Efficient Computation of Algebraic Immunity for Algebraic and Fast Algebraic Attacks

Frederik Armknecht[1], Claude Carlet[2], Philippe Gaborit[3], Simon Künzli[4], Willi Meier[4], and Olivier Ruatta[3]

[1] Universität Mannheim, 68131 Mannheim (Germany), armknecht@th.informatik.uni-mannheim.de
[2] INRIA, Projet CODES, BP 105, 78153 Le Chesnay Cedex; also with Univ. of Paris 8 (France), claude.carlet@inria.fr
[3] Université de Limoges, 87060 Limoges (France), {gaborit,olivier.ruatta}@unilim.fr
[4] FH Nordwestschweiz, 5210 Windisch (Switzerland), {simon.kuenzli,willi.meier}@fhnw.ch

Abstract. In this paper we propose several efficient algorithms for assessing the resistance of Boolean functions against algebraic and fast algebraic attacks when implemented in LFSR-based stream ciphers. An algorithm is described which permits to compute the algebraic immunity d of a Boolean function with n variables in $\mathcal{O}(D^2)$ operations, for $D \approx \binom{n}{d}$, rather than in $\mathcal{O}(D^3)$ operations necessary in all previous algorithms. Our algorithm is based on multivariate polynomial interpolation. For assessing the vulnerability of arbitrary Boolean functions with respect to fast algebraic attacks, an efficient generic algorithm is presented that is not based on interpolation. This algorithm is demonstrated to be particularly efficient for symmetric Boolean functions. As an application it is shown that large classes of symmetric functions are very vulnerable to fast algebraic attacks despite their proven resistance against conventional algebraic attacks.

Key words. Algebraic Attacks, Algebraic Degree, Boolean Functions, Fast Algebraic Attacks, Stream Ciphers, Symmetric Functions.

1 Introduction

Many keystream generators consist of combining several linear feedback shift registers (LFSRs) and possibly some additional memory. One example is the E_0 keystream generator which is part of the Bluetooth standard. LFSRs are very efficient in hardware and can be designed such that the produced bitstream has maximum period and good statistical properties. Various approaches to the cryptanalysis of LFSR-based stream ciphers were discussed in literature (e.g., time-memory-tradeoff, fast correlation attacks or BDD-based attacks). For some keystream generators, algebraic attacks and fast algebraic attacks outmatched all previously known attacks [3, 12, 13].

For LFSR-based filter or combining generators, their security mainly relies on a nonlinear Boolean output function f filtering the contents of one LFSR or combining the outputs of several ones. The present paper studies the resistance of this kind of stream ciphers to (fast) algebraic attacks.

In view of algebraic attacks, the notion of algebraic immunity (or annihilator immunity) has been introduced (the algebraic immunity $\mathcal{AI}$ of a Boolean function f is the minimum value of d such that f or $f+1$ admits a function g of degree d such that $fg = 0$). The construction of Boolean functions for LFSR-based stream ciphers with large algebraic immunity achieved much attention recently, [5–7, 15, 17]. However, many of these functions do not allow for other good cryptographic properties like large non-linearity or large orders of resiliency, and as will be shown later, have undesirable properties with regard to fast algebraic attacks. It seems therefore relevant to be able to efficiently determine the immunity of existing and newly constructed Boolean functions against algebraic and fast algebraic attacks.

Until now, the best algorithms known for computing the algebraic immunity d of a function with n variables work roughly in $\mathcal{O}(D^3)$ operations, where $D \approx \binom{n}{d}$. This is impractical for functions with 20 or more variables. In this paper, we give an algorithm which computes the $\mathcal{AI}$ of a function in $\mathcal{O}(D^2)$ operations. The algorithm is based on multivariate polynomial interpolation, and it is applied to two particular families of Boolean functions: the inverse functions and the Kasami power functions. The quadratic nature of the algorithm is experimentally verified, and for the first time, the $\mathcal{AI}$ of a function with 20 variables is computed to be $\mathcal{AI} = 9$.

Resistance against fast algebraic attacks is not fully covered by algebraic immunity, as has been demonstrated, e.g., by a fast algebraic attack on the eSTREAM candidate SFINKS, [11]. For determining immunity against fast algebraic attacks, we give a new algorithm that is based on methods different from interpolation, and that for general Boolean functions allows to efficiently assess immunity against fast algebraic attacks. The complexity of our second algorithm is in $\mathcal{O}(DE^2)$, where $E \approx \binom{n}{e}$ and e in many cases of interest is much smaller than d. This compares favorably with the known algorithms, which are in $\mathcal{O}(D^3)$. The algorithm is applied to several of the above mentioned classes of Boolean functions with optimal algebraic immunity, including symmetric Boolean functions, like the majority functions. Symmetric functions are attractive as the hardware complexity grows only linearly with the number of input variables. However, it is shown in this paper that the specific structure of these functions can be exploited in a much refined algorithm for determining resistance against algebraic attacks that is particularly efficient. It is concluded that large classes of symmetric functions are very vulnerable to fast algebraic attacks despite their optimal algebraic immunity. A symmetric function would not be implemented by itself but rather in combination with other nonlinear components in stream ciphers. It seems nevertheless essential to know the basic cryptographic properties of each component used.

The paper is organized as follows. In Section 2, the basics of algebraic and fast algebraic attacks are described. Section 3 derives an algorithm for efficient computation of the algebraic immunity as well as a modified algorithm to determine all minimal degree annihilators. In Section 4, an algorithm for efficient computation of immunity against fast algebraic attacks is presented. In Section 5, the algorithm is adapted and improved for symmetric functions, and it is proven that the class of majority functions which have maximum $\mathcal{AI}$ is very vulnerable to fast algebraic attacks. We finally conclude in Section 6.

2 Algebraic Attacks and Fast Algebraic Attacks

2.1 Algebraic Attacks

For an LFSR L with N entries filtered by a Boolean function f with n variables, algebraic attacks consist of two steps [12]:

- **First step**. Finding functions g of low degree d such that $fg = 0$ or $(f + 1)g = 0$. Until this paper, the complexity of this step was roughly in D^3, for $D := \sum_{i=0}^{d} \binom{n}{i}$ (which is about $\binom{n}{d}$ for $d < n/2$) and where 3 is taken for the exponent of the matrix inversion.
- **Second step**. Solving a nonlinear system of multivariate equations $g(L^i(x_1, \dots, x_N)) = 0$ for adequate i, induced by the functions g of the annihilator sets $\mathrm{Ann}(f)$ and $\mathrm{Ann}(f + 1)$. Usually this system is solved by linearization with a complexity of D_N^3 for $D_N := \sum_{i=0}^{d} \binom{N}{i}$. The number of required bits of keystream is proportional to D_N, whereas this value can be reduced if several annihilators of f and/or $f \oplus 1$ with minimum degree are known. Alternatively, this system can be solved by Gröbner basis, but then the complexity of solving is difficult to evaluate, see [4, 19].

The lowest degree of the function $g \neq 0$ for which $fg = 0$ or $(f+1)g = 0$ is called the algebraic (or annihilator) immunity $\mathcal{AI}$ of f. In [12] it has been shown that for any function f with n-bit input vector, functions $g \neq 0$ and h exist, with $fg = h$ such that e and d are at most $\lceil n/2 \rceil$. This implies that $\mathcal{AI}(f) \leq \lceil n/2 \rceil$.

2.2 Fast Algebraic Attacks

Fast algebraic attacks were introduced by Courtois in [13]. They were confirmed and improved later by Armknecht in [3] and Hawkes and Rose in [20]. A prior aim of fast algebraic attacks is to find a relation $fg = h$ with $e := \deg g$ small and $d := \deg h$ larger. In classical algebraic attacks, the degree d of h would necessarily lead to considering a number of unknowns of the order of D_N. In fast algebraic attacks, one considers that the sequence of the functions $h(L^i(x_1, \cdots, x_N))$ can be obtained as an LFSR with linear complexity D_N. One uses then the Berlekamp-Massey algorithm to eliminate all monomials of degree superior to e in the equations, such that eventually one only needs to solve a system in $E_N := \sum_{i=0}^{e} \binom{N}{i}$ unknowns. The complexity of fast algebraic attacks can be summarized in these four steps:

- **Relation search step**. One searches for functions g and h of low degrees such that $fg = h$. For g and h of degrees e and d respectively, with associated values $D := \sum_{i=0}^{d} \binom{n}{i}$ and $E := \sum_{i=0}^{e} \binom{n}{i}$, such g and h can be found when they exist by solving a linear system with $D + E$ equations, and with complexity $\mathcal{O}((D + E)^3)$. Usually one considers $e < d$.
- **Pre-computation step**. In this step, one searches for particular linear relations which permit to eliminate monomials with degree greater than e in the equations. This step needs a sequence of $2D_N$ bits of stream and has a complexity of $\mathcal{O}(D_N \log^2(D_N))$, see [20].
- **Substitution step**. At this step, one eliminates the monomials of degrees greater than e. This step has a natural complexity in $\mathcal{O}(E_N^2 D_N)$ but using discrete Fourier transform, it is claimed in [20] that a complexity $\mathcal{O}(E_N D_N \log(D_N))$ can be obtained.
- **Solving step**. One solves the system with E_N linear equations in $\mathcal{O}(E_N^3)$.

Notice that, for arbitrary non-zero functions f, g, h, the relation $fg = h$ implies $fh = h$, thus we have $d \geq \mathcal{AI}(f)$ and we can restrict to values e with $e \leq d$. Fast algebraic attacks are always more efficient than conventional algebraic attacks if $d = \mathcal{AI}(f)$ and $e < d - 1$. In case that e turns out to be large for this d, it is of interest to determine the minimum e where d is slightly larger than $\mathcal{AI}(f)$.

3 Efficient Computation of the Algebraic Immunity

In this section, we present an algorithm which computes the algebraic immunity $\mathcal{AI}$ of a Boolean function in $\mathcal{O}(D^2)$ operations. In particular, the algorithm returns a non-zero annihilator of minimum degree d, without necessitating a prior guess of d. The algorithm is based on the notion of multivariate polynomial interpolation, it generalizes the classical incremental Newton interpolation algorithm to the multivariate case. We also explain how to modify the algorithm to return the set of *all* non-zero annihilators with minimum degree. Eventually we give experimental results of our algorithm.

3.1 Multivariate Lagrange Interpolation

Before stating what is the multivariate Lagrange interpolation problem when it is specified to binary polynomials, we need to introduce some notation. We denote by $\mathbb{F}$ the finite field GF(2) and by $\mathbb{F}^k$ the vector space of dimension k over $\mathbb{F}$. Consider $x := x_1, \ldots, x_k$ a set of k binary variables, $\alpha := (\alpha_1, \ldots, \alpha_k) \in \mathbb{F}^k$ a multi-index, $z := (z_1, \ldots, z_k)$ an element of $\mathbb{F}^k$. We denote $x^\alpha := x_1^{\alpha_1} \cdots x_k^{\alpha_k}$ and $z^\alpha := z_1^{\alpha_1} \cdots z_k^{\alpha_k}$. Let $E := \{\alpha_1, \ldots, \alpha_D\} \subseteq \mathbb{F}^k$ be a set of multi-indices, then we denote by $x^E := \{x^{\alpha_1}, \ldots, x^{\alpha_D}\}$ the set of associated monomials. We identify the ring of boolean functions in n variables with $\mathbb{F}[x]/\langle x_i^2 - x_i, i = 1, \ldots, n\rangle$, the quotient ring of the ring of polynomials with coefficients in $\mathbb{F}$ by the ideal generated by the relations $x_i^2 - x_i$, $i \in \{1, \ldots, n\}$. We will use, explicitly or not, several times this identification. In our framework, the multivariate Lagrange problem can be stated as follows:

Problem 1. Let $E := \{\alpha_1, \ldots, \alpha_D\} \subseteq \mathbb{F}^n$, $\mathcal{Z} := \{z_1, \ldots, z_D\} \subseteq \mathbb{F}^n$ and $\bar{v} := (v_1, \ldots, v_D) \in \mathbb{F}^D$. Does there exist a polynomial $g \in \mathbb{F}[x_1, \ldots, x_n]$ whose monomial support is included in x^E and such that $g(z_i) = v_i$, $\forall i \in \{1, \ldots, D\}$?

Remark 1. The general multivariate Lagrange interpolation problem has been addressed in [23], but the proposed algorithm has cubic complexity (on the number of monomials). We will present an algorithm with a quadratic complexity over $\mathbb{F}$ instead.

An answer to Problem 1 in terms of existence and uniqueness is presented by means of the following definition:

Definition 1. *Let $\mathcal{Z} := \{z_1, \ldots, z_D\} \subseteq \mathbb{F}^n$ and $E := \{\alpha_1, \ldots, \alpha_D\} \subseteq \mathbb{F}^n$, we define the Vandermonde matrix as*

$$V_{\mathcal{Z},E} := \begin{pmatrix} z_1^{\alpha_1} & \cdots & z_1^{\alpha_D} \\ \vdots & \ddots & \vdots \\ z_D^{\alpha_1} & \cdots & z_D^{\alpha_D} \end{pmatrix} , \tag{1}$$

and we define the Vandermonde determinant to be $v_{\mathcal{Z},E} := \det(V_{\mathcal{Z},E})$.

Proposition 1. *There exists an unique solution $g \in \mathbb{F}[x]$ to Problem 1 if $v_{\mathcal{Z},E} \neq 0$. Furthermore, the solution g is given by $g(x) = \bigoplus_{j=1}^{D} g_{\alpha_j} x^{\alpha_j}$, where the vector $\bar{g} := (g_{\alpha_1}, \ldots, g_{\alpha_D})^t$ is the only solution of the system*

$$V_{\mathcal{Z},E}\ \bar{g} = \bar{v} \ . \tag{2}$$

Remark 2. Given the set $\mathcal{Z} := \{z_1, \ldots, z_D\} \subseteq \mathbb{F}^n$, the existence of a set $E := \{\alpha_1, \ldots, \alpha_D\} \subseteq \mathbb{F}^n$ such that $v_{\mathcal{Z},E} \neq 0$ is ensured since it is enough to take for E the set of monomials which are not in the monomial ideal generated by the leading monomials of a Gröbner basis of the ideal of the polynomials vanishing at each point of $\mathcal{Z}$.

With the following proposition, the minimum annihilator problem can be reduced to a multivariate Lagrange interpolation problem:

Proposition 2. *Let f be a Boolean function, $\mathcal{Z} := f^{-1}(1)$ and E such that x^E is the complementary of the monomial ideal generated by the leading monomials of a Gröbner basis for a graduated order of the ideal of the polynomials vanishing at each point of $\mathcal{Z}$. Then, if $\beta \notin E$ is of minimum weight, the function R_β defined below is a minimum-degree annihilator of f,*

$$R_\beta := \det \begin{pmatrix} x^\beta & x^{\alpha_1} & \ldots & x^{\alpha_D} \\ z_1^\beta & z_1^{\alpha_1} & \ldots & z_1^{\alpha_D} \\ \vdots & \vdots & \ddots & \vdots \\ z_D^\beta & z_D^{\alpha_1} & \ldots & z_D^{\alpha_D} \end{pmatrix} .$$

Furthermore, $R_\beta = x^\beta \oplus g$ where g is the solution of Problem 1 with $\bar{v} = (z_1^\beta, z_2^\beta, \ldots, z_D^\beta)$.

Proof. The function $R_\beta(x)$ is an annihilator of f, as for an argument $x \in \mathcal{Z}$ the above matrix becomes singular. In addition, R_β has minimum degree because E is the complementary of the monomial ideal generated by the leading terms of a Gröbner basis for a graduated monomial order. The relation $R_\beta = x^\beta \oplus g$ is obtained by developing the determinant defining R_β with respect to the first row, and by considering g obtained with Cramer's rule in Eq. 2. $\square$

3.2 General Description of the Algorithm

The general idea of the algorithm is to apply Prop. 2 incrementally with a linear complexity at each step. Let us introduce some more notation: E^d is the set of all α of weight equal to d. Then $E^{\leq d} := E^0 \cup \ldots \cup E^d$ (ordered by increasing weight) and $E_i := \{\alpha_1, \ldots, \alpha_i\}$, which are the first i elements of $E^{\leq d}$. Let $\mathcal{Z} := f^{-1}(1) \subseteq \mathbb{F}^n$ (with arbitrary ordering) and $\mathcal{Z}_i := \{z_1, \ldots, z_i\}$. We assume $v_{\mathcal{Z}_i, E_i} \neq 0$ for all $i \in \{1, \ldots, |\mathcal{Z}|\}$, this condition[5] is sufficient to apply Prop. 2 on the sets $\mathcal{Z}_i$, E_i.

Then the algorithm works as follows: apply Prop. 2 for an intermediate set of points $\mathcal{Z}_i$ and an associated set of exponents E_i, with $\beta = \alpha_{i+1}$. A particular solution $g_i = R_\beta \oplus x^\beta$ is an intermediate annihilator of f on the set $\mathcal{Z}_i$. If one can verify that g_i is also an annihilator of f on the global set $\mathcal{Z}$, then a minimum degree annihilator of f is found. Otherwise, one considers a new point z_{i+1} and $\mathcal{Z}_{i+1}$ with associated set of exponents E_{i+1}, until an annihilator of f on $\mathcal{Z}$ is found.

Remark 3. Notice that the original interpolation problem with $\bar{v} = 0$ is turned into a sequence of interpolation problems with (in general) non-zero $\bar{v}$, depending on the exponent α_{i+1} used at each step. In particular, the fact $\bar{v} = 0$ on $f^{-1}(1)$ is used implicitly in the computation of the ordered set E associated to $f^{-1}(1)$.

For each intermediate step, the updating procedure can be done in linear time, resulting in an overall complexity of $\mathcal{O}(D^2)$ rather than $\mathcal{O}(D^3)$. In fact, this is a multivariate generalization of the Newton interpolation scheme: recall that a Newton basis for the polynomial interpolation problem allows to introduce interpolation nodes one by one (without the requirement to recalculate previous coefficients). In addition, the Newton basis leads to a triangular Vandermonde matrix, which can be solved in quadratic time (on the number of interpolation nodes).

3.3 Computing a Minimum Degree Annihilator

Define $V_i := V_{\mathcal{Z}_i, E_i}$, and consider an LU-decomposition of V_i, i.e. $V_i = L_i U_i$, where U_i is triangular superior and L_i is triangular inferior. Then, the system $V_i \bar{g}_i = \bar{v}_i$ with $\bar{v}_i := (v_1, \ldots, v_i)$ is equivalent to $U_i \bar{g}_i = L_i^{-1} \bar{v}_i$, and the solution $\bar{g}_i$ can be found by solving two triangular systems (i.e computing the inverses of

[5] Such kind of ordered sets of points and exponents always exists and can be computed incrementally in quadratic time (see [23]), so we do not lose any generality.

U_i and L_i). If the polynomial associated to $\bar{g}_i$ is not an annihilator of f, then we solve the system for V_{i+1} and $\bar{v}_{i+1} = (\bar{v}_i, v_{i+1},)^t$. However, instead of computing a complete LU-decomposition of V_{i+1}, we write

$$V_{i+1} = \begin{pmatrix} V_i & C_{i+1} \\ R_i & z_{i+1}^{\alpha_{i+1}} \end{pmatrix} = \begin{pmatrix} L_i & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} U_i & L_i^{-1} C_{i+1} \\ R_i & z_{i+1}^{\alpha_{i+1}} \end{pmatrix}$$

with $C_{i+1} := (z_1^{\alpha_{i+1}}, \ldots, z_i^{\alpha_{i+1}})^t$ and $R_i := (z_{i+1}^{\alpha_1}, \ldots, z_{i+1}^{\alpha_i})$. Consequently, knowledge of an LU-decomposition of V_i yields an almost LU-decomposition of V_{i+1} (with the exception of R_i). This is a basic fact usually exploited to design efficient LU-factorization algorithms.

In our framework, one can avoid a direct computation of L_i as follows. Denote $X_i := (x_1, \ldots, x_i)^t$, where the elements x_j are considered as indeterminate, and denote $P_i(x_1, \ldots, x_i) := L_i^{-1} X_i$. Then we have

$$V_{i+1} = \begin{pmatrix} L_i & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} U_i & P_i(z_1^{\alpha_{i+1}}, \ldots, z_i^{\alpha_{i+1}}) \\ z_{i+1}^{\alpha_1} \cdots z_{i+1}^{\alpha_i} & z_{i+1}^{\alpha_{i+1}} \end{pmatrix} \tag{3}$$

$$\bar{v}_{i+1} = \begin{pmatrix} L_i & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} P_i(v_1, \ldots, v_i) \\ v_{i+1} \end{pmatrix} . \tag{4}$$

Consequently, the system of equations $V_{i+1}\bar{g}_{i+1} = \bar{v}_{i+1}$ is equivalent to

$$\begin{pmatrix} U_i & P_i(z_1^{\alpha_{i+1}}, \ldots, z_i^{\alpha_{i+1}}) \\ z_{i+1}^{\alpha_1} \cdots z_{i+1}^{\alpha_i} & z_{i+1}^{\alpha_{i+1}} \end{pmatrix} \bar{g}_{i+1} = \begin{pmatrix} P_i(v_1, \ldots, v_i) \\ v_{i+1} \end{pmatrix} . \tag{5}$$

Triangulation of the left matrix is an easy task since U_i is triangular (it is achieved by elimination of the i first entries of the last row by row operations). The same operations are carried out on the right matrix. This yields U_{i+1} and $P_{i+1}(x_1, \ldots, x_{i+1})$, and the system $U_{i+1}\bar{g}_{i+1} = P_{i+1}(v_1, \ldots, v_{i+1})$ can be solved. If the polynomial associated to $\bar{g}_{i+1}$ is not an annihilator of f (i.e. if $\exists z \in \mathcal{Z}$ such that $g(z) \neq 0$), the subsequent elements in $E^{\leq d}$ and $\mathcal{Z}$ are added and so on. Practically, the only points introducing new constraints on the annihilator are those for which the input polynomial does not vanish already. The method terminates because the degree of the annihilator is bounded. Denote by t be the number of iterations of Alg. 1.

As an input of the algorithm, we do not take a monomial expansion of f, but the vector of its evaluation at points z_i. In the case $t = |\mathcal{Z}|$, this vector can be computed with asymptotically $\mathcal{O}(t \log(t))$ operations using a method based on fast Fourier transform, and more easily in $\mathcal{O}(tN_m)$ operations over the ground field (where N_m is the number of monomials in the algebraic normal form of f) by simply adding the evaluation of each monomial at the t points. The algorithm incrementally computes the values of the annihilator at every point and lifts them in the monomial basis in order to compute the power expansion. Let us discuss the most costly operations at the i^{th} step of the algorithm:

- The triangulation in step 4 requires i arithmetic operations. As U_i is already a upper triangular matrix, we only need to eliminate the first $i - 1$ entries

Algorithm 1 Computation of an annihilator of minimum degree

Input: f, $\mathcal{Z} := f^{-1}(1)$, $E^{\leqslant \lceil n/2 \rceil}$.
Output: An annihilator of f of minimum degree.
1: Initialization: $U_1 \leftarrow (z_1^{\alpha_1})$, $v_1 \leftarrow f(z_1) \oplus 1$, $\bar{g} \leftarrow 1$, $P \leftarrow (x_1)$, $i \leftarrow 1$.
2: **while** the polynomial associated to $\bar{g}$ is not an annihilator of f **do**
3: $\quad i \leftarrow i + 1$.
4: $\quad \begin{pmatrix} U_i & P(z_1^{\alpha_i}, \dots, z_{i-1}^{\alpha_i}) \\ z_i^{\alpha_1} \dots z_i^{\alpha_{i-1}} & z_i^{\alpha_i} \end{pmatrix} \overset{\text{row op.}}{\mapsto} \begin{pmatrix} U_i & P(z_1^{\alpha_i}, \dots, z_i^{\alpha_i}) \\ 0 \dots 0 & \end{pmatrix} =: U_{i+1}$.
5: $\quad$ Use the same row operations from $\left(P(z_1^{\alpha_i}, \dots, z_{i-1}^{\alpha_i}), z_i^{\alpha_i}\right) \mapsto P(z_1^{\alpha_i}, \dots, z_i^{\alpha_i})$ to perform the update $(P(v_1, \dots, v_{i-1}), v_i) \mapsto P(v_1, \dots, v_i)$.
6: $\quad$ Solve $U_i \bar{g}_i = P(v_1, \dots, v_i)$ with $\bar{g}_i = (g_1, \dots, g_i)$.
7: **end while**
8: Output $g(x) := \bigoplus_{j=1}^{i} g_j x^{\alpha_j}$.

in the last row, and update the entry in the bottom right corner. This is done by replacing $z_i^{\alpha_1}, \dots, z_i^{\alpha_{i-1}}$ by 0 and $z_i^{\alpha_i}$ by $z_i^{\alpha_i} - \sum_{j=1}^{i-1} z_i^{\alpha_1} \cdot P_{i,j}$ where $(P_{i,1}, \dots, P_{i,i-1}) = P(z_1^{\alpha_i}, \dots, z_{i-1}^{\alpha_i})$.
- The updating process of P requires i arithmetic operations.
- Solving the system in step 6 basically requires i^2 arithmetic operations. However, this is also feasible with i arithmetic operations by the following remark, allowing to correct g_i in order to compute g_{i+1}:

$$U_{i+1}\ g_{i+1} = \begin{pmatrix} U_i & P_i(z_1^{\alpha_{i+1}}, \dots, z_i^{\alpha_{i+1}}) \\ 0 & * \end{pmatrix} \begin{pmatrix} g_i' \\ * \end{pmatrix} = \begin{pmatrix} P_i(v_1, \dots, v_i) \\ v_{i+1} \end{pmatrix} .$$

We do not introduce any new complex computation to check whether g is an annihilator of f. Namely, we compute the values of g at points which are not introduced yet. This can be done by updating a vector storing the evaluations of g at each point considered so far, where a new step leads to a linear number of operations (corresponding to the number of coordinates). Again, the overal cost of this computation is quadratic on the number of points.

The arithmetic complexity $AC(N)$ of the proposed algorithm is given by $AC(N) = AC(N-1) + \text{const} \cdot i + \mathcal{O}(D)$. An simple computation shows that $AC(N) = \mathcal{O}(t^2 + tD)$. Since t is the number of monomials occurring in a minimum degree annihilator of f, t has the same order of magnitude as D. This is summarized in the following proposition:

Proposition 3. *The arithmetic complexity of Alg. 1 to compute the minimum degree d of an annihilator of f is $\mathcal{O}(D^2)$.*

In order to obtain the quadratic behavior, it is necessary to handle memory allocation with care (in particular, management of the extension operations on the matrix are delicate, and a bad memory allocation leads to an implementation cubic in space and time). We finally remark that the above method can also be used to construct functions with high algebraic immunity.

3.4 Computing All Minimum Degree Annihilators

In this section, we explain how to modify Alg. 1 to compute all minimum-degree annihilators g of a polynomial f. Notice that $\mathrm{Ann}(f,d) := \{g \in \langle x^{E^{\leq d}} \rangle \subseteq \mathbb{F}[x] | fg = 0\}$ is a vector space. Consequently, we only have to compute a basis of $\mathrm{Ann}(f,d)$, and this for the minimum value of d. The idea of the method proposed here is to run Alg. 1 until we find the first annihilator together with d. Then, the algorithm searches for further annihilators, considering only exponents in $E^{\leq d}$. In addition, if α_i is the exponent lastly introduced and resulting in an annihilator, we can execute a further search without α_i (this can be implemented by backtracking the last update). The reason is that if g and $g' \neq g$ are both annihilators which contain x^{α_i}, then one can construct another annihilator $g \oplus g'$ which is independent of x^{α_i}. Hence, the new algorithm can still be run incrementally, and it terminates after introduction of α^D. As the number of steps required to find the first annihilator is of the same order of magnitude as D, the asymptotic performance of the new algorithm does not increase. This is resumed in the following proposition:

Proposition 4. *The above modifications of Alg. 1 allow to compute the minimum degree d of an annihilator of f, and a basis of $Ann(f,d)$, using $\mathcal{O}(D^2)$ arithmetic operations.*

3.5 Experimental Results

In this section, we apply Alg. 1 to two particular families of Boolean power functions: the inverse functions and the Kasami type functions (see [10]). We verified that an implementation of the algorithm in C code followed the announced quadratic time complexity on the number of variables.

The inverse function is of particular interest, since this function is used with $n = 8$ variables in the S-box of AES, and almost directly as a filtering function in SFINKS [6]. For different values of n, Tab. 1 lists the power exponent of the function f (which is equal to -1 here), its weight, its algebraic degree, its nonlinearity and its algebraic immunity.

The Kasami functions in n variables have exponents of the form $2^{2k} - 2^k + 1$ with $\gcd(k,n) = 1$ and $k \leq n/2$. These functions are of interest since we can see that, for the number n of variables which is currently usual in cryptography, they have a high algebraic immunity.[6] We consider several Kasami type exponents (where $\gcd(k,n)$ may be different from 1), see Tab. 1. For $n = 12, 16, 20$, we converted non-balanced functions to balanced ones by flipping the first entries in the truth tables. For the first time, we accomplish computation of the $\mathcal{AI}$ of a function with 20 variables, $\mathcal{AI} = 9$ and good nonlinearity.

[6] However, it is shown in [24] that Kasami functions have bad algebraic immunity when n is very large.

Table 1. Computation of the weight, degree, nonlinearity and algebraic immunity for the inverse function and some Kasami power functions for $12 \le n \le 20$

	Inverse function					Kasami power functions				
n	exp.	weight	deg.	nonlin.	$\mathcal{AI}$	exp.	weight	deg.	nonlin.	$\mathcal{AI}$
12	-1	2048	11	1984	5	993	2048	11	1984	5
13	-1	4096	12	4006	6	993	4096	6	$2^{12}-2^6$	6
14	-1	8192	13	8064	6	4033	8192	6	$2^{13}-2^7$	6
15	-1	2^{14}	14	16204	6	4033	2^{14}	7	$2^{14}-2^8$	7
16	-1	2^{15}	15	$2^{15}-2^8$	6	$2^{14}-2^7+1$	2^{15}	15	$2^{15}-2^7$	7
17	-1	2^{16}	16	65174	7	$2^{14}-2^7+1$	2^{16}	8	$2^{16}-2^8$	8
18	-1	2^{17}	17	$2^{17}-2^9$	7	$2^{16}-2^8+1$	2^{17}	9	$2^{17}-2^9$	8
19	-1	2^{18}	18	261420	7	$2^{16}-2^8+1$	2^{18}	9	$2^{18}-2^9$	9
20	-1	2^{19}	19	$2^{19}-2^{10}$	7	$2^{18}-2^9+1$	2^{19}	19	$2^{19}-2^9$	9

4 Efficient Computation of Immunity against Fast Algebraic Attacks

Let us first introduce some notation for this section. Any Boolean function f with an n-bit input vector $x := (x_1, \ldots, x_n)$ can be characterized by its truth table $T(f) := (f(0), \ldots, f(2^n - 1)) \in \mathbb{F}^{2^n}$ or by its algebraic normal form $f(x) = \bigoplus_\alpha f_\alpha x^\alpha$, with coefficients $f_\alpha \in \mathbb{F}$, multi-indices $\alpha \in \mathbb{F}^n$ (which can also be identified by their integers) and the abbreviation $x^\alpha := x_1^{\alpha_1} \cdots x_n^{\alpha_n}$. Consequently, we define the coefficient vector of f by $C(f) := (f_0, \ldots, f_{2^n-1}) \in \mathbb{F}^{2^n}$.

Given a Boolean function f with n input variables, the goal is to decide whether g of degree e and h of degree d exist, such that $fg = h$. The known function f is represented preferably by the truth table $T(f)$, which allows to efficiently access the required elements, and the unknown functions g and h are represented by the coefficient vectors $C(g)$ and $C(h)$, which leads to the simple side conditions $g_\beta = 0$ for $|\beta| > e$ and $h_\gamma = 0$ for $|\gamma| > d$. In order to decide if g and h exist, one has to set up a number of linear equations in g_β and h_γ. Such equations are obtained, e.g., by evaluation of $f(z) \cdot \bigoplus_\beta g_\beta z^\beta = \bigoplus_\gamma h_\gamma z^\gamma$ for some values of z. There are $D + E$ variables, so one requires at least the same number of equations. The resulting system of equations can be solved by Gaussian elimination with time complexity $\mathcal{O}((D+E)^3) = \mathcal{O}(D^3)$. If any $D+E$ equations are linearly independent, then no nontrivial g and h of corresponding degree exist. Otherwise, one may try to verify a nontrivial solution. Certainly, there are more sophisticated algorithms, namely we are able to express a single coefficient h_γ as a linear combination of coefficients g_β. If these relations hold for any value of γ, one may choose γ with $|\gamma| > d$ such that $h_\gamma = 0$, in order to obtain relations in g_β only. Consequently, equations for coefficients of g can be completely separated from equations for coefficients of h. As there are only E variables g_β, one requires at least E equations, and the system of equations can be solved in $\mathcal{O}(E^3)$. Depending on the parameters n, d, e and on the structure of f, there are different strategies how to efficiently set up equations.

4.1 Setting up Equations

In this section, we consider the product $fg = h$ where f, g and h are arbitrary Boolean functions in n variables. Here are some additional notational conventions: For $\alpha, \beta, \gamma \in \mathbb{F}^n$, let $\alpha \subseteq \beta$ be an abbreviation for $\mathrm{supp}(\alpha) \subseteq \mathrm{supp}(\beta)$, where $\mathrm{supp}(\alpha) := \{i | \alpha_i = 1\}$, and let $\alpha \vee \beta := (\alpha_1 \vee \beta_1, \ldots, \alpha_n \vee \beta_n)$. For $B, C \in \mathbb{F}^{2^n}$, we define the scalar product $B \cdot C := \bigoplus_{k=0}^{2^n-1} [B]_k \cdot [C]_k$. All expressions are modulo 2 here. With the following theorem, we are able to express a single coefficient h_γ as a linear combination of coefficients g_β, where the linear combination is computed either with $T(f)$ or with $C(f)$.

Theorem 1. *Let $f(x) = \bigoplus_\alpha f_\alpha x^\alpha$ and $g(x) = \bigoplus_\beta g_\beta x^\beta$. Set $h(x) = \bigoplus_\gamma h_\gamma x^\gamma := f(x) \cdot g(x)$. With $A_{i,j} \in \mathbb{F}$ and $B_{i,j} \in \mathbb{F}^{2^n}$, we have for each γ*

$$h_\gamma = \bigoplus_\beta \binom{\gamma}{\beta} A_{\gamma,\beta} \cdot g_\beta \tag{6}$$

$$A_{i,j} := B_{i,j} \cdot T(f) = B_{i,i-j} \cdot C(f) \tag{7}$$

$$[B_{i,j}]_k := \binom{i}{k} \cdot \binom{k}{j} \,. \tag{8}$$

Proof. The binary Moebius transform relates the ANF of a Boolean function with the corresponding truth table, namely considering Lucas' theorem $f(k) = \bigoplus_\alpha \binom{k}{\alpha} f_\alpha = \bigoplus_{\alpha \subseteq k} f_\alpha$ and $f_k = \bigoplus_\alpha \binom{k}{\alpha} f(\alpha) = \bigoplus_{\alpha \subseteq k} f(\alpha)$. We obtain the relation $h_\gamma = \bigoplus_{\alpha \subseteq \gamma} f(\alpha) g(\alpha)$. With $g(\alpha) = \bigoplus_{\beta \subseteq \alpha} g_\beta$, this becomes $h_\gamma = \bigoplus_{\alpha \subseteq \gamma} \bigoplus_{\beta \subseteq \alpha} g_\beta f(\alpha)$. Rearranging the coefficients, we finally have the product $h_\gamma = \bigoplus_{\beta \subseteq \gamma} g_\beta \bigoplus_{\beta \subseteq \alpha \subseteq \gamma} f(\alpha) = \bigoplus_\beta \binom{\gamma}{\beta} g_\beta B_{\gamma,\beta} \cdot T(f)$. In order to prove the second relation, we multiply the ANF of both functions and obtain $h_\gamma = \bigoplus_{\alpha \vee \beta = \gamma} f_\alpha g_\beta$. This binary sum can then be partitioned according to $h_\gamma = \bigoplus_{\beta \subseteq \gamma} g_\beta \bigoplus_{\alpha \subseteq \gamma; \alpha \vee \beta = \gamma} f_\alpha$. With Lucas' theorem again, we have the relation $h_\gamma = \bigoplus_{\beta \subseteq \gamma} g_\beta \bigoplus_{\gamma - \beta \subseteq \alpha \subseteq \gamma} f_\alpha = \bigoplus_\beta \binom{\gamma}{\beta} g_\beta B_{\gamma,\gamma-\beta} \cdot C(f)$. □

4.2 Determining the Existence of Solutions

We propose an efficient algorithm to determine the existence of g and h with corresponding degrees, see Alg. 2. The algorithm is based on the equation $h_\gamma = \bigoplus_{\beta \subseteq \gamma} g_\beta \bigoplus_{\beta \subseteq \alpha \subseteq \gamma} f(\alpha)$, which is a variant of Th. 1.

Let us discuss the complexity of Alg. 2. Initialization of G takes at most $\mathcal{O}(E^2)$ time and memory, and $\mathcal{I}$ can be constructed in $\mathcal{O}(E)$ time. Iteration initiates by choosing a fixed γ of weight $d+1$, this step will be repeated E times to set up the same number of equations. Notice that the set $\{\gamma : |\gamma| = d+1\}$ is sufficient to choose E different values of γ, as $E < \binom{n}{d+1}$ in the case of $e \ll d$ and $d \approx n/2$ (which is the typical scope of fast algebraic attacks). Thereafter, one chooses a fixed β of weight b. This step will be repeated for all $\binom{d+1}{b}$ elements of weight b, and for all $b = 0, \ldots, e$. Given this choice of γ and β, we find $|\mathcal{A}| = 2^{d+1-b}$, which corresponds to the number of operations to compute

Algorithm 2 Determine the existence of g and h for any f

Input: A Boolean function f with n input variables and two integers $0 \le e \le \mathcal{AI}(f)$ and $\mathcal{AI}(f) \le d \le n$.

Output: Determine if g of degree at most e and h of degree at most d exist such that $fg = h$.

Initialize an $E \times E$ matrix G, and let each entry be zero.
Compute an ordered set $\mathcal{I} \leftarrow \{\beta : |\beta| \le e\}$.
for i from 1 to E **do**
    Choose a random γ with $|\gamma| = d + 1$.
    Determine the set $\mathcal{B} \leftarrow \{\beta : \beta \subseteq \gamma,\ |\beta| \le e\}$.
    for all β in $\mathcal{B}$ **do**
        Determine the set $\mathcal{A} \leftarrow \{\alpha : \beta \subseteq \alpha \subseteq \gamma\}$.
        Compute $A \leftarrow \bigoplus_{\mathcal{A}} f(\alpha)$.
        Let the entry of G in row i and column β (in respect to $\mathcal{I}$) be 1 if $A = 1$.
    end for
end for
Solve the linear system of equations, and output `no` g `and` h `of corresponding degree` if there is only a trivial solution.

A. Overall complexity of the iteration becomes $E \sum_{b=0}^{e} \binom{d+1}{b} 2^{d+1-b} < E(e + 1)\binom{d+1}{e} 2^{d+1} < DE^2$, where the last inequality holds in the specified range of parameters. Time complexity of the final step of Alg. 2 is $\mathcal{O}(E^3)$. The dominating term, and hence complexity of Alg. 2 corresponds to $\mathcal{O}(DE^2)$. Compared to the complexity $\mathcal{O}(D^3)$ of Alg. 2 in [21], Alg. 2 is very efficient for g of low degree.

4.3 Experimental Results

In [15], a class of (non-symmetric) Boolean functions f with maximum algebraic immunity is presented; these functions will be referred here as DGM functions. Application of Alg. 2 on their examples for $n = 5, 6, 7, 8, 9, 10$ reveals that h and g exist with $d = \mathcal{AI}(f) = \lceil n/2 \rceil$ and $e = 1$. We point out that this is the most efficient situation for a fast algebraic attack. Explicit functions g with corresponding degree are also obtained by Alg. 2, see Tab. 2 (where dim denotes the dimension of the solution space for g of degree e). A formal expansion of $f(x) \cdot g(x)$ was performed to verify the results. A reaction on this attack is presented in [16].

5 Efficient Computation of Immunity for Symmetric Functions

Consider the case that $f(x)$ is a symmetric Boolean function. This means that $f(x) = f(x_1, \ldots, x_n)$ is invariant under changing the variables x_i. Therefore, we have $f(y) = f(y')$ if $|y| = |y'|$ and we can identify f with its (abbreviated) truth table $T^s(f) := (f^s(0), \ldots, f^s(n)) \in \mathbb{F}^{n+1}$ where $f^s(i) := f(y)$ for a y with $|y| = i$. Let $\sigma_i(x) := \bigoplus_{|\alpha|=i} x^\alpha$ denote the elementary symmetric polynomial of

Table 2. Degrees of the functions h and g for DGM functions f with n input variables

n	$\deg f$	$\deg h$	$\deg g$	g	dim
5	4	3	1	$1+x_4$	4
6	4	3	1	$1+x_6$	4
7	5	4	1	$1+x_4+x_5$	1
8	5	4	1	$1+x_5+x_6$	1
9	8	5	1	$x_4+x_5+x_6+x_7$	1
10	8	6	1	$x_5+x_6+x_7+x_8$	1

degree i. Then, each symmetric function f can be expressed by $f(x) = \bigoplus f_i^s \sigma_i(x)$ with $f_i^s \in \mathbb{F}$. Similarly to the non-symmetric case, f can be identified with its (abbreviated) coefficient vector $C^s(f) := (f_0^s, \dots, f_n^s) \in \mathbb{F}^{n+1}$.

In this section, we present a general analysis of the resulting system of equations for symmetric functions and propose a generic and a specific algorithm in order to determine the existence of g and h of low degrees.

5.1 Setting up Equations

One can derive a much simpler relation for the coefficients h_γ in the case of symmetric functions f.

Corollary 1. *Let $f(x) = \bigoplus_{i=0}^n f_i^s \sigma_i(x)$ be a symmetric function and $g(x) = \bigoplus_\beta g_\beta x^\beta$. Set $h(x) = \bigoplus_\gamma h_\gamma x^\gamma := f(x) \cdot g(x)$. Then, with $A_{i,j}^s \in \mathbb{F}$ and $B_{i,j}^s \in \mathbb{F}^{n+1}$, we have for each γ*

$$h_\gamma = \bigoplus_\beta \binom{\gamma}{\beta} A^s_{|\gamma|,|\beta|} \cdot g_\beta \tag{9}$$

$$A^s_{i,j} := B^s_{i,j} \cdot T^s(f) = B^s_{i,i-j} \cdot C^s(f) \tag{10}$$

$$\left[B^s_{i,j}\right]_k := \binom{i-j}{i-k} . \tag{11}$$

Proof. Notice that Th. 1 holds for any function f, including symmetric functions. Computation of $A_{\gamma,\beta} = B_{\gamma,\beta} \cdot T(f)$ for symmetric functions may be simplified by collecting all terms of the truth table with the same weight. Therefore, let $i := |\gamma|$ and $j := |\beta|$ and define $[B^s_{i,j}]_k := \bigoplus_{|\alpha|=k} [B_{\gamma,\beta}]_\alpha$, such that $A_{\gamma,\beta} = A^s_{i,j} := B^s_{i,j} \cdot T^s(f)$. For $j \le i$ we have $\bigoplus_{|\alpha|=k} \binom{\gamma}{\alpha}\binom{\alpha}{\beta} = \bigoplus_{|\alpha|=k; \beta \subseteq \alpha \subseteq \gamma} 1$. Counting the number of choices of the k elements of the support of α, we find that the above sum equals $\binom{i-j}{k-j}$. The proof of $A^s_{i,j} = B^s_{i,i-j} \cdot C^s(f)$ is similar. □

5.2 Determining the Existence of Solutions

Given a symmetric function f, the existence of g and h with corresponding degrees can be determined by an adapted version of Alg. 2 (which will be referred as Alg. 2^s): step 7 is omitted, and step 8 is replaced by $A \leftarrow A^s_{i,j}$. The discussion of this slightly modified algorithm is similar to Sect. 4.2. However, computation of $A^s_{i,j}$ requires only $n+1$ evaluations of the function f, which can be neglected in terms of complexity. Consequently, time complexity to set up equations is only about $\mathcal{O}(E^2)$, and overall complexity of Alg. 2^s becomes $\mathcal{O}(E^3)$.

Next, we will derive a method of very low (polynomial) complexity to determine the existence of g and h of low degree for a symmetric function f, but with the price that the method uses only sufficient conditions (i.e. some solutions may be lost). More precisely, we constrict ourselves to homogeneous functions g of degree e (i.e. g contains monomials of degree e only), and Eq. 9 becomes $h_\gamma = A^s_{|\gamma|,e} \bigoplus_{|\beta|=e} \binom{\gamma}{\beta} g_\beta$. Remember that $h_\gamma = 0$ for $|\gamma| > d$, so the homogeneous function g is determined by the corresponding system of equations for all γ with $|\gamma| = d+1, \ldots, n$. In this system, the coefficient $A^s_{|\gamma|,e}$ is constant for $\binom{n}{|\gamma|}$ equations. If $A^s_{|\gamma|,e} = 0$, then all these equations are linearly dependent (i.e. of type $0 = 0$). On the other hand, if $A^s_{|\gamma|,e} = 1$, then a number of $\binom{n}{|\gamma|}$ additional equations is possibly linearly independent. Consequently, if the sum of all possibly linearly independent equations for $|\gamma| = d+1, \ldots, n$ is smaller than the number of variables $\binom{n}{e}$, then nontrivial homogeneous functions g exist. This sufficient criterion is formalized by

$$\sum_{i=d+1}^{n} A^s_{i,e} \cdot \binom{n}{i} < \binom{n}{e} . \tag{12}$$

Given some degree e, the goal is to find the minimum value of d such that Eq. 12 holds. This can be done incrementally, starting from $d = n$. We formalized Alg. 3 of polynomial complexity $\mathcal{O}(n^3)$. This algorithm turned out to be very powerful (but not necessarily optimal) in practice, see Sect. 5.4 for some experimental results.

Algorithm 3 Determine the degrees of g and h for symmetric f

Input: A symmetric Boolean function f with n input variables.
Output: Degrees of specific homogeneous functions g and h such that $fg = h$.

for e from 0 to $\lceil n/2 \rceil$ **do**
  Let $d \leftarrow n$, number of equations $\leftarrow 0$, number of variables $\leftarrow \binom{n}{e}$.
  while number of equations $<$ number of variables **and** $d + 1 > 0$ **do**
    Compute $A \leftarrow A^s_{d,e}$.
    Add $A \cdot \binom{n}{d}$ to the number of equations.
    $d \leftarrow d - 1$.
  end while
  Output $\deg g = e$ and $\deg h = d + 1$.
end for

For a specified class of symmetric Boolean functions f, it is desirable to prove some general statements concerning the degrees of g and h for any number of input variables n. In the next section, we apply technique based on Alg. 3 in order to prove a theorem for the class of majority functions.

5.3 Fast Algebraic Attacks on the Majority Function

We denote by f the symmetric Boolean majority function with $n \geq 2$ input variables, defined by $f^s(i) := 0$ if $i \leq \lfloor n/2 \rfloor$ and $f^s(i) := 1$ otherwise. For example, $T^s(f) := (0, 0, 1)$ for $n = 2$, and $T^s(f) := (0, 0, 1, 1)$ for $n = 3$. The algebraic degree of this function is $2^{\lfloor \log_2 n \rfloor}$. In [7] and [17], it could be proven independently that f has maximum algebraic immunity[7]. However, in the following theorem, we disclose the properties of f (and related functions) with respect to fast algebraic attacks.

Theorem 2. *Let f be the majority function with any $n \geq 2$ input variables. Then there exist Boolean functions g and h such that $fg = h$, where $d := \deg h = \lfloor n/2 \rfloor + 1$ and $e := \deg g = d - 2^j$, and where $j \in \mathbb{N}^0$ is maximum so that $e > 0$.*

Proof. According to Eq. 9 for symmetric functions, we set up a system of equations in the coefficients of g only. The coefficients $A^s_{i,j}$ of Eq. 10 have a simple form in the case of the majority function, namely $A^s_{i,j} = \bigoplus_{k \geq d} \binom{i-j}{k-j} = \bigoplus_{k \geq d} \binom{i-j-1}{k-j-1} + \bigoplus_{k \geq d} \binom{i-j-1}{k-j} = \binom{i-j-1}{d-j-1} + 2 \bigoplus_{k \geq d} \binom{i-j-1}{k-j} = \binom{i-j-1}{d-j-1}$ for $i > d$. Additionally, we assume that g is homogeneous of degree $e := d - 2^j$ where j is chosen maximum such that $e \geq 1$. According to Lucas' theorem, we find $A^s_{d+i,e} = 0$ for $1 \leq i < d - e$. Consequently, only equations with $|\gamma| = 2d - e, \ldots, n$ may impose conditions on the coefficients g_β. As we can show that $\sum_{i=0}^{e-1} \binom{n}{i} < \binom{n}{e}$, the sufficient criterion (12) is satisfied, and nontrivial solutions exist. $\square$

Algebraic and fast algebraic attacks are invariant with regard to binary affine transformations in the input variables. Consequently, Th. 2 is valid for all Boolean functions which are derived from the majority function by means of affine transformations. We notice that such a class of functions was proposed in a recent paper, discussing design principles of stream ciphers [5, 6].

5.4 Experimental Results

Application of Alg. 2^s reveals that Th. 2 is optimal for the majority function where $d = \lfloor n/2 \rfloor + 1$ (verification for $n = 5, 6, \ldots, 16$). An explicit homogeneous function g can be constructed according to $g(x) = \prod_{i=1}^{e} (x_{2i-1} + x_{2i})$. We verified that Alg. 3 can discover the solutions of Th. 2.

In [7], a large pool of symmetric Boolean functions with maximum algebraic immunity is presented (defined for n even). One of these functions is the majority function, whereas the other functions are nonlinear transformations of

[7] Notice that for n odd, it is verified in [17] up to $n = 11$ that the majority function is the only symmetric Boolean function with maximum $\mathcal{AI}$.

the majority function. Application of Alg. 3 brings out that Th. 2 is valid for all functions f (verification for $n = 6, 8, \ldots, 16$). For some functions f, Alg. 3 finds better solutions than predicted by Th. 2 (e.g. for $T^s(f) := (0, 0, 0, 1, 1, 0, 1)$ where $d = 3$ and $e = 1$), which means that Th. 2 is not optimal for all symmetric functions. All solutions found by Alg. 3 can be constructed according to the above equation. Furthermore, Alg. 2^s finds a few solutions which are (possibly) better than predicted by Alg. 3 (e.g. for $T^s(f) := (0, 0, 0, 1, 1, 1, 0)$ where $d = 3$ and $e = 2$), which means that Alg. 3 is not optimal for all symmetric functions.

6 Conclusions

In this paper, several efficient algorithms have been derived for assessing resistance of LFSR-based stream ciphers against conventional as well as fast algebraic attacks. This resistance is directly linked to the Boolean output function used. In many recent proposals, the number of inputs for this function is about 20 or larger. For such input sizes, verification of immunity against (fast) algebraic attacks by existing algorithms is infeasible. Due to improved efficiency of our algorithms, provable resistance of these stream ciphers against conventional and fast algebraic attacks has become amenable. Our algorithms have been applied to various classes of Boolean functions. In one direction the algebraic immunity of two families of Boolean power functions, the inverse functions and Kasami type functions, have been determined. For the first time, the algebraic immunity $\mathcal{AI}$ of a highly nonlinear function with 20 variables is computed to be as large as $\mathcal{AI} = 9$. In another direction, our algorithms have been applied to demonstrate that large classes of Boolean functions with optimal algebraic immunity are very vulnerable to fast algebraic attacks. This applies in particular to classes of symmetric functions including the majority functions.

Acknowledgments

The first author has been supported by grant Kr 1521/7-2 of the DFG (German Research Foundation). The fourth and fifth author are supported in part by grant 5005-67322 of NCCR-MICS (a center of the Swiss National Science Foundation). The fifth author also receives partial funding through GEBERT RÜF STIFTUNG. We would like to thank Subhamoy Maitra for valuable discussions.

References

1. F. Armknecht, and G. Ars. Introducing a New Variant of Fast Algebraic Attacks and Minimizing Their Successive Data Complexity. In *Progress in Cryptology - Mycrypt 2005*, LNCS 3715, pages 16–32. Springer Verlag, 2005.
2. F. Armknecht. Algebraic Attacks and Annihilators. In *WEWoRC 2005*, volume P-74 of *LNI*, pages 13–21. Gesellschaft für Informatik, 2005.
3. F. Armknecht. Improving Fast Algebraic Attacks. In *Fast Software Encryption 2004*, LNCS 3017, pages 65–82. Springer Verlag, 2004.

4. G. Ars. Application des Bases de Gröbner à la Cryptographie. Thèse de l'Université de Rennes, 2005.
5. A. Braeken, and J. Lano. On the (Im)Possibility of Practical and Secure Nonlinear Filters and Combiners. In *Selected Areas in Cryptography - SAC 2005*, LNCS 3897, pages 159–174. Springer Verlag, 2006.
6. A. Braeken, J. Lano, N. Mentens, B. Preneel, and I. Verbauwhede. SFINKS: A Synchronous Stream Cipher for Restricted Hardware Environments. In *eSTREAM, ECRYPT Stream Cipher Project*, Report 2005/026. Available at `http://www.ecrypt.eu.org/stream`.
7. A. Braeken, and B. Preneel. On the Algebraic Immunity of Symmetric Boolean Functions. In *Progress in Cryptology - INDOCRYPT 2005*, LNCS 3797, pages 35–48. Springer Verlag, 2005.
8. P. Camion, C. Carlet, P. Charpin, and N. Sendrier. On Correlation-Immune Functions. In *Advances in Cryptology - CRYPTO 1991*, LNCS 576, pages 86–100. Springer Verlag, 1991.
9. A. Canteaut, and M. Videau. Symmetric Boolean Functions. In *IEEE Transactions on Information Theory*, volume 51/8, pages 2791–2811, 2005.
10. C. Carlet, and P. Gaborit. On the Construction of Boolean Functions with a Good Algebraic Immunity. In *Boolean Functions: Cryptography and Applications - BFCA*, 2005.
11. N. Courtois. Cryptanalysis of SFINKS. To appear in *Information Security and Cryptology - ICISC*, 2005.
12. N. Courtois, and W. Meier. Algebraic Attacks on Stream Ciphers with Linear Feedback. In *Advances in Cryptology - EUROCRYPT 2003*, LNCS 2656, pages 345–359. Springer Verlag, 2003.
13. N. Courtois. Fast Algebraic Attacks on Stream Ciphers with Linear Feedback. In *Advances in Cryptology - CRYPTO 2003*, LNCS 2729, pages 176–194. Springer Verlag, 2003.
14. N. Courtois, and J. Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In *Advances in Cryptology - ASIACRYPT 2002*, LNCS 2501, pages 267–287. Springer Verlag, 2002.
15. 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 2005*, LNCS 3557, pages 98–111. Springer Verlag, 2005.
16. D. K. Dalai, K. C. Gupta, and S. Maitra. Notion of Algebraic Immunity and its Evaluation related to Fast Algebraic Attacks. In *Second International Workshop on Boolean Function Cryptography and Applications*, 2006.
17. D. K. Dalai, S. Maitra, and S. Sarkar. Basic Theory in Construction of Boolean Functions with Maximum Possible Annihilator Immunity. To appear in *Design, Codes and Cryptography*. Springer Verlag, 2006.
18. N. J. Fine. Binomial Coefficients Modulo a Prime. In *The American Mathematical Monthly*, volume 54, pages 589–592, 1947.
19. J.-C. Faugère, and G. Ars. An Algebraic Cryptanalysis of Nonlinear Filter Generators using Gröbner bases. In *Rapport de Recherche INRIA*, volume 4739, 2003.
20. P. Hawkes, and G. G. Rose. Rewriting Variables: The Complexity of Fast Algebraic Attacks on Stream Ciphers. In *Advances in Cryptology - CRYPTO 2004*, LNCS 3152, pages 390–406. Springer Verlag, 2004.
21. W. Meier, E. Pasalic, and C. Carlet. Algebraic Attacks and Decomposition of Boolean Functions. In *Advances in Cryptology - EUROCRYPT 2004*, LNCS 3027, pages 474–491. Springer Verlag, 2004.

22. W. Meier, and O. Staffelbach. Nonlinearity Criteria for Cryptographic Functions. In *Advances in Cryptology - EUROCRYPT 1989*, LNCS 434, pages 549–562. Springer Verlag, 1990.
23. B. Mourrain, and O. Ruatta. Relations Between Roots and Coefficients, Interpolation and Application to System Solving. In *J. Symb. Comput.*, volume 33/5, pages 679–699, 2002.
24. Y. Nawaz, G. Gong, and K. Gupta. Upper Bounds on Algebraic Immunity of Power Functions. To appear in *Fast Software Encryption 2006*. Springer Verlag, 2006.
25. P. J. Olver. On Multivariate Interpolation. In *Stud. Appl. Math.*, volume 116, pages 201–240, 2006.
26. T. Siegenthaler. Correlation-Immunity of Nonlinear Combining Functions for Cryptographic Applications. In *IEEE Transactions on Information Theory*, volume 30/5, pages 776–780, 1984.
27. T. Siegenthaler. Decrypting a Class of Stream Ciphers Using Ciphertext Only. In *IEEE Transactions on Computer*, volume 34/1, pages 81–85, 1985.