In particular, there are no nonzero affine functions in $An(f)$.

Proof. In order that $f * g = 0$ the function g must satisfy $g(x) = 0$ whenever $f(x) = 1$. Since f is balanced $S_0(g) \geq 2^{n-1}$. Then if g is to be balanced it must be that $g = 1+f$. In particular, since any affine function is balanced and $1+f$ is nonlinear by assumption, there are no nonzero affine functions in $An(f)$. ∎

Corollary 1 *There is exactly one nonzero annihilator of degree one for any affine function $a \in \mathcal{A}_n$ given by $(1+a)$.*

Proof. Since a is affine the only balanced annihilator is of the form $1+a$ which is an affine function. ∎

4.1 Concatenating annihilators with application to Maiorana-McFarland class

We know that $M = \{1, x_1, \ldots, x_n, x_1x_2, \ldots, x_{n-1}x_n, \ldots, x_1x_2\cdots x_n\}$, the set of 2^n monomials, constitutes the basis of $\mathcal{R}_n$ which we call the *monomial basis.* An alternative basis may be derived by considering all the products of the form $\prod_{i=1}^{n}(x_i+\tau_i+1)$ when τ runs through $\mathbb{F}_2^n$. It is clear that any such product $\prod_{i=1}^{n}(x_i+\tau_i+1)$ specifies the function defined to be nonzero exactly for $x=\tau$ and zero otherwise. Hence the set $E_n = \{\prod_{i=1}^{n}(x_i+\tau_i+1) \mid \tau \in \mathbb{F}_2^n\}$ constitutes the basis of $\mathcal{R}_n$ which will be called *polynomial basis.* In fact distinct basis elements are orthogonal to each other, that is $e * e' = 0$ for $e \neq e' \in E_n$ with exception that for any $e \in E_n$ we have $e * e = e$ which is in accordance to the property that any element in the Boolean ring $\mathcal{R}_n$ is idempotent.

An important application of these ideas is a general result on the set of annihilators.

Theorem 2. *Let f be a balanced Boolean function in $\mathcal{R}_n$. In general, for a positive integer m, $1 \leq m \leq n-1$, write f as*

$$f(y,x) = \bigoplus_{\tau \in \mathbb{F}_2^{n-m}} \Big(\prod_{i=1}^{n-m}(y_i+\tau_i+1)\Big) r_\tau(x),$$

for $(y,x) \in \mathbb{F}_2^{n-m} \times \mathbb{F}_2^m$, and not necessarily distinct functions r_τ in $\mathcal{R}_m$. Then any annihilator of f can be written in the form,

$$g(y,x) = \bigoplus_{\tau \in \mathbb{F}_2^{n-m}} \Big(\prod_{i=1}^{n-m}(y_i+\tau_i+1)\Big) g_\tau(x), \tag{3}$$

where g_τ is any annihilator of r_τ.

Proof. Due to the orthogonality of distinct products $\prod_{i=1}^{n-m}(y_i + \tau_i + 1)$ and the fact that g_τ is annihilator of r_τ for any $\tau \in \mathbb{F}_2^{n-m}$, it is easily verified that $fg = 0$. By Theorem 1 for a function $r_\tau \in \mathcal{R}_m$ there are $2^{2^m - S_1(r_\tau)}$ distinct annihilators. Let $G = \{g \mid g_\tau \in A(r_\tau), \tau \in \mathbb{F}_2^{n-m}\}$ and denote by $r_0, \ldots, r_{2^{n-m}-1}$ the subfunctions of f when τ runs through $\mathbb{F}_2^{n-m}$. Then,

$$|G| = 2^{2^m - |S_1(r_0)|} \ldots 2^{2^m - |S_1(r_{2^{n-m}-1})|} = 2^{2^{n-m} \cdot 2^m - \sum_{i=0}^{2^{n-m}-1} |S_1(r_i)|} = 2^{2^{n-1}},$$

which is in accordance with Theorem 1, that is $|G| = |An(f)|$. It is obvious that the functions in G are two-by-two distinct, hence all annihilators are in G. ∎

This approach is a very efficient method for annihilating the functions which have a subfunction of low degree on some $(n-m)$-dimensional flat.

Example 1 *The functions in the standard Maiorana-McFarland class may be viewed as a concatenation of affine functions from some smaller variable space. That is* $f(y, x) = \bigoplus_{\tau \in \mathbb{F}_2^{n-m}} \left(\prod_{i=1}^{n-m}(y_i + \tau_i + 1) \right) a_\tau(x)$, *where* $a_\tau(x) \in \mathcal{A}_m$ *are affine functions in* m *variables for all* τ. *Then the annihilators of degree* $n-m+1$ *are for instance obtained by choosing* $g_{\tau^c}(x) = 1 + a_{\tau^c}(x)$ *in (3) for a fixed* $\tau^c \in \mathbb{F}_2^{n-m}$ *and otherwise* $g_\tau(x) = 0$. *But the degree of such an annihilator is* $n-m+1$ *which equals to the maximum degree of the Maiorana-McFarland class of functions and therefore not of practical use.* ∎

The result above is more successfully applied to the degree optimized Maiorana-McFarland class that has been introduced in [18]. Here some affine functions in $\mathcal{A}_m$ (at least one) are replaced by suitably chosen nonlinear function(s) h_i of degree $m - t - 1$, t being the order of resiliency. Then the degree of f is optimized, i.e. $deg(f) = n - t - 1$. Still, multiplying this function by $g(y, x) = \left(\prod_{i=1}^{n-m}(y_i + \tau_i + 1) \right)(1 + a_\tau(x))$ (for $\tau \in \mathbb{F}_2^{n-m}$ chosen such that f is affine on that m-dimensional flat) the degree of f is decreased from $n-t-1$ to $n-m+1$. As $m > n/2$ when $t > 0$ for this class, in many cases one obtains annihilators of degree $< n/2$.

5 How to decide the (non-) existence of annihilators

In this section we derive an efficient algorithm to decide whether a given boolean function f in n variables $x = (x_1, ..., x_n)$ has low algebraic immunity, i.e., whether f or $f+1$ has an an annihilator of low degree. From ([9], proof of Theorem C.0.1) one deduces the following algorithm for determining annihilating functions for f, i.e., functions g such that $f(x) * g(x) = 0$ for all x:

A necessary and sufficient condition for $f * g = 0$ is that the function g vanishes for all arguments x for which $f(x) = 1$. The algebraic normal form ANF of a function g in n variables of degree d is a sum of a constant and monomials $a_{i_1,i_2,\ldots,i_m} x^{i_1} x^{i_2} \cdots x^{i_m}$, $1 \le m \le d$, determined by its coefficients $a_{i_1,i_2,\ldots,i_m}$, whose number equals to $\sum_{i=0}^{d} \binom{n}{i}$. In some complexity estimates, we approximate this number by the summand $\binom{n}{d}$, which is dominant for $d < n/2$. In order to determine the unknown coefficients of an annihilating function g, substitute all arguments x in $g(x)$ with $f(x) = 1$. For balanced f these are 2^{n-1} arguments. We thus get 2^{n-1} linear equations for the coefficients of g, which can be solved by Gaussian elimination. This method immediately allows to decide whether there is an annihilator g of degree at most d, and if so, to determine a set of linearly independent annihilators (of degree at most d). In view of Theorem C.0.1 in [9] we assume $d \leq \lceil n/2 \rceil$.

Algorithm 1

1. Substitute all N arguments x with $f(x) = 1$ in the ANF of a general boolean function $g(x)$ of degree d. This gives a system of N linear equations for the coefficients of $g(x)$.
2. Solve this linear system.
3. If there is no (nontrivial) solution, output `no annihilator of degree` d, else determine sets of coefficients for linearly independent annihilators.

For n not much larger than about 10, solving this system of linear equations is quite easy. However, in [12] it is recommended that the combining function f in a stream cipher should have more than 10 (e.g. 32) arguments, to prevent algebraic attacks.

For such numbers of inputs, Algorithm 1 becomes infeasible, as the number of equations is on the order of 2^{n-1}, and the complexity of Gaussian elimination already for $n = 20$ inputs is about 2^{57}. In [11] there are given two alternative algorithms for determining low degree annihilators and low degree multiples of functions, both of which are based on Gröbner bases. The examples of functions given in [11] have at most $n = 10$ variables. No complexity estimates are given in [11] for determining the necessary Gröbner bases for general n, however it seems that these methods become infeasible as well for larger numbers of variables.

Here we propose an accelerated method for deciding whether a Boolean function has an annihilator of low degree d. As in Algorithm 1, let the (candidate) annihilators g of degree d of f be described as ANF with unknown coefficients.

We assume that f behaves roughly like a random function, i.e., the coefficients in the ANF of f are roughly chosen at random. If this is not the case, e.g., if the nonzero coefficients are sparse, the algorithm may be adapted to be even more efficient. (However, for cipher design, we do not advocate sparse functions.) Suppose f is (close to) balanced. Then the number of arguments x with weight $w \leq d$ and $f(x) = 1$ is about half the number of coefficients of $g(x)$.

The idea is to exploit some specific structure of the system of equations occurring in Algorithm 1. To see this, start with arguments x with Hamming weight $w = 1$. Suppose the only value 1 in x is at position i. Then substituting this x in $g(x) = 0$ gives $a_i + a_0 = 0$. Thus $a_i = a_0$. There are about $n/2$ arguments x of weight 1 with $f(x) = 1$. Assume $d \geq 2$. Consider all arguments x of weight 2 with $f(x) = 1$, and with value 1 in positions i and j. Then one gets $a_{ij} + a_i + a_j + a_0 = 0$. Hence a_{ij} for these indices can be expressed by coefficients of monomials of degrees 0 and 1. In general, for any argument x of weight w, $1 \leq w \leq d$, the resulting linear equation in the coefficients of $g(x)$ has a similar structure: There is exactly one coefficient of a monomial of degree w, (we term this a *coefficient of weight* w) which can immediately be expressed by coefficients of lower weight. By iterating this process for increasing weight w, until $w = d$, we can eliminate roughly half of the coefficients in $g(x)$ almost for free. We describe a basic version of an algorithm which for low degree d will later be considerably improved.

Algorithm 2

1. Let weight $w = 1$.
2. For all x of weight w with $f(x) = 1$ substitute 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. Choose random arguments x of arbitrary weight such that $f(x) = 1$ and substitute in $g(x) = 0$, until there are the same number of equations as unknowns.
5. Solve the linear system. If there is no solution, output `no annihilator of degree` d.

Algorithm 2 is aimed at showing that f has *no* annihilator of given degree d. However, if the system turns out to be solvable, one may try another set of arguments x in step 5. If the new system is again solvable, one checks whether the solutions found are consistent. In case the number of variables n of f is not too large, one may directly verify whether one has found an annihilator, by formally expanding $f(x) * g(x)$ and by checking whether the result is identically 0.

We estimate the computational and data complexity of Algorithm 2. The expressions of those coefficients that in step 2 have been replaced by linear combinations of coefficients of lower weight, need to be memorized for step 4. As the number of coefficients involved in these expressions is of order $\frac{1}{2}\binom{n}{d-1}$, and we have a number of $\frac{1}{2}\binom{n}{d}$ memorized coefficients in step 2, the number of memory bits is of order $M = \frac{1}{4}\binom{n}{d} \cdot \binom{n}{d-1}$. In the evaluation of $g(x)$ in step 4, one has to substitute the linear expressions found in step 2. The complexity of substituting x depends on its weight, and is at most of order M elementary operations. This needs to be done for about $\frac{1}{2}\binom{n}{d}$ values of x, as we have about this number of remaining unknowns. Hence we get a computational complexity in step 4 of order $\frac{1}{2}\binom{n}{d} * M = \frac{1}{8}\binom{n}{d}^2 \cdot \binom{n}{d-1}$. The computational complexity of step 5, and hence of Algorithm 2, is of order $\frac{1}{8}\binom{n}{d}^3$, if the exponent for Gaussian elimination $\omega = 3$. Thus Algorithm 2 does run roughly 8 times faster than Algorithm 1, when modified for low degree d (i.e., by taking a number of linear equations equal to the number of unknown coefficients in $g(x)$). To summarize, Algorithm 2 has the complexities as shown:

Memory	$\frac{1}{4}\binom{n}{d} \cdot \binom{n}{d-1}$
Complexity	$\frac{1}{8}\binom{n}{d}^3$

Note that the memory requirement is not stringent when compared to Algorithm 1, where a linear system of equations with about $\binom{n}{d}^2$ coefficients needs to be memorized.

In order to improve efficiency over Algorithm 2, we use arguments x of higher weight than d: Consider all arguments x with weight $d+1$ such that $f(x) = 1$. For each such x, a linear equation arises where $\binom{d+1}{d} = d + 1$ coefficients of weight d (and coefficients of lower weight) are involved. In some fraction of arguments x, exactly d coefficients of weight d were already expressed by coefficients of lower weight. Thus the remaining coefficient can be expressed as well by coefficients of lower weight. This procedure can be iterated for $w = d+2$, and so on, with higher number of coefficients of weight d involved, but with higher probability that a coefficient has already been replaced in an earlier step. The gain of efficiency for increasing weight is dependent on n and d. The necessary estimates are given in a Lemma.

Lemma 2. *Let f be a random Boolean function with n variables, and let d be the degree of an annihilator g of f. Then the following statements hold:*

a) A fraction

$$p = \frac{1}{2} + (n - d) \cdot 2^{-(d+2)} \tag{4}$$

of weight d coefficients can be replaced by lower weight coefficients by substituting all weight w arguments x with $f(x) = 1$, and with $w \leq d + 1$.

b) Suppose that according to a) a fraction p of coefficients of weight d have been replaced. Then an additional number A of coefficients can be replaced by substituting arguments of weight $w = d + 2$, where

$$A = \frac{1}{2}\binom{n}{d+2} \cdot \binom{d+2}{2}(1-p)p^{\binom{d+2}{2}-1} \tag{5}$$

Proof. a): By following steps 1 to 3 of Algorithm 2, about $\frac{1}{2}\binom{n}{d}$ coefficients of weight d have already been replaced by lower weight coefficients. There are about $\frac{1}{2}\binom{n}{d+1}$ arguments x of weight $w = d + 1$ with $f(x) = 1$. Substitute these in $g(x)$. Then in the average, for $\frac{1}{2}\binom{n}{d+1} * (d+1) * 2^{-(d+1)}$ of arguments, we have that amongst the $d+1$ weight d coefficients involved, exactly d coefficients have previously been expressed by coefficients of lower weight. Thus the remaining coefficient can be expressed by coefficients of lower weight. The average fraction of coefficients of weight d replaced by now is got by dividing by $\binom{n}{d}$ and is as claimed.

b) is similar, and is omitted. ∎

The improved algorithm is illustrated for degrees $d = 4$ and $d = 5$.

Case $d = 4$**:** Let the number of variables of f be $n \geq 20$. Search for potential annihilators of degree $d = 4$. First assume $n = 20$. Formula (4) shows that by using all arguments of weight up to $w \leq 5$, a fraction $p = 0.75$ of the $\binom{20}{4}$ coefficients of weight 4 can be replaced. Thus with $n = 20$, there remain 1211 coefficients to be replaced. According to Formula (5), an average number A of new coefficients of weight d can be replaced by using arguments of weight $d+2$. With $n = 20$, $d = 4$, and $p = 0.75$, one gets 1294. Thus with high probability (almost) all coefficients of weight $d = 4$ can be replaced. Using formulas (4) and (5) one can show that this probability quickly increases for increasing n. Hence the number of remaining unknowns (and equations) is of order $\frac{1}{2}\binom{n}{d-1}$. Thus we are able to reduce deciding the existence of annihilators of degree at most 4 from $\binom{n}{d}^3$, when using Algorithm 1, to $\frac{1}{8}\binom{n}{d-1}^3$, when using our refinement of Algorithm 2.

If $n = 32$, i.e., one of our target values, this complexity is about $\frac{1}{8}\binom{32}{3}^3 \approx 2^{34}$, compared to about $\binom{32}{4}^3 \approx 2^{45}$, when Algorithm 1 (modified to $d = 4$) would be directly applied.

Recall that the final system of linear equations to be solved, is found by substituting linear relations for coefficients of $g(x)$, for various arguments x. This should be done in a way such that it doesn't exceed the cost for solving this system. To get a linear system of largest possible rank, one should take arguments with arbitrary weight, so that all monomials in f contribute to the evaluation of f. A majority of arguments x have weight about $n/2$. Hence only about $\binom{n/2}{d}$ monomials in $g(x)$ are nonzero. Thus in this case the complexity of substituting linear expressions in $g(x)$ to get a linear equation in unknowns has complexity about $\binom{n}{d-1} \cdot \binom{n/2}{d}$. Doing this for $\binom{n}{d-1}$ equations, for values n and d under consideration, the average complexity is not larger than $\binom{n}{d-1}^3$. When taking arguments with weight close to n, one better computes the linear equation got from the weight n argument x, and then modifies this equation by setting some components in x to 0.

Case $d = 5$**:** Let $n \geq 32$. Assume $n = 32$, (the case $n > 32$ works even better). Then according to formula (4), $p = 0.7109375$. The number of coefficients of weight $d = 5$ after using all arguments of weight up to $d + 1 = 6$ is 58210. After using weight $d + 2 = 7$ arguments, we can replace another 22229 coefficients of weight 5. Hence there remain 35981, which is of the same order as $\binom{32}{4} = 35960$. As half of coefficients of weight at most 4 have already been replaced by basic step 4 of Algorithm 2, and as the case $n > 32$ is more favorable, we conclude that the remaining number of unknowns is of order $\binom{n}{4}$. Hence the complexity of deciding existence of an annihilator of degree at most 5 is of order $\binom{n}{4}^3$, e.g., for $n = 32$, it is of order 2^{45} (compared to 2^{53}, when modified Algorithm 1 would be directly applied).

The cases $d < 4$ work similar as the cases $d = 4$ and $d = 5$ just given. However, for $d = 6$, and $n < 50$, formula (4) shows that the probability p is already close to 0.5, so that in this case by using arguments with weight larger than 6 only weak refinements over the basic Algorithm 2 may be expected.

6 Bounds on the probability of annihilators' existence

In the last section we have proposed an algorithm for deciding whether a given function f admits annihilators of degree $\leq d$. However the complexity of the algorithm is strongly related with the inputs n, d and it turns out that this task becomes infeasible for $n \geq 32$ and $d \geq 6$. Hence using more inputs to the function might be an obvious solution to protect from algebraic attacks. It cannot be precluded however that finding annihilators for larger n and d may still be feasible by using methods related to Gröbner basis, although this seems open. In such a setting it is important to derive bounds on the probability that a function admits annihilators.

An easy upper bound for the probability that an n-variable balanced function admits an annihilator of degree at most d, is deduced from the minimum weight of any nonzero function of degree less or equal to d. As f is assumed to be balanced, this extends to a statement on the algebraic immunity of f:

Proposition 3 *The probability that a random n-variable balanced function f has algebraic immunity at most d is upper bounded by the number:*

$$Pb\{AI(f) \leq d\} \leq \frac{2(2^{1+n+\cdots+\binom{n}{d}} - 1)\binom{2^n - 2^{n-d}}{2^{n-1} - 2^{n-d}}}{\binom{2^n}{2^{n-1}}}. \tag{6}$$

Proof. The size of the set A of nonzero functions of degrees at most d equals $2^{1+n+\cdots+\binom{n}{d}} - 1$. For every such function g, the number of balanced functions f such that the support of g is included in $S_0(f)$ equals $N_g = \binom{2^n - wt(g)}{2^{n-1} - wt(g)}$, where $wt(g)$ denotes the Hamming weight of g. Since every such function g has weight at least 2^{n-d}, we have $\binom{2^n - wt(g)}{2^{n-1} - wt(g)} \leq \binom{2^n - 2^{n-d}}{2^{n-1} - 2^{n-d}}$. Thus, the number of balanced functions admitting an annihilator of degree at most d is smaller than or equal to $\sum_{g \in A} N_g \leq (2^{1+n+\cdots+\binom{n}{d}} - 1)\binom{2^n - 2^{n-d}}{2^{n-1} - 2^{n-d}}$; indeed, the size of a union of sets is smaller than or equal to the sum of the sizes of the sets. Since $\binom{2^n}{2^{n-1}}$ is the number of balanced functions, this completes the proof. ∎

Even though this bound is not tight, it helps us to determine the asymptotic behavior of the probability of annihilator's existence.

Theorem 3. *Let d_n be a sequence of positive integers such that $d_n \leq \mu n$ where $\mu = \frac{1}{2}(1 + \frac{\ln 2}{2} - \sqrt{(1 + \frac{\ln 2}{2})^2 - 1}) \approx 0.22$. Then*

$$Pb\{AI(f) \leq d_n\} \rightarrow 0,\ n \rightarrow \infty. \tag{7}$$

Proof. We know that, for every positive integer N and every $0 < \lambda < 1/2$:

$$\sum_{0 \le i \le \lambda N} \binom{N}{i} \le 2^N e^{-2N(1/2-\lambda)^2},$$

(e.g., see C. Carlet [5]). We deduce that for every n and every $d_n < n/2$:

$$1 + n + \cdots + \binom{n}{d_n} \le 2^n e^{-2n(1/2-d_n/n)^2},$$

and denoting the number $\frac{1/2-2^{-d_n}}{1-2^{-d_n}}$ by λ_n we have:

$$\binom{2^n - 2^{n-d_n}}{2^{n-1} - 2^{n-d_n}} \le 2^{2^n-2^{n-d_n}} e^{-2(2^n-2^{n-d_n})(1/2-\lambda_n)^2}.$$

Thus

$$(2^{1+n+\cdots+\binom{n}{d_n}} - 1)\binom{2^n - 2^{n-d_n}}{2^{n-1} - 2^{n-d_n}} \le$$

$$2^{2^n e^{-2n(1/2-d_n/n)^2} + 2^n - 2^{n-d_n}} e^{-2(2^n-2^{n-d_n})(1/2-\lambda_n)^2},$$

and therefore

$$\log_2 \left[(2^{1+n+\cdots+\binom{n}{d_n}} - 1)\binom{2^n - 2^{n-d_n}}{2^{n-1} - 2^{n-d_n}} \right] \le$$

$$2^n e^{-2n(1/2-d_n/n)^2} + 2^n - 2^{n-d_n} - 2(\log_2 e)(2^n - 2^{n-d_n})(1/2-\lambda_n)^2.$$

We have also $\binom{2^n}{2^{n-1}} \sim k 2^{2^n - n/2}$, where k is a constant, according to Stirling formula. Hence, if $n/2$ is negligible with respect to

$$2^{n-d_n} - 2^n e^{-2n(1/2-d_n/n)^2} + 2(\log_2 e)(2^n - 2^{n-d_n})(1/2-\lambda_n)^2 =$$

$$2^n \left[2^{-d_n} - e^{-2n(1/2-d_n/n)^2} + 2(\log_2 e)(1 - 2^{-d_n})(1/2-\lambda_n)^2 \right]$$

then $\frac{(2^{1+n+\cdots+\binom{n}{d_n}} - 1)\binom{2^n - 2^{n-d_n}}{2^{n-1} - 2^{n-d_n}}}{\binom{2^n}{2^{n-1}}}$ tends to zero.

A sufficient condition is that $2^{-d_n} \ge e^{-2n(1/2-d_n/n)^2}$ and that $n/2$ is negligible with respect to $2^n \left[2(\log_2 e)(1 - 2^{-d_n})(1/2-\lambda_n)^2 \right]$. We have

$$2^{-d_n} \ge e^{-2n(1/2-d_n/n)^2} \Leftrightarrow d_n \le 2n(\log_2 e)(1/2 - d_n/n)^2,$$

that is,

$$d_n/n \le 2(\log_2 e)(1/2 - d_n/n)^2.$$

The equation $x = 2(\log_2 e)(1/2 - x)^2$ is equivalent to $x \ln 2/2 = (1/2 - x)^2$, that is, $x^2 - x(1 + \frac{\ln 2}{2}) + \frac{1}{4} = 0$, which roots are both positive.

Its smallest root is μ. Thus $d_n \leq \mu n$ implies $2^{-d_n} \geq e^{-2n(1/2-d_n/n)^2}$. If $d_n \leq \mu n$, then

$$(1-2^{-d_n})(\frac{1}{2}-\lambda_n)^2 = (1-2^{-d_n})\left(\frac{1}{2} - \frac{1/2-2^{-d_n}}{1-2^{-d_n}}\right)^2 = \frac{2^{-2d_n}}{4(1-2^{-d_n})} \geq \frac{2^{-2\mu n}}{4(1-2^{-\mu n})}.$$

Hence, since 2μ is strictly smaller than 1, then $n/2$ is negligible with respect to

$$2^n\left[2(\log_2 e)(1-2^{-d_n})(1/2-\lambda_n)^2\right].$$

□

For practical applications we are interested in concrete values of this bound for moderate n rather than the asymptotical values. For instance, we can compute the probability that a random balanced function f in $n = 32$ variables admits annihilators of degree $d \leq 6$. In view of Theorem 3, $d = 6$ satisfies the inequality $d \leq 0.22n$ for $n = 32$. Then computing (6) for $n = 32$, $d = 6$, gives a probability of order 10^{-300} which is negligibly small. Notice that in this case, due to the complexity reasons, we cannot confirm the (non)existence of annihilators through Algorithm 2.

However the upper bound as derived above is based on the property that all annihilators have weights at least 2^{n-d}. This bound can be sharpened by using some known results on the weight distribution and enumeration of the codewords in the Reed-Muller code $\mathcal{R}(d,n)$. Let us denote by A_w the number of codewords of weight w in $\mathcal{R}(d,n)$; then $A_{2^{n-d}}$ equals $2^d\prod_{i=0}^{n-d-1}\frac{2^{n-i}-1}{2^{n-d-i}-1}$ due to McWilliams-Sloane [16]. Furthermore, Kasami and Tokura [14] have done the weight enumeration of codewords of weight w in $\mathcal{R}(d,n)$ for all $2^{n-d} < w < 2^{n-d+1}$. These results are found in [16, pg. 446] and can be used to derive a tighter upper bound from the following easy improvement of Proposition 3:

Theorem 4. *For a random balanced function $f \in \mathcal{B}_n$ the upper bound on the probability, denoted Pb^d, that $AI(f) \leq d$ is given by*

$$Pb^d \leq \sum_{w=2^{n-d}}^{w<2^{n-d+1}} A_w \cdot \frac{\binom{2^n-w}{2^{n-1}-w}}{\binom{2^n}{2^{n-1}}} + \left(2^{\sum_{i=0}^{d}\binom{n}{i}} - \sum_{w=2^{n-d}}^{w<2^{n-d+1}} A_w\right) \cdot \frac{\binom{2^n-2^{n-d}}{2^{n-1}-2^{n-d}}}{\binom{2^n}{2^{n-1}}}. \quad (8)$$

Note that, for every w, we have $\frac{\binom{2^n-w}{2^{n-1}-w}}{\binom{2^n}{2^{n-1}}} \leq \left(\frac{1}{2}\right)^w$.

Remark 2 *This upper bound can be further tightened by using more values of w for which the exact number of codewords is known. This has been done in [15] for the weights w in the range $2^{n-d} \leq w < 2.5 \cdot 2^{n-d}$.*

For the bound of Theorem 4, it seems to be much harder to estimate the value of μ as it has been done in Theorem 3. By computations one can deduce the same behavior of this bound but with slightly shifted limit value, that is $\mu' \approx 0.27$. This gives a better value than Theorem 3 as for increasing n the sequence $d_n \leq \mu' n$ has a larger range.

The upper bounds above are important tools for estimating the security of a stream cipher. For instance assuming that the computational complexity of breaking a cipher whose multiples are of degrees strictly greater than say $d = 5$, then Theorem 4 gives $n = 18$ which is the lowest value of n such that the probability that there exists annihilators of degree $d \leq 5$ is close to zero. Hence assuming that f has no particular structure that might be exploited, the value of $n = 18$ and the key length of $k = 128$ should guarantee that the known attacks are infeasible. Assuming the existence of multiples/annihilators of degree $d = 6$ this would give a computational complexity of order $\approx \binom{128}{6}^{\omega} = (2^{32})^{\omega}$, which for $\omega = 3$ yields 2^{96}. If a more secure cipher is preferred then the obvious method is to increase n. In Table 1 below we list some other interesting cases. Each entry relates a given degree of annihilators d to the minimum value of n for which $Pb\{AI(f) \leq d\} \approx 0$. We apply the results above to the stream cipher LILI-128,

$n; P_b$	$d = 5$	$d = 6$	$d = 7$	$d = 8$
	$18; 10^{-1134}$	$22; 10^{-6326}$	$26; 10^{-23138}$	$31; 10^{-10^7}$

Table 1. Upper bound on the probability for the annihilators

for which 14 linearly independent annihilators of degree $d = 4$ have been found in [8].

Example 2 *In [8], Courtois and Meier (see also [11]) have investigated the algebraic properties of LILI-128. They have found that the function f in $n = 10$ variables used in LILI-128 is rather weak, since one could find 14 linearly independent annihilators of degree 4.*

Note that the probability $Pb\{AI(f) \leq 5\}$ is equal to 1 due to the Theorem 6.0.1 in [8]. The upper bound is not tight for $d = 4, 5$ giving a probability greater than 1. However, applying the upper bound for the case $d = 3$ one deduces that

$$Pb\{AI(f) \leq 3\} \leq 0.30 \cdot 10^{-24}.$$

□

Example 2 shows that the upper bound in particular for low values of n is not tight. However, Table 1 illustrates that this bound gives very strong estimates for larger n of interest.

Acknowledgment We are indebted to Jean-Pierre Tillich for hepful discussions.

References

1. F. Armknecht and Krause M. Algebraic attacks on stream combiners with memory. In *Advances in Cryptology—CRYPTO 2003*, volume LNCS 2729, pages 162–176. Springer-Verlag, 2003.

2. P. Camion, C. Carlet, P. Charpin, and N. Sendrier. On correlation-immune functions. In *Advances in Cryptology—EUROCRYPT'91*, volume LNCS 547, pages 86–100. Springer-Verlag, 1991.
3. A. Canteaut and M. Trabbia. Improved fast correlation attacks using parity-check equations of weight 4 and 5. In *Advances in Cryptology—EUROCRYPT 2000*, volume LNCS 1807, pages 573–588. Springer-Verlag, 2000.
4. C. Carlet. A larger class of cryptographic Boolean functions via a study of the Maiorana-McFarland constructions. In *Advances in Cryptology—CRYPTO 2002*, volume LNCS 2442, pages 549–564. Springer-Verlag, 2002.
5. C. Carlet. On the algebraic thickness and non-normality of Boolean functions. In *Proceedings of 2003 IEEE Information Theory Workshop, Paris, France*, 2003.
6. N. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology—CRYPTO 2003*, volume LNCS 2729, pages 176–194. Springer-Verlag, 2003.
7. N. Courtois and Pieprzyk J. Cryptanalysis of block ciphers with overdefined systems of equations. In *Advances in Cryptology—ASIACRYPT 2002*, volume LNCS 2501. Springer-Verlag, 2002.
8. N. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology—EUROCRYPT 2003*, volume LNCS 2656, pages 346–359. Springer-Verlag, 2003.
9. N. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. Extended version of [8], available at http://www.cryptosystem.net/stream/, 2003.
10. J. F. Dillon. Elementary Haddamard Difference Sets. Ph. D. thesis, University of Maryland, U.S.A., 1974.
11. J.-Ch. Faugère and G. Ars. An algebraic cryptanalysis of nonlinear filter generators using Gröbner bases. Available on the web, 2003. http://www.inria.fr/rrrt/rr-4739.html.
12. J. Dj. Golić. On the security of nonlinear filter generators. In *Fast Software Encryption'96*, volume LNCS 1039, pages 173–188. Springer-Verlag, 1996.
13. T. Johansson and F. Jönsson. Fast correlation attacks through reconstruction of linear polynomials. In *Advances in Cryptology—CRYPTO 2000*, volume LNCS 1880, pages 300–315. Springer-Verlag, 2000.
14. T. Kasami and N. Tokura. On the weight structure of Reed-Muller codes. *IEEE Trans. on Inform. Theory*, IT-16(6):pages 752–759, 1970.
15. T. Kasami, N. Tokura, and S. Asumi. On the weight enumeration of weights less than $2.5d$ of Reed-Muller codes. *Information and Control*, vol. 30(4):pages 380–395, 1974.
16. F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam, 1977.
17. W. Meier and O. Staffelbach. Fast correlation attacks on stream ciphers. In *Advances in Cryptology—EUROCRYPT'88*, volume LNCS 330, pages 301–314. Springer-Verlag, 1988.
18. E. Pasalic. Degree optimized resilient Boolean functions from Maiorana-McFarland class. In *9-th IMA Conference on Cryptography and Coding*, 2003.
19. T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Trans. on Inform. Theory*, IT-30:pages 776–780, 1984.