Cryptogr. Commun. (2012) 4:25–45
DOI 10.1007/s12095-011-0057-z

A design of Boolean functions resistant to (fast) algebraic cryptanalysis with efficient implementation

Enes Pasalic

Received: 9 March 2011 / Accepted: 18 August 2011 / Published online: 6 September 2011

Abstract In this paper the possibilities of an iterative concatenation method towards construction of Boolean functions resistant to algebraic cryptanalysis are investigated. The notion of $\mathcal{AAR}$ (Algebraic Attack Resistant) function is introduced as a unified measure of protection against classical algebraic attacks as well as fast algebraic attacks. Then, it is shown that functions that posses the highest resistance to fast algebraic attacks are necessarily of maximum algebraic immunity, thus opposing a maximum resistance to algebraic cryptanalysis in general. The developed theoretical framework allows us to iteratively construct functions with maximum $\mathcal{AI}$, and of almost optimized resistance to fast algebraic cryptanalysis. This infinite class for the first time, apart from almost optimal resistance to algebraic cryptanalysis, in addition generates functions that allow an extremely efficient hardware implementation, possess high nonlinearity and maximum algebraic degree; thus unifying most of the relevant cryptographic criteria.

Keywords Boolean function · Fast algebraic attacks · Algebraic immunity · Annihilators · Algebraic attack resistant · High degree product

Mathematics Subject Classification (2010) 94A60

This is an extended and modified version of the manuscript presented at the 11th International Conference on Information Security and Crytpology, ICISC 2008, Seoul, Korea, 3–5 December, 2008. An extremely efficient hardware implementation is given in due details and more simulation results are added. For completeness, most of the theoretical results of the original paper are kept without some major modifications.

E. Pasalic (✉)
University of Primorska, FAMNIT and PINT, Glagoljaška 8, Koper, Slovenia
e-mail: enes.pasalic6@gmail.com

1 Introduction

The design of LFSR-based stream ciphers traditionally resides on the use highly nonlinear Boolean functions as filtering functions; the two major representatives being nonlinear filter generators and nonlinear combiners [22]. For instance, in the case of nonlinear filter generators n stages of a single Linear Feedback Shift Registers (LFSRs) (whose initial state consists of the secret key) are filtered by a nonlinear Boolean function $f : GF(2)^n \to GF(2)$ to provide the keystream sequence.

Apart from already established cryptographic criteria such as nonlinearity, algebraic degree, and resiliency, it turned out that the Boolean function must also have a certain order of algebraic immunity. This is due to recently introduced algebraic attacks based on the low degree annihilation of Boolean functions [9, 12]. These attacks reflect the property of certain cipher schemes for which the selection of function f of high algebraic degree that follows early ideas of Shannon's concept of confusion [25], and linear complexity attacks [22], is not a sufficient criterion any longer. Due to algebraic attacks, instead of setting up a system of equations of degree determined by the degree of function f, the attacker can consider a lower degree system if there either exists a low degree function g (called annihilator) such that $fg = 0$ or alternatively $(1 + f)g = 0$ [12, 20]. The minimum degree of nonzero annihilators g of either f or $1 + f$ is called *algebraic immunity* ($\mathcal{AI}$). Algebraic attacks currently present one of the most efficient cryptanalytic tool in stream cipher cryptanalysis; the applications include many prominent algorithms such as Bluetooth encryption algorithm E_0 analyzed in e.g. [11].

A few construction methods that generate functions reaching the upper bound on algebraic immunity $\lceil \frac{n}{2} \rceil$ (maximum $\mathcal{AI}$) functions) has recently been proposed [4, 6, 13, 14, 19]. However, all these methods do not succeed in optimization of other cryptographic criteria at the same time. Furthermore, though a high order of $\mathcal{AI}$ implies resistance to classical algebraic attacks this property is only a necessary but not sufficient criterion. The emergence of fast algebraic cryptanalysis [1, 10] still successfully invalidates any design for which there exists low degree function g such that $fg = h$ is of relatively low degree as well. For instance, fast algebraic attack was successfully applied to eSTREAM [15] proposal Sfinks [3], though the cipher was designed to withstand classical algebraic attacks. Denoting by e and d the degree of g and h respectively, the resistance to fast algebraic cryptanalysis is optimized if $e + d \geq n$ for any non-annihilating g and $e \in [1, \lceil \frac{n}{2} \rceil - 1]$.

Very recently a new design approach based on the modification of the so-called partial spread class of bent functions has been employed in e.g. [29–31]. Apart from this approach, several other design ideas (using minimal polynomials), have been recently proposed in e.g. [32, 33]. These classes of functions satisfy most of the cryptographic criteria, but still none of the methods ensure in a deterministic way either optimized or suboptimized resistance to fast algebraic attacks, including the construction method of Carlet and Feng [8]. Furthermore, an efficient hardware implementation of the new classes of cryptographically significant functions is not achievable.

This work is mainly motivated by the fact that for the time being the construction methods fail to provide functions unifying all the important cryptographic criteria. This is especially true when the fast algebraic attacks and the efficiency of implementation are taken into account. The reader should recall that the main advantage

of certain LFSR-based encryption schemes (e.g. filter generators) is their small hardware footprint, which was actually emphasized during the recent eSTREAM project [15]. While a hardware implementation of an LFSR is very efficient, the implementation of a Boolean functions with a moderate number of inputs may require an unacceptable amount of circuitry. A straightforward implementation using a table look-up or logical gates will require $O(2^n)$ memory cells (flip-flops for instance) for an n-variable function. This amount should be compared with hardware efficient designs such as stream ciphers TRIVIUM and GRAIN-128 that require approximately 2000–4000 logical gates.

The construction method proposed here, based on the concatenation of small initial functions proves to be very efficient in terms of hardware implementation. For instance, for a 20-variable function that uses 6-variable initial functions only ca. 900 flip-flops are required which is a tremendous improvement compared to the other methods. In general, the implementation cost only grows linearly (not exponentially) with the number of variables. In addition, the infinite class of functions proposed here optimizes almost all the cryptographic criteria, apart from achieving very high nonlinearity and offering a suboptimized resistance to fast algebraic attacks. This makes the functions in this class more vulnerable to (fast) correlation attacks (due to a lower nonlinearity) than other good classes. On the other hand, the class is characterized by an extremely efficient implementation, and there is a room for further improvements regarding its nonlinearity.

It should be noticed that the class of functions discussed in this manuscript is a class of balanced functions of maximum algebraic degree, and therefore these functions cannot be resilient (this is due to the Siegenthaler [26] trade-off which claims that the order of resiliency t satisfies $t \leq n - \deg(f) - 1$). Nevertheless, the resiliency criterion is in the first place important in the design of nonlinear combiners and nonlinear filter generators. For the former design approach a divide-and-conquer method of Siegenthaler [27] and the fast correlation attacks of Meier and Stafellbach [21] may be directly applied to a subset of the combining LFSRs (the cardinality of the subset is strictly larger than the resiliency order) by utilizing the statistical properties of the observed keystream sequence. In the case of nonlinear filter generators there is no possibility for applying a divide-and-conquer method directly. Thus, one first transforms a given filter generator scheme into an equivalent system (a nonlinear combiner generating the same running key sequence as the filter generator), and then such a system is attacked using the similar techniques as above. The technique of finding the equivalent system, using a Walsh orthogonal expansion of the state filter function, was suggested by Siegenthaler [28]. Later, the fast correlation algorithm of Meier and Stafellbach was also modified to be applicable to filter generators [16].

Nevertheless, it is not clear whether the process of finding an equivalent system is feasible for the typical lengths of LFSRs used in real-life applications. While Sigenthaler in [28] concludes that the maximum length of LFSR is ca. 50, there is no exact estimate for a maximum length of the LFSR for which modification of fast correlation attacks is applicable to filter generators, see [16]. The complexity of the algorithm in [16] seems to grow exponentially with the LFSR length, and only under certain statistical assumptions concerning the filtering function the experiments could be successfully carried out for an LFSR of length 100 (assuming further a sparse connection polynomial). We notice, that in a recent work [17] a detailed

analysis of correlation attacks and their applicability to various design schemes has been conducted. In brief, when the security of filter generators is of concern the authors deduce that a sufficient condition for protection against correlation attacks is a property of first order "quasi-immunity". For further details regarding this result, and for exact definition of the first order quasi-immunity (which is a concept closely related to first order resiliency) the reader is referred to [17].

Due to impossibility of attaining the maximum algebraic degree and certain resiliency order at the same time, we only focus on the degree optimized functions thus indirectly referring to the applications for which the above attacks cannot be applied in a straightforward manner (but also referring to the design of filter generators that use an LFSR of sufficient length).

To reach our goal of designing functions that oppose a maximum resistance to algebraic cryptanlysis, some theoretical results that relate the notions of algebraic immunity and resistance to fast algebraic attacks are first derived. A unified measure against both fast and classical cryptanalysis is introduced, the notion that we name $\mathcal{AAR}$ (algebraic attack resistance). The notion of $\mathcal{AAR}$ includes the maximum $\mathcal{AI}$ property per definition, but it is shown that another related concept *high degree product* actually includes the maximum $\mathcal{AI}$ property. This framework is then used in deriving the set of sufficient conditions for a certain recursive, concatenation-based construction to generate $\mathcal{AAR}$ functions. These conditions being extremely hard to satisfy, the optimum requirement $e + d \geq n$ is slightly relaxed ($e + d \geq n - 1$ is used instead) which then enables an iterative method for constructing functions satisfying all the relevant cryptographic criteria in the design of nonlinear filter generators.

The rest of the paper is organized as follows. Basic definitions and notations are introduced in Section 2. Section 3 gives a thorough treatment regarding the algebraic properties of the iterative construction technique based on the concatenation of four functions. Furthermore, the new notion of $\mathcal{AAR}$ functions is introduced, and the relationship between the optimal resistance to fast and to classical algebraic attacks is deduced. These results are then utilized in Section 4 for proposing an iterative method for generation of suboptimized $\mathcal{AAR}$ functions, with overall good cryptographic properties. The cryptographic properties are discussed in details, both from the security and implementation point of view. Some concluding remarks are given in Section 5.

2 Preliminaries

We denote the Galois field of order 2^n by $\mathbb{F}_{2^n}$ and the corresponding vector space by $\mathbb{F}_2^n$. For the rest of this paper, if otherwise not stated, x will denote a vector containing n input binary variables, that is $x = (x_1, \ldots, x_n) \in \mathbb{F}_2^n$. A Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is usually represented via so called *algebraic normal form* (ANF),

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

Then 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 $\lambda_u \neq 0$. 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,

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

The set of all Boolean functions in n variables is denoted by $\mathcal{B}_n$, and functions of degree at most one are called *affine* functions, whose associated set is denoted $\mathcal{A}_n$.

Definition 1 The *nonlinearity* of an n-variable function f is defined as,

$$\mathcal{N}_f = min_{g \in \mathcal{A}_n}(d_H(f, g)), \tag{2}$$

where d_H is the Hamming distance between two binary vectors, that is

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

The support set of function $f \in \mathcal{B}_n$, denoted by $supp(f)$, is the set of input values where f has a nonzero evaluation, that is,

$$supp(f) = \{x \in \mathbb{F}_2^n \mid f(x) = 1\}.$$

A function f is said to be balanced if it outputs equal number of zeros and ones, that is

$$\#\{x \in \mathbb{F}_2^n : f(x) = 1\} = \#\{x \in \mathbb{F}_2^n : f(x) = 0\}.$$

In the context of algebraic cryptanalysis related to Boolean functions, an annihilator of f is a non-zero function g such that $f(x)g(x) = 0$, which is most often written as $fg = 0$. The notion of algebraic immunity was introduced in [20] as a measure of resistance to algebraic attacks.

Definition 2 The algebraic immunity $AI(f)$ is the minimum value of d such that f or $f + 1$ admits an annihilating function of degree d.

In this context, for arbitrary $f \in \mathcal{B}_n$, the set of functions that annihilate f respectively $1 + f$ may be defined as,

$$An(f) = \{g \in \mathcal{B}_n, g \neq 0 \mid fg = 0\}; \quad An(1 + f) = \{g \in \mathcal{B}_n, g \neq 0 \mid (1 + f)g = 0\}.$$

The following notation is then useful,

$$deg(An(f)) = \min_{g \in An(f)} deg(g); \quad deg(An(1 + f)) = \min_{g \in An(1+f)} deg(g).$$

3 Theoretical framework towards resistance to algebraic attacks

A construction method based on the concatenation of functions from smaller variable space has been frequently used as an efficient tool in the design of cryptographically strong Boolean functions. Nevertheless, the known methods have failed so far in providing good functions resistant to both fast and classical algebraic cryptanalysis. These classes of functions are also attractive in terms of an efficient hardware

implementation. In this section we study the algebraic properties of an iterative concatenation method involving four subfunctions. In addition, a general relation that interlinks the optimum resistance to fast and classical algebraic cryptanalysis is derived.

3.1 Some properties of functions with maximum $\mathcal{AI}$

The purpose of this section is to identify some basic conditions that any function with maximum $\mathcal{AI}$ must satisfy with respect to its subfunctions. For the rest of this manuscript we focus on the representation of $f \in \mathcal{B}_{n+2}$ as a concatenation of four functions, that is, $f = f_1||f_2||f_3||f_4 \in \mathcal{B}_{n+2}$, where each $f_i \in \mathcal{B}_n$ has maximum $\mathcal{AI}$. Using the shortened notation (f_i denoting $f_i(x)$), the ANF of function f is given by:

$$f = x_{n+1}x_{n+2}\,(f_1 + f_2 + f_3 + f_4) + x_{n+1}\,(f_1 + f_2) + x_{n+2}\,(f_1 + f_3) + f_1. \tag{3}$$

A similar expression is then valid for any annihilator g of f,

$$g = x_{n+1}x_{n+2}(g_1 + g_2 + g_3 + g_4) + x_{n+1}(g_1 + g_2) + x_{n+2}(g_1 + g_3) + g_1, \tag{4}$$

where g_i is arbitrary annihilator of f_i (including the trivial annihilation $g_i = 0$). Let g_i denote any minimum degree nonzero annihilator of $f_i \in \mathcal{B}_n$. If $deg(g_i) = d$ then we also use,

$$g_i(x) = T_d(g_i(x)) + T_{d-1}(g_i(x)) + \cdots + T_0(g_i(x)),$$

where each $T_r(g_i)$, for $0 \le r \le d$, contains only degree r monomial terms. Then in connection to the representation of annihilator g of f given in (4), the following simple properties are deduced.[1]

Lemma 1 *Let $f = f_1||f_2||f_3||f_4$, where $f_i \in B_n$ are functions with maximum $\mathcal{AI}$. Then any nonzero annihilator g of f represented as in (4) satisfies the following :*

(i) *If any $g_i = 0$ then $deg(g) \ge \lceil \frac{n}{2} \rceil + 1$.*
(ii) *If any g_i is such that $deg(g_i) > \lceil \frac{n}{2} \rceil$ then $deg(g) \ge \lceil \frac{n}{2} \rceil + 1$.*
(iii) *If there exists g such that $deg(g) < \lceil \frac{n}{2} \rceil + 1$ then $deg(g_i) = \lceil \frac{n}{2} \rceil$ for all $i \in [1, 4]$ and furthermore,*

$$T_d(g_1) = T_d(g_2) = T_d(g_3) = T_d(g_4); \qquad T_{d-1}\left(\sum_{i=1}^{4} g_i\right) = 0. \tag{5}$$

Proof

(i) Without loss of generality assume $g_1 = 0$, then $x_{n+1}(g_1 + g_2)$ is of degree at least $\lceil \frac{n}{2} \rceil + 1$ unless $g_2 = 0$. But $g_1 = g_2 = 0$ implies that $g_3 = 0$ due to the term $x_{n+2}(g_1 + g_3)$, as otherwise $deg(g) \ge \lceil \frac{n}{2} \rceil + 1$.

[1] This result was independently derived in [2, Ch. 4]. The author of this article made this result available on Cryptology eprint archive in 2005, but the paper was soon withdrawn due to one erroneous result.

(ii) The similar idea is used here. Taking any g_i so that $deg(g_i) > \lceil \frac{n}{2} \rceil$, implies that $deg(g) \geq \lceil \frac{n}{2} \rceil + 1$.

(iii) Assuming $deg(g) < \lceil \frac{n}{2} \rceil + 1$ gives that,

$$T_d\left(\sum_{i=1}^{4} g_i\right) = 0 \quad T_{d-1}\left(\sum_{i=1}^{4} g_i\right) = 0 \quad T_d(g_1 + g_2) = 0 \quad T_d(g_1 + g_3) = 0,$$

and the result easily follows. □

The result of Lemma 1, in particular item (iii), is a useful tool for establishing the algebraic properties of given function. Showing that subfunctions $f_1, \ldots, f_4$ of maximum $\mathcal{AI}$ are chosen so that item (iii) above cannot be satisfied for neither f nor $1 + f$ is equivalent to proving $f = f_1||f_2||f_3||f_4$ has a maximum $\mathcal{AI}$. It has been used in [2], where the iterative method of designing the maximum $\mathcal{AI}$ functions was proposed. The construction requires three suitable input functions $f_1^0, f_2^0, f_3^0 \in \mathbb{F}_2^{n_0}$ to iteratively generate maximum $\mathcal{AI}$ functions on $f_1^i, f_2^i, f_3^i \in \mathbb{F}_2^{n_0+2i}$,

$$\begin{aligned} f_1^i &= f_1^{i-1}||f_2^{i-1}||1 + f_3^{i-1}||f_1^{i-1} \\ f_2^i &= f_2^{i-1}||1 + f_3^{i-1}||f_1^{i-1}||f_2^{i-1} \quad i \geq 1 \\ f_3^i &= 1 + f_3^{i-1}||f_1^{i-1}||f_2^{i-1}||f_3^{i-1}. \end{aligned} \tag{6}$$

This method generates the maximum $\mathcal{AI}$ functions with relatively good nonlinearity, at least the nonlinearity value is superior compared to the constructions in [4, 6, 7, 13, 14, 19], but in general slightly worse when compared to the method in [8]. A set of initial functions satisfying the conditions for the above recursion to generate maximum $\mathcal{AI}$ is e.g. $f_1^0 = x_1x_2 + x_3$, $f_2^0 = x_2x_3 + x_4$, $f_3^0 = x_2x_4 + x_1$ [2]. The main problem with this construction is its unknown susceptibility to fast algebraic attacks.

The conditions on the input functions may be significantly relaxed if we only allow one iteration step. This means that starting with four maximum $\mathcal{AI}$ functions on $\mathbb{F}_2^n$ (with an additional condition on one function) we may generate a maximum $\mathcal{AI}$ function on $\mathbb{F}_2^{n+2}$ using the following construction method.

Proposition 1 *Let $f = f_1||f_2||f_3||f_4$ be a function in $\mathcal{B}_{n+2}$, n even, whose subfunctions $f_i \in \mathcal{B}_n$ have maximum $\mathcal{AI}$, that is $\mathcal{AI}(f_i) = \lceil \frac{n}{2} \rceil$. Furthermore, let $f_3 = 1 + f_1$, and f_1 is such that for any function $\tilde{g}$ of $deg(\tilde{g}) = e$, $e \in [1, \lceil \frac{n}{2} \rceil - 1]$, we have $deg(f_1\tilde{g}) = d \geq \mathcal{AI}(f_1)$, and $e + d \geq n$. Then $\mathcal{AI}(f) = \lceil \frac{n}{2} \rceil + 1$, i.e. f has maximum $\mathcal{AI}$.*

Proof We have to show that any nonzero function g that annihilates either f or $(1 + f)$ is of degree $deg(g) \geq \lceil \frac{n}{2} \rceil + 1$. If $g_1, \ldots, g_4$ are annihilators of respectively $f_1, \ldots, f_4$ then $f_ig_i = 0$ for $i \in [1, 4]$, where $deg(g_i) \geq \lceil \frac{n}{2} \rceil$ or some of g_is are zero. Furthermore,

$$g = x_{n+1}x_{n+2}(g_1 + g_2 + g_3 + g_4) + x_{n+1}(g_1 + g_2) + x_{n+2}(g_1 + g_3) + g_1.$$

Assume on contrary that $deg(g) < \lceil \frac{n}{2} \rceil + 1$, implying also $deg(g_i) < \lceil \frac{n}{2} \rceil + 1$. Then obviously $deg(g_1 + g_3) < \lceil \frac{n}{2} \rceil$ due to the term $x_{n+2}(g_1 + g_3)$ in the ANF of g above. On the other hand

$$\left.\begin{aligned} f_1g_1 &= 0, \\ f_3g_3 = (1 + f_1)g_3 &= 0, \end{aligned}\right\} \implies f_1(g_1 + g_3) = g_3.$$

Since f_1 has maximum $\mathcal{AI}$, the condition $deg(f_1(g_1+g_3)) = d \geq \mathcal{AI}(f_1)$ and $e+d \geq n$, together with $deg(g_1+g_3) < \lceil \frac{n}{2} \rceil$, then implies $deg(g_3) = d \geq \lceil \frac{n}{2} \rceil + 1$. This contradicts the assumption that $deg(g) < \lceil \frac{n}{2} \rceil + 1$, unless $g_1 = g_3 = 0$. The case $g_1 = g_3 \neq 0$ gives $f_1 \cdot 0 = g_3$, a contradiction. Thus, if $deg(g) < \lceil \frac{n}{2} \rceil + 1$ we must necessarily have $g_1 = g_3 = 0$, which then implies $g_2 = 0$ due to the term $x_{n+1}(g_1 + g_2)$. Then similarly, to have $deg(g) < \lceil \frac{n}{2} \rceil + 1$ the degree of g_4 should be less than $\lceil \frac{n}{2} \rceil - 1$, violating the assumption that $AI(f_4) = \lceil \frac{n}{2} \rceil$. Hence, nonzero annihilators of f are of degree $\geq \lceil \frac{n}{2} \rceil + 1$.

For the reasons of symmetry the annihilators of the complement function $1+f$ are of the same degree as for f. The same reasoning as above applies to $1+f = 1+f_1||1+f_2||f_1||1+f_4$ as the position of f_1 and $1+f_1$ are just interchanged. Thus f has maximum $\mathcal{AI}$. □

The main problem here is that the construction idea above cannot be easily turned into an iterative design method. It is interesting to note that the condition on f_4 may be relaxed, i.e. it is sufficient to have $AI(f_4) = \lceil \frac{n}{2} \rceil - 1$. Also, the condition that f_1 satisfies the degree relation $deg(fg) = d$ for the above specified parameters e, d actually implies that $AI(f_1) = \lceil \frac{n}{2} \rceil$. This important result will be proved in the next section.

3.2 Functions resistant to (fast) algebraic attacks

It was already mentioned that functions optimizing the algebraic immunity does not protect from fast algebraic attacks in case there exists a degree e function g such that $fg = h$ is of degree d, and $e + d < n$. The efficiency of the fast algebraic attack depends on both parameters and finding a tuple (e, d) so that $e + d$ is substantially smaller than n will result in an overall lower attack complexity. A more elaborate description of how fast algebraic attacks work can be found in e.g. [1, 10]. For convenience, the main steps of algorithm and corresponding complexities are summarized here.

1. *Search for relations*. Finding the low-valued (e, d) equation[s] for f of type $fg = h$ (sometimes also denoted $zX^e + X^d$ [1]). The complexity is roughly $\binom{n}{d}$ and is negligible in comparison to other steps.
2. *Pre-computation step*. For a given LFSR of length L and known characteristic polynomial, a universal binary string α of length $D = \sum_{i=0}^{d} \binom{L}{i}$ can be computed in $D \log^2 D$ operations [1, 10, 18].
3. *Substitution step*. The original degree d equations are rewritten via substitution process to yield degree e equations. This step takes about $2ED \log^2 D$ operations [18], where $E = \sum_{i=0}^{e} \binom{L}{i}$.
4. *Solving step*. The degree e system of equations is solved by linearization; this requires E^ω operations, where ω is the complexity of solving linear system (usually $\omega = 3$ as a conservative estimate).

Assuming the existence of small e, the dominating step in terms of complexity is the so-called substitution step that takes about $2ED \log^2 D$ operations [18], where $D = \sum_{i=0}^{d} \binom{L}{i}$, $E = \sum_{i=0}^{e} \binom{L}{i}$. Therefore the fast algebraic attacks imply reduced computational complexity (compared to classical algebraic attacks) whenever there exists (e, d) tuple(s) such that $2ED \log^2 D < D_{\mathcal{AI}}^{\omega}$, where $D_{\mathcal{AI}} = \sum_{i=0}^{\mathcal{AI}(f)} \binom{L}{i}$.

In [10], it was proved that there always exists a tuple (e, d) (denoting the degree of functions g and h respectively) if e, d satisfy the bound $e + d \geq n$.

A straightforward relationship between the existence of (e, d)-relations and the degree of function f can be easily deduced [2, 10].

Theorem 1 [2, Ch.3] [10] *If f is of degree k then f satisfies a $(k, k + i)$-relation for any $i < k$.*

Proof For any functions f and g of degree k respectively i, $deg(fg) \leq k + i$. □

For a properly chosen algebraic immunity (to resist classical algebraic attacks), ensuring that (e, d) satisfy $e + d \geq n$ for any choice of e, d will imply protection against fast algebraic attacks partially due to the following result:

Lemma 2 [2, Ch.3] *For any functions $f, g \in \mathcal{B}_n$ such that $g \neq 0$ is not an annihilator of f we have $deg(fg) = d \geq \mathcal{AI}(f)$.*

The sufficiency of the condition $e + d \geq n$ comes from the complexity estimate of the substitution step given below.

Proposition 2 *The complexity of the substitution step in the fast algebraic attack is approximately the same (up to a logarithmic constant) for any choice of e, d satisfying $e + d = n$, $e \in [1, \mathcal{AI}(f) - 1]$, $d \in [\mathcal{AI}(f), n - 1]$.*

Proof For a given state size S, the complexity of substitution step is $2ED\log^2 D$, where $D = \sum_{i=0}^{d} \binom{S}{i}$, and $E = \sum_{i=0}^{e} \binom{L}{i}$. Neglecting the logarithmic term, and approximating $\sum_{i=0}^{u} \binom{S}{i} \approx S^u$ (for $u \ll S$) we have,

$$2ED\log^2 D \approx 2ED = 2S^e S^d = 2S^{e+d} = 2S^n.$$

□

Note that in case n is odd, for a function f of maximum $\mathcal{AI}$ there will always exist a tuple $(e, d) = (\lceil \frac{n}{2} \rceil - 1, \lceil \frac{n}{2} \rceil)$ and therefore the upper bound on the security for an LFSR based stream cipher application is estimated through the complexity of fast algebraic attacks with above (e, d). The goal is to ensure that (e, d) satisfies $e + d \geq n$ for any $1 \leq e \leq \mathcal{AI}(f) - 1$, and $d \geq \mathcal{AI}(f)$.

The constant behavior of $2ED\log^2 D$ is illustrated in the following practical context of usage. Assume that a nonlinear filtering generator uses an LFSR of length 160 and a filtering Boolean function $f : \mathbb{F}_2^{16} \to \mathbb{F}_2$. In case the generator is designed for 80 bits security and $e + d \geq 16$ then the complexity of the substitution step is given in Table 1. Thus, it seems to be well-motivated to introduce a new quantity

Table 1 Complexity of the substitution step for various (e, d); $L = 160$ and $n = 16$

(e, d)	(1,15)	(2,14)	(3,13)	(4,12)	(5,11)	(6,10)	(7,9)
$2ED\log^2 D$	2^{88}	2^{91}	2^{93}	2^{95}	2^{96}	$2^{96.7}$	2^{97}

that would measure the resistance of function to both algebraic and fast algebraic attacks.

Definition 3 Let f be a Boolean function on $\mathbb{F}_2^n$, with n of arbitrary parity. Then the function f is called algebraic attack resistant ($\mathcal{AAR}$) if f has a maximum $\mathcal{AI}$, that is $\mathcal{AI}(f) = \lceil \frac{n}{2} \rceil$, and furthermore for any non-annihilating function g of degree e, $1 \leq e \leq \lceil \frac{n}{2} \rceil - 1$, we necessarily have that $deg(fg) = d$ satisfies $e + d \geq n$. The latter property is referred to as $\mathcal{HDP}$ (High Degree Product) of order n, denoted $\mathcal{HDP}^{(n)}$.

The property of $\mathcal{HDP}^{(n)}$ is irrelevant to the complement operation.

Proposition 3 *If function $f \in \mathcal{B}_n$ satisfies the $\mathcal{HDP}$ property of order n so does the function $1 + f$.*

Proof By assumption for any non-annihilating function g of degree e and $h = fg$ of degree d, we have $e + d \geq n$. Then for any $deg(g) = e$ function g,

$$(1 + f)g = fg + g = h + g,$$

and consequently $deg((1 + f)g = deg(g + h)$. Since $deg(h) \geq n - e$, then for any $e \in [1, \lceil \frac{n}{2} \rceil - 1]$ we have $n - e > e$ and therefore $deg(g + h) = deg(h)$. □

The $\mathcal{AAR}$ property appears to be somewhat related to the concept of algebraic immunity through the result given by Lemma 2. Indeed, there is an explicit relationship connecting the notions of $\mathcal{AI}$ and $\mathcal{HDP}^{(n)}$ (the upper bound on the $e + d$). It turns out that the functions satisfying the $\mathcal{HDP}^{(n)}$ property automatically achieve the maximum $\mathcal{AI}$.

Theorem 2 *Let $f \in \mathcal{B}_n$. Assume that $fg = h$ satisfies $e + d \geq n$ for any choice of non-annihilating function g of degree e, and h of degree d, for $e \in [1, \mathcal{AI}(f)]$, and $d \in [\mathcal{AI}(f), n - 1]$. Then f has maximum $\mathcal{AI}$.*

Proof On contrary assume that $\mathcal{AI}(f) < \lceil \frac{n}{2} \rceil$, i.e. f has not maximum $\mathcal{AI}(f)$. When n is odd $\mathcal{AI}(f) = \frac{n+1}{2}$ if and only if $deg(An(f)) = \frac{n+1}{2}$, that is $deg(An(f)) = deg(An(1 + f)) = \frac{n+1}{2}$ [5] (for even n the relationship between the degrees of $An(f)$ and $An(1 + f)$ is an open problem). Let $\tilde{g} \in An(1 + f)$ such that $deg(\tilde{g}) < \lceil \frac{n}{2} \rceil$. Then,

$$(1 + f)\tilde{g} = 0 \Longrightarrow f\tilde{g} = \tilde{g}.$$

Since $deg(\tilde{g}) < \lceil \frac{n}{2} \rceil$ we have actually found (e, d) not satisfying $e + d \geq n$ (as $e = d = deg(\tilde{g}) < \lceil \frac{n}{2} \rceil$), contradicting the assumption on f.

For n even we consider two cases. If $\tilde{g} \in An(1 + f)$ such that $deg(\tilde{g}) < \lceil \frac{n}{2} \rceil$ then the proof is exactly the same as above. For $\tilde{g} \in An(f)$ such that $deg(\tilde{g}) < \lceil \frac{n}{2} \rceil$, by Proposition 3 function $(1 + f)$ satisfy the $\mathcal{HDP}$ property. That is, for any g, h of degree e and d respectively, $e + d \geq n$ in the product $(1 + f)g = h$. Then considering the product $(1 + f)\tilde{g} = \tilde{g}$, as $f\tilde{g} = 0$, would contradict the $\mathcal{HDP}$ property if $deg(\tilde{g}) < \lceil \frac{n}{2} \rceil$. □

This result states that $\mathcal{HDP}^{(n)} \Rightarrow \mathcal{AI}$, therefore the criteria for optimum $\mathcal{AI}$ and resistance to fast algebraic analysis is unified through the $\mathcal{HDP}$ property, that is $\mathcal{AAR} \Leftrightarrow \mathcal{HDP}^{(n)}$. Still, a simple attempt to generate $\mathcal{AAR}$ function in $n+1$ variables by relating the $\mathcal{AAR}$ subfunctions in n variables will fail as demonstrated by the example below.

Example 1 Let $f \in \mathcal{B}_n$ be an $\mathcal{AAR}$ function, for n odd. Since f is an $\mathcal{AAR}$ function then it is easy to show that $f' = f||1+f$ has maximum $\mathcal{AI}$. Now for any non-annihilating function g of fixed degree e we have to prove that $d \geq n+1-e$, where $deg(f'g) = d$. Note that $e \in [1, \frac{n+1}{2} - 1]$, as trivially there is a tuple $(e,d) = (\frac{n+1}{2}, \frac{n+1}{2})$, and we are interested in cases $e < d$, with $d \geq \mathcal{AI}(f') = \frac{n+1}{2}$. Any function $g \in \mathcal{B}_{n+1}$ can be written as,

$$g(x, x_{n+1}) = x_{n+1}(g_1(x) + g_2(x)) + g_1(x), \quad g_1, g_2 \in \mathcal{B}_n.$$

Then, $f'g = x_{n+1}[g_2 + f(g_1+g_2)] + fg_1$ and taking $g_1 = g_2$ gives $f'g = [x_{n+1} + f]g_1$ which only satisfies the relation $e + d \geq n$ but not $e + d \geq n+1$. Hence the function $f' = f||1+f$ is of maximum $\mathcal{AI}$ but not necessarily an $\mathcal{AAR}$ function.

4 An iterative design of almost optimal $\mathcal{AAR}$ functions

In general, when f and g are represented as a concatenation of four functions (cf. (3) and (4)), the product $fg \in \mathcal{B}_{n+2}$ can be after some simplification written as,

$$\begin{aligned} fg = x_{n+2}x_{n+1} \Bigg[& g_4 \sum_{j=1}^{4} f_j + (f_1+f_2+f_3) \sum_{j=1}^{4} g_j + (f_1+f_2)(g_1+g_3) \\ & + (f_1+f_3)(g_1+g_2) \Bigg] \\ & + x_{n+1}(f_1g_1 + f_2g_2) + x_{n+2}(f_1g_1 + f_3g_3) + f_1g_1. \end{aligned} \tag{7}$$

The only way to analyze the behavior of the above product is to put certain restrictions on the form of the subfunctions f_j. In order to simplify the above expression we select three distinct function on $\mathbb{F}_2^n$, denoting them f_1, f_2, f_4 and introduce the dependency on f_3, that is $f_3 = 1 + f_1$. Then, the derived class of functions on $\mathbb{F}_2^{n+2}$ is closely related to the construction given by (6).

Theorem 3 *Let $f = f_1||f_2||f_3||f_4$ be a function on $\mathbb{F}_2^{n+2}$, n even, whose subfunctions $f_i \in B_n$ satisfy the following:*

1. *$f_1, \ldots, f_4$ are $\mathcal{AAR}$ functions with $f_3 = 1 + f_1$*
2. *For any function $g = g_1||g_2||g_3||g_4$ of degree e, the functions f_2, f_4 satisfy $deg(g_3 + f_2g_2 + f_4g_4) \geq n - e$, where not both functions g_2, g_4 are zero and $e \in [1, \lceil \frac{n}{2} \rceil]$.*

Then, $f \in B_{n+2}$ is an $\mathcal{AAR}$ function.

Proof To prove the $\mathcal{AAR}$ property, by Theorem 2 we only need to show that f satisfies degree relation $e + d \geq n+2$.

Due to the $\mathcal{AAR}$ assumption on f_i, we have $deg(f_i g_j) \geq n - e$ for any degree e function g_j, $e \in [1, \lceil \frac{n}{2} \rceil]$. Using the relation $f_3 = 1 + f_1$ the product fg in (7) may be written as,

$$fg = x_{n+2}x_{n+1}[g_3 + f_4 g_4 + f_1(g_1 + g_3) + f_2 g_2] + x_{n+1}[f_1 g_1 + f_2 g_2] + x_{n+2}[g_3 + f_1(g_1 + g_3)] + f_1 g_1.$$

We want to show that any nonzero choice of function g of fixed degree e, $e \in [1, \lceil \frac{n}{2} \rceil]$, implies that $deg(fg) = d \geq n + 2 - e$. Recall that,

$$g = x_{n+1}x_{n+2}(g_1 + g_2 + g_3 + g_4) + x_{n+1}(g_1 + g_2) + x_{n+2}(g_1 + g_3) + g_1.$$

implying $deg(g_i) \leq e$. Consider the coefficient $g_3 + f_1(g_1 + g_3)$ of x_{n+2} in the product fg. Obviously, we must have $deg(g_1 + g_3) \leq e - 1$ as otherwise the degree of g is greater than e, due to the term $x_{n+2}(g_1 + g_3)$. The $\mathcal{AAR}$ condition on f_1 implies that $deg(f_1 g_a) \geq n - e$ for any nonzero degree e function g_a. By Lemma 2, $n - e \geq \lceil \frac{n}{2} \rceil$.

We now show that $deg(f_1(g_1 + g_3)) > deg(g_3)$ for any choice of g_1, g_3 such that $g_1 + g_3 \neq 0$. The condition $deg(g_1 + g_3) \leq e - 1$ implies $deg(f_1(g_1 + g_3)) \geq n - (e - 1) = n + 1 - e$, and therefore $deg(fg) \geq n - e + 2$. Since f_1 is an $\mathcal{AAR}$ function $n - e \geq \lceil \frac{n}{2} \rceil$. Then $n + 1 - e > \lceil \frac{n}{2} \rceil$ and consequently $deg(f_1(g_1 + g_3)) > deg(g_3)$, as $\deg(g_3) \leq \lceil \frac{n}{2} \rceil$. Hence, the degree of $g_3 + f_1(g_1 + g_3)$ is governed by $f_1(g_1 + g_3)$, and because $deg(f_1(g_1 + g_3)) \geq n + 1 - e$ the function fg is an $\mathcal{AAR}$ function unless $g_1 = g_3$.

The subcase $g_1 = g_3 = 0$ results in an $\mathcal{AAR}$ function f due to the following. The term $x_{n+1}(g_1 + g_2)$ in function g implies that $deg(g_2) \leq e - 1$ assuming $g_1 = 0$. Consequently $deg(f_2 g_2) \geq n + 1 - e$ and fg is of degree $\geq n + 2 - e$ due to $x_{n+1}[f_1 g_1 + f_2 g_2]$. Thus, $g_1 = g_3 = 0$ would imply $g_2 = 0$, implying restriction $deg(g_4) = e - 2$ (because g is of degree e), so that $deg(fg) \geq n + 4 - e$ due to the term $x_{n+2}x_{n+1}[g_3 + f_4 g_4 + f_1(g_1 + g_3) + f_2 g_2]$.

Hence if f is not an $\mathcal{AAR}$ function we must necessarily have $g_1 = g_3 \neq 0$, and we get somewhat simplified expressions for g and fg,

$$g = x_{n+1}x_{n+2}(g_2 + g_4) + x_{n+1}(g_1 + g_2) + g_1,$$

$$fg = x_{n+2}x_{n+1}[g_3 + f_4 g_4 + f_2 g_2] + x_{n+1}[f_1 g_1 + f_2 g_2] + x_{n+2}g_3 + f_1 g_1. \quad (8)$$

By assumption $deg(g_3 + f_2 g_2 + f_4 g_4) \geq n - e$, implying $fg \geq n + 2 - e$. □

Remark 1 The second condition in Theorem 3 may be slightly relaxed by requiring that $deg(f_2 g_2 + f_4 g_4) \geq n - e$. In this case the above result holds for any $e \in [1, \lceil \frac{n}{2} \rceil - 1]$ except for the case $e = \lceil \frac{n}{2} \rceil$, as there may exist $g_1, \ldots, g_4$, $deg(g_i) = \lceil \frac{n}{2} \rceil$ such that $deg(g_3 + f_2 g_2 + f_4 g_4) < n - e = \lceil \frac{n}{2} \rceil$. This would imply the possibility of finding tuple $(e, d) = (\lceil \frac{n}{2} \rceil, \lceil \frac{n}{2} \rceil + 1)$, thus violating $e + d \geq n + 2$.

4.1 Iterative construction of maximum $\mathcal{AI}$ functions with $e + d \geq n - 1$

To use the result of Theorem 3 recursively the conditions on initial functions turn out to be extremely hard to satisfy. Note first, that a similar set of constraints is obtained

after the replacement $f_1 \leftarrow f_2$, $f_2 \leftarrow 1 + f_2$, $f_3 \leftarrow f_1$, $f_4 \leftarrow f_3$, thus referring to the function f_2^i in (6). The product $f_2^i g$ can then be written as,

$$f_2^i g = x_{n+2}x_{n+1}[g_2 + f_1 g_3 + f_3 g_4 + f_2(g_1 + g_2)] + x_{n+1}[f_2(g_1 + g_2) + g_2] + x_{n+2}[f_2 g_1 + f_1 g_3] + f_2 g_1.$$

Similarly to the proof of Theorem 3, we get that the condition $g_1 = g_2 \neq 0$ from the term $x_{n+1}[f_2(g_1 + g_2) + g_2]$ then implies that $deg(g_2 + f_1 g_3 + f_3 g_4) \geq n - e$ for any choice of g_3 and g_4. Thus, a very similar condition as in Theorem 3 applies here, only different subfunctions being involved.

The main question now is what kind of conditions the set of initial functions must satisfy so that the $\mathcal{AAR}$ property is preserved in the recursion given by (6). In other words, assuming that the functions f_1^{i-1}, f_2^{i-1}, f_3^{i-1} satisfy particular set of conditions we would like to show that the $\mathcal{AAR}$ property holds also for f_1^i, f_2^i, f_3^i. One can show that these conditions for f_1^i, f_2^i, f_3^i become rather complicated involving the all three subfunctions and multiplicand g as well, yielding the degree constraint of the form,

$$deg\left[\sum_{j=1}^{4} a_j g_j^{i-1} + f_1^{i-1}\left(\sum_{j=1}^{4} b_j g_j^{i-1}\right) + f_2^{i-1}\left(\sum_{j=1}^{4} c_j g_j^{i-1}\right) + f_3^{i-1}\left(\sum_{j=1}^{4} d_j g_j\right)\right] \geq n - e, \tag{9}$$

for some binary coefficients $a_i, \ldots, d_i$.

However, the main obstacle in satisfying such initial conditions turns out to be the term $\sum_{j=1}^{4} a_j g_j^{i-1}$. As already indicated in Remark 1 the conditions are "slightly" relaxed if we allow a small deviation from optimality. That is, allowing the initial functions to satisfy $e + d \geq n - 1$ (instead of $e + d \geq n$) in the product $fg = h$, we may much easier find suitable initial functions to be used in a recursive manner. The condition that $e + d \geq n - 1$ implies that the degree of the expression $g_i + f_j g_k$ for functions $g_i, f_j, g_k \in \mathcal{B}_n$ is always dominated by $f_j g_k$. This is because we now only consider $e \in [1, \lceil \frac{n}{2} \rceil - 1]$ and regardless the parity of n we have,

$$deg(f_j g_k) \geq \mathcal{AI}(f_j) = \left\lceil \frac{n}{2} \right\rceil > e.$$

We notice that allowing the suboptimized case of degree relation $e + d \geq n - 1$, Theorem 2 is not applicable any longer and we are now forced to induce the optimality of $\mathcal{AI}$ through the construction. A class of function achieving maximum $\mathcal{AI}$ and satisfying the relation $e + d \geq n - 1$ is then called *suboptimized* $\mathcal{AAR}$ class. Therefore, we utilize the design ideas given in (6) but in a different manner. We slightly refine the construction to enable the usage of nonbalanced functions as initial functions too, though generating balanced functions in subsequent iterations. This modification will have a great impact on the cryptographic properties of the functions. In the first place the nonlinearity is improved, the functions are of maximum algebraic degree, optimized $\mathcal{AI}$ and they satisfy the $\mathcal{HDP}$ property of order $n - 1$.

The most suitable configuration of subfunctions that allows the use of nonbalanced initial functions seems to be the following one,

$$\begin{aligned} f_1^i &= f_1^{i-1}||f_2^{i-1}||1 + f_1^{i-1}||f_3^{i-1} \\ f_2^i &= f_2^{i-1}||1 + f_3^{i-1}||f_1^{i-1}||1 + f_2^{i-1} \\ f_3^i &= 1 + f_3^{i-1}||f_1^{i-1}||f_2^{i-1}||f_3^{i-1}. \end{aligned} \tag{10}$$

One may readily check that selecting $f_i^0 \in \mathcal{B}_n$ such that their Hamming weight equal to $wt(f_1^0) = wt(f_3^0) = 2^{n-1} - c$, and $wt(f_2^0) = 2^{n-1} + c$ will result in balanced functions f_j^i. It remains to show that a suitable selection of the input functions will initiate the recursion so that any f_j^i, $i \geq 0$ and $j = 1, 2, 3$, is a maximum $\mathcal{AI}$ satisfying $e + d \geq n - 1$.

Theorem 4 *Let $f_1^0, f_2^0, f_3^0 \in \mathcal{B}_n$ be maximum $\mathcal{AI}$ functions satisfying the set of conditions given in Lemma 1 (iii) with respect to the configuration in (10). In addition, let for any $g = g_1^0||g_2^0||g_3^0||g_4^0 \in \mathcal{B}_{n+2}$ of degree $e \in [1, \lceil \frac{n}{2} \rceil - 1]$ the following is satisfied,*

$$deg\left[f_1^0\left(\sum_{j=1}^{4} b_j g_j^0\right) + f_2^0\left(\sum_{j=1}^{4} c_j g_j^0\right) + f_3^0\left(\sum_{j=1}^{4} d_j g_j^0\right)\right] \geq n - e - 1; \quad b_j, c_j, d_j \in \mathbb{F}_2. \tag{11}$$

Then the function $f_j^i \in \mathcal{B}_{n+2i}$, $i \geq 0$ and $j = 1, 2, 3$, defined by (10), are maximum $\mathcal{AI}$ functions with almost optimized $\mathcal{HDP}$, that is satisfying $e + d \geq n + 2i - 1$ for $e \in [1, \lceil \frac{n}{2} \rceil + i - 1]$.

Proof The fact that f_j^i have maximum $\mathcal{AI}$ follows from hypothesis. The result concerning the $e + d$ relation is proved by induction. The case $i = 0$ follows directly from the assumption. Suppose the conditions are satisfied for all $k < i$. We show that the conditions hold for $k + 1$ as well. By assumption, the functions $f_1^k, f_2^k, f_3^k \in \mathcal{B}_{n+2k}$ are such that,

$$deg\left[f_1^{k-1}\left(\sum_{j=1}^{4} b_j g_j^{k-1}\right) + f_2^{k-1}\left(\sum_{j=1}^{4} c_j g_j^{k-1}\right) + f_3^{k-1}\left(\sum_{j=1}^{4} d_j g_j^{k-1}\right)\right] \geq n + 2k - e - 1, \tag{12}$$

where $f_j^{k-1}, g_j^{k-1} \in \mathcal{B}_{n+2k-2}$. W.l.o.g. we consider the function $f_1^{k+1} = f_1^k||f_2^k||1 + f_1^k||f_3^k$. Then for a degree e function $g^{k+1} \in \mathcal{B}_{n+2k+2}$ we need to show that,

$$deg(f_1^{k+1} g^{k+1}) \geq n + 2k - e + 1,$$

for any $e \in [1, \lceil \frac{n}{2} \rceil + k - 1]$. Consider the highest degree term in the product $f_1^{k+1} g^{k+1}$, i.e.,

$$x_{n+2k+1} x_{n+2k+2}\left[g_3^k + f_4^k g_4^k + f_1^k(g_1^k + g_3^k) + f_2^k g_2^k\right],$$

where we represent $g^{k+1} = g_1^k||g_2^k||g_3^k||g_4^k$. Now since $deg(g_3^k) \leq e < n + 2k - e - 1$ for any $e \in [1, \lceil \frac{n}{2} \rceil + k - 1]$, the degree of the terms in the brackets above is dominated by the sum $f_4^k g_4^k + f_1^k(g_1^k + g_3^k) + f_2^k g_2^k$. This sum when written in terms of the

subfunctions f_j^{k-1} and g_j^{k-1} gives the condition (12) which is satisfied by induction hypothesis. Thus,

$$deg\{x_{n+2k+1}x_{n+2k+2}[g_3^k + f_4^k g_4^k + f_1^k(g_1^k + g_3^k) + f_2^k g_2^k]\} \geq n + 2k - e + 1,$$

which proves the statement. □

4.2 Cryptographic properties of the construction

Through computer simulations (very non-exhaustive) we have found many sets of initial functions on $\mathbb{F}_2^4$ satisfying the conditions of Theorem 4, of which one instance is given below.[2]

$$f_1 = x_1 + x_1x_2 + x_3x_4 + x_1x_2x_3 + x_1x_2x_3x_4,$$
$$f_2 = x_2 + x_4 + x_1x_2 + x_2x_4 + x_3x_4 + x_1x_2x_3 + x_1x_3x_4 + x_2x_3x_4 + x_1x_2x_3x_4,$$
$$f_3 = x_2 + x_3 + x_1x_2 + x_2x_3 + x_3x_4 + x_1x_2x_3 + x_1x_2x_3x_4.$$

Remark 2 Though the conditions of Theorem 4 seem to be hard to satisfy, computer simulations indicate that a great majority of input functions over $\mathbb{F}_2^4$ does fulfil these conditions. Therefore a great variety of functions resistant to algebraic cryptanalysis may be found; the only remianing task being the nonlinearity optimization.

Algebraic degree of f_j^i Any f_j^i in the iteration is of optimized algebraic degree, i.e. $deg(f_j^i = n - 1$ for $f_j^i \in \mathcal{B}_n$. For instance, by inspecting the ANF of f_1^1 (using $f_3 = 1 + f_1$), the degree of f_1 is dominated by $x_{n+1}x_{n+2}(1 + f_2^0 + f_3^0)$ Also, the highest degree terms cannot be canceled out in further iterations which justifies the above statement.

Resistance to probabilistic algebraic attacks Probabilistic algebraic attacks, formally introduced as scenarios S4 and S6 in [12], are based on a low degree approximation of state equations (or filtering function), so that relatively simple equations that are true with probability close to 1 (preferably) are derived. This approach was successfully applied in cryptanalysis of Toyocrypt [9] due to a serious design flaw of Toyocrypt.[3] A low degree approximation to a filtering function constructed using the iterative method described above seems to be rather unrealistic. This is due to the fact that each iteration step essentially combine/add the monomials of two different functions, the sum being multiplied by new variables, cf. (3). Thus, it can be easily verified that the algebraic normal form of the resulting function will contain many high order terms, and therefore approximating these relations would require guessing many variables which in turn would reduce the probability that these relations hold.

[2] We have only performed a local search by selecting the three random functions and then manually modifying few bits in the truth tables of functions. An exhaustive search would imply checking around 2^{45} different choices for f_1^0, f_2^0, f_3^0.

[3] Notice that an application of classical algebraic attack on Toyocrypt yields even lower time complexity compared to probabilistic algebraic attacks.

Nonlinearity A rather loose lower bound on nonlinearity was derived in [2], the minimum value is estimated as

$$\mathcal{N}_f \geq n^{(f_1,1+f_1)} \cdot N_{f_1} + n^{(f_2,1+f_2)} \cdot N_{f_2} + n^{(f_3,1+f_3)} \cdot N_{f_3},$$

where $n^{(f_i,1+f_i)}$ denotes the number of times the tuple $(f_i, 1 + f_i)$ appears in the overall concatenation. Actually, we may derive a simple lower bound on the nonlinearity reasoning as follows. Let f_1^0, f_2^0, f_3^0 be the initial functions with the nonlinearities $\mathcal{N}_{f_1^0}, \mathcal{N}_{f_2^0}, \mathcal{N}_{f_3^0}$, and let furthermore $\mathcal{N}_{f^0} = \min_i \mathcal{N}_{f_i^0}$. In terms of the Walsh spectra the maximum absolut value in the spectra of any initial function satisfies $W_{f_i^0}(\alpha) \leq 2^{n_0-1} - \mathcal{N}_{f^0}$, for any $\alpha \in \mathbb{F}_2^{n_0}$. Consequently, for the functions f_1^i, f_2^i, $f_3^i \in \mathcal{B}_{n_0+2i}$, constructed in the i-th iteration step, the following is valid,

$$\mathcal{N}_{f_j^i} \geq 2^{n_0+2i-1} - \frac{1}{2}2^{2i}(2^{n_0-1} - \mathcal{N}_{f^0}) = 2^{n_0+2i-2} + 2^{2i-1}\mathcal{N}_{f^0}. \tag{13}$$

Remark 3 Starting with bent functions on $\mathbb{F}_2^4$ with nonlinearity $\mathcal{N}_{f_j^0} = 6$ the first two iteration steps give $\mathcal{N}_{f_j} \geq 2^6 + 8 \times 6 = 112$, which corresponds to an optimal value of nonlinearity $2^{n-1} - 2^{n/2}$ for $n = 8$. However, in the subsequent iterations the nonlinearity value will detorate from the optimal value. The lower bound, computed via (13), for $n = 10, 12$ is 448 and 1792 respectively (cf. Table 2)

A comparison in terms of relevant cryptographic criteria to the iterative construction methods in [7] and in [2] is given in Table 2. Here the functions ϕ_n and f^n are optimized $\mathcal{AI}$ functions obtained through the methods in [7] and [2] respectively, and f_*^n and corresponding bold face entries denotes our design. Obviously, both f^n and f_*^n are favorable to ϕ_n in terms of the nonlinearity, and degree. Nevertheless, our class is superior to f^n, providing functions *satisfying the $\mathcal{HDP}$ property of order $n-1$ thus providing a better resistance to fast algebraic attacks, having better nonlinearity and optimized algebraic degree*. The nonlinearity values related to our construction are obtained by running a computer program, whereas the algebraic properties (also confirmed by computer simulations) follows from Theorem 4 and above discussion on the algebraic degree.

The construction has a rather irregular behavior in terms of nonlinearity value. For instance, starting with another initial set of functions one may obtain the following sequence of nonlinearities 104/448/1888/7800/31948 resulting in lower values for small n but higher values for larger n. Further examples and improvements are provided in the next subsection.

Table 2 Comparison of relevant cryptographic criteria

Function	Degree	Nonlinearity	$\mathcal{AI}$	(e, d)
$\phi_8/f^8/f_*^8$	5/6/**7**	88/104/**104**	4 all	?/(2,4)/$e+d \geq 7$
$\phi_{10}/f^{10}/f_*^{10}$	8/8/**9**	372/452/**452**	5 all	?/(3,5)/$e+d \geq 9$
$\phi_{12}/f^{12}/f_*^{12}$	?/10/**11**	?(low)/1884/**1890**	6 all	?/(1,9)/$e+d \geq 11$
$\phi_{14}/f^{14}/f_*^{14}$	?/12/**13**	?(low)/7696/**7780**	7 all	?/(2,10)/$e+d \geq 13$
$\phi_{16}/f^{16}/f_*^{16}$	?/14/**15**	?(low)/31296/**31766**	8 all	?/(1,13)/$e+d \geq 15$

Remark 4 The results given above only consider the construction for even n. Though the same technique is applicable when n is odd, good initial functions seem to be harder to find then. The function space $\mathcal{B}_3$ is quite insufficient in this context, whereas selecting $f_j^0 \in \mathcal{B}_5$ gives on the other hand far too many possibilities. A suitable set, that generates highly nonlinear functions, is therefore to be found through sophisticated computer program.

Implementation In the view of a new version of algebraic attacks introduced in [23], whose running time is significantly lower than for the fast/classical algebraic attacks, an efficient implementation of filtering function seems to be of great importance. Since the running time of this attack is approximately $\mathcal{O}(D)$, where $D = \sum_{i=0}^{\deg(f)} \binom{L}{i}$ (L being as before the length of LFSR), the functions of more than 20 variables are required to guard against different modes of algebraic analysis (note however that the data complexity is much larger when compared to standard algebraic attacks). Then the implementation issue actually contradicts the fundamental ideas behind the design of filter generators, as these are designed for restricted hardware environments. This implies that the filtering function must have a sufficient algebraic structure for the ease of implementation, especially if the input space of the function is as large as some 20 variables.

The functions in the i-th iteration step of our construction given by (10) (this is of course true for the original construction of [2]) are the concatenation of 2^{2^i} initial functions $F = \{f_1^0, f_2^0, f_3^0, 1 + f_1^0, 1 + f_2^0, 1 + f_3^0\}$ is needed. Given the input value $x = (x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+2i})$ it can be shown that a simple loop of i iterations is required to compute the output value. For instance, evaluating the function $f_1^2 \in \mathcal{B}_8$, choosing the initial functions $f_1^0, f_2^0, f_3^0 \in \mathcal{B}_4$, for a given input $x = (x_1, \ldots, x_8) \in \mathbb{F}_2^8$ is done by computing the integer value of $(x_5, \ldots, x_8)$ and then evaluating the function enumerated by this number on the input $(x_1, \ldots, x_4)$ in the concatenated sequence,

$$f_1^2 = f_1^0 || f_2^0 || \bar{f}_1^0 || f_3^0 || f_2^0 || \bar{f}_3^0 || f_1^0 || \bar{f}_2^0 || \bar{f}_1^{\,0} || \bar{f}_2^0 || f_1^0 || \bar{f}_3^0 || \bar{f}_3^0 || f_1^0 || f_2^0 || f_3^0, \tag{14}$$

where $\bar{f}_i = 1 + f_i$. For simplicity, let $(x_5, x_6, x_7, x_8) = (0, 0, 0, 0)$. Then $(x_7, x_8) = (0, 0)$ indicates that the evalution is done among the first four functions in the concatenated sequence, and $(x_5, x_6) = (0, 0)$ finally would specify the first function f_1^0.

This approach, of iteratively identifying the initial function, is depicted in Fig. 1. We assume that the initial functions are k-variable functions, and due to the construction $n - k$ is even. For a given k-tuple of inputs $(x_1, \ldots, x_k)$ the circuit performs a look-up operation and evaluates the output for the three initial functions f_1^0, f_2^0 and f_3^0, along with their complemented values. At the same time, the logic circuit processes $n - k$ variables $(x_{k+1}, \ldots, x_n)$ in $r = (n - k)/2$ steps, starting with the most significant tuple (x_{n-1}, x_n).

To implement efficiently the logic circuit we need some kind of coding for identifying the initial function used (or its complement). Let us assign 3 bits to encode the initial functions as follows:

$$\begin{array}{rrr} f_1^i \leftarrow 000 & f_2^i \leftarrow 001 & f_3^i \leftarrow 010 \\ 1 + f_1^i \leftarrow 111 & 1 + f_2^i \leftarrow 110 & 1 + f_3^i \leftarrow 101 \end{array}$$

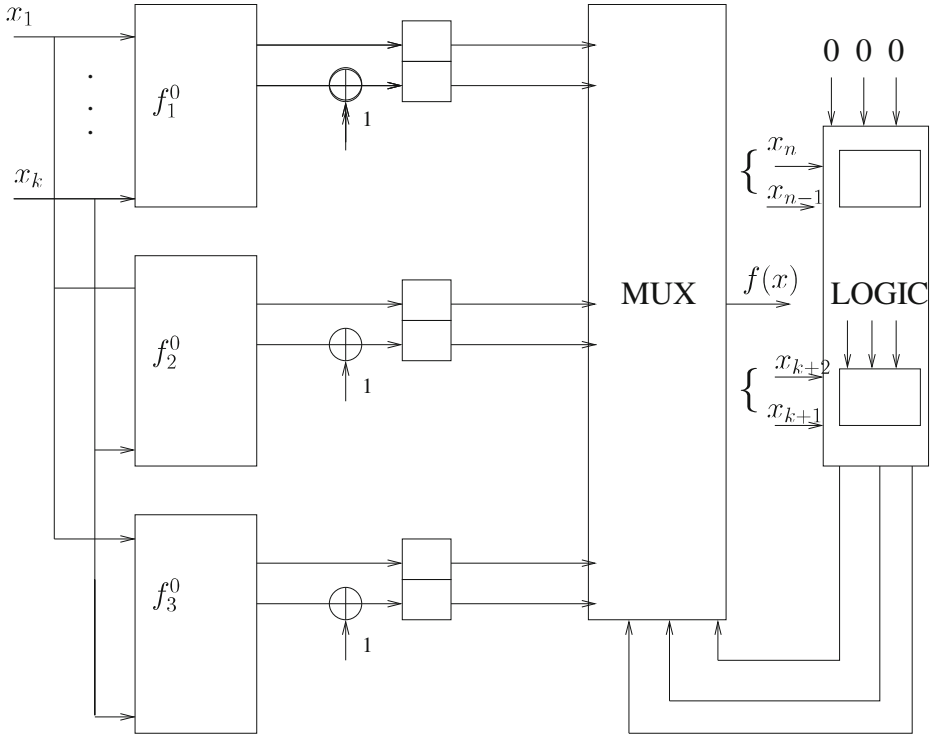

Fig. 1 High-scaled implementation circuit

Notice that the first bit of such an encoding will provide us with the information whether a particular function or its complement is identified. The $r = (n-k)/2$ boxes that take two input bits together with the three bits from the previous iteration are fixed with respect to the particular placement of subfunctions (cf. the configuration given by (10). Let us assume that the function f_1^r is used as a combining function. In the very first iteration the initial values are set to zero, and therefore only the input bits (x_{n-1}, x_n) determine the choice of initial function. For instance, the input $(x_{n-1}, x_n) = (0, 1)$ must give $(1, 1, 1)$ as the output, so that the function $1 + f_1^{r-1}$ is identified. The triple $(1, 1, 1)$ is now passed to the next iteration that processes the input variables (x_{n-3}, x_{n-2}). Depending on the input values the next subfunction is identified but inside the concatenation rule valid for the previous function. That is,

$$1 + f_1^{r-1} = 1 + f_1^{r-2} || 1 + f_2^{r-2} || f_1^{r-2} || 1 + f_3^{r-2},$$

and if for instance $(x_{n-3}, x_{n-2}) = (1, 1)$ the subfunction is $1 + f_3^{r-2}$. Therefore the corresponding output would be 101. The computation continues to the last round that finally outputs 3 bits used as the selector bits in the multiplexor.

A total hardware complexity is estimated as follows. To store the 3 look-up tables that implement the initial functions, 3×2^k memory cells (e.g. flip-flops) are needed. Apart from a six-input multiplexor, we additionally need $3 \times 32 \times (n-k)/2$ many flip-flops to implement the logic circuit. This is justified by viewing the computation in each round as a realization of a 5-input 3-output binary function. Hence, if these

blocks are also implemented as look-up tables (though there might be more efficient implementations) we get the following estimate,

$$\text{Mem. cost } = 1 \times 6\text{-input MUX } + (3 \times 2^k + 3 \times 32 \times (n-k)/2) \text{ flip-flops.}$$

For instance, to implement a 20-variable function using the 6-variable initial functions only one multiplexor and $3 \times 2^6 + 3 \times 2^5 \times 7 = 27 \times 32 = 864$ flip-flops are required.

In this context it is of importance to compare the concatenation method to the recently proposed approach by Carlet and Feng [8]. In their approach the function f is specified by its support set which is taken as $supp(f) = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{2^{n-1}-2}\}$, α being a primitive element in $\mathbb{F}_{2^n}$. Then the ANF of f is expected to contain a large number of terms, most likely this number is around 2^{n-1} as for a randomly generated function. This implies that an efficient hardware implementation is extremely impractical for the values of n needed in real-life applications, for $n > 20$. Nevertheless, our method compares favourably to most of the known design methods. One of a few exceptions is a Maiorana-McFarland class of functions which can be efficiently implemented as described in [24]. Furthermore, our method is efficiently implemented in software as well, as there is no need to store gigabytes of data to represent the truth table of the function.

4.3 Finding good initial functions

We have already noticed that the lower bounds on the nonlinearity given in Section 4.2 are rather lose, especially in case the gap between k and n is large (many iterations are used). A straightforward method of extending our ideas is to use initial functions from a larger variable space. In this way, the nonlinearity of the iterated functions may increase, but on the other hand there are two main problems with such an approach. In the first place, increasing the size of initial functions results in a larger complexity of implementation (it grows exponentially with the number of variables of initial functions). Secondly, the conditions in Theorem 4 are harder to check, if not infeasible for relatively large k. Nevertheless, for moderate sizes of the input space of initial functions the iteration may yield functions with better nonlinearity, and hopefully with same algebraic properties.

In Table 3 we give the nonlinearity and algebraic properties for some selections of initial functions with different input sizes. The algebraic immunity of these functions is not specified since in all the cases it attains the maximum value $n/2$. The initial functions seem to satisfy the same set of conditions in Theorem 4, though the condition given there is further relaxed so that the functions satisfy the relation $e + d \geq n - 2$ instead of $e + d \geq n - 1$. We have only conducted a restricted local

Table 3 Nonlinearity of iterated functions for initial functions from $\mathcal{B}_4$ and $\mathcal{B}_8$

Function $k = 4/k = 8$	Degree	Nonlinearity	$c : e + d \geq c$
$f^8_{(4)}/f^8_{(8)}$	7/7	104/116	7/6
$f^{10}_{(4)}/f^{10}_{(8)}$	9/9	452/478	9/8
$f^{12}_{(4)}/f^{12}_{(8)}$	11/11	1890/1918	11/10
$f^{14}_{(4)}/f^{14}_{(8)}$	13/13	7780/7898	13/12
$f^{16}_{(4)}/f^{16}_{(8)}$	15/15	31766/31878	15/14

search for good initial functions, thus the above results are likely to be improved further.

5 Conclusion

This paper proposes an iterative construction method for designing almost fully optimized Boolean functions satisfying most of the cryptographic criteria. The construction is very efficient from the implementation point of view making it attractive even when the input space exceeds some 30 variables. It remains to find some optimal set of initial functions (especially for odd n) to further increase the nonlinearity, thus making the functions (ciphers) more resistant to fast correlation and distinguishing attacks.

References

1. Armknecht, F.: Improving Fast Algebraic Attacks. LNCS, vol. 3017, pp. 65–82. Springer (2004)
2. Braeken, A.: Cryptographic properties of Boolean functions and S-boxes. Ph.D. thesis, Katholieke Universiteit Leuven, Belgium (2006)
3. Braeken, A., Lano, J., Mentens, N., Preneel, B., Verbauwhede, I.: Sfinks specification and source code. Avaliable on ECRYPT Stream Cipher Project page, http://www.ecrypt.eu.org/stream/sfinks.html (2005)
4. Braeken, A., Preneel, B.: On the algebraic immunity of symmetric Boolean functions. In: Advances in Cryptology—INDOCRYPT 2005. LNCS, vol. 3797, pp. 35–48. Springer (2005)
5. Canteaut, A.: Invited talk: open problems related to algebraic attacks stream ciphers. In: Proceedings of the 2005 International Workshop on Coding and Cryptography (WCC 2005), Bergen, Norway. LNCS, vol. 3969, pp. 120–134. Springer (2005)
6. Carlet, C.: Improving the algebraic immunity of resilient and nonlinear functions and constructing bent functions. Cryptology ePrint Archive, Report 2004/276, http://eprint.iacr.org/ (2004)
7. Carlet, C., Dalai, K.D., Gupta, C.K., Maitra, S.: Algebraic immunity for cryptographically significant Boolean functions: analysis and construction. IEEE Trans. Inf. Theory **IT-52**(7), 3105–3121 (2006)
8. Carlet, C., Feng, K.: An infinite class of balanced functions with optimal algebraic immunity, good immunity to fast algebraic attacks and good nonlinearity. In: Advances in Cryptology—ASIACRYPT 2008. LNCS, vol. 5350, pp. 425–440. Springer (2008)
9. Courtois, N.: Higher order correlation attacks, XL algorithm and cryptoanalysis of toyocrypt. In: International Conference on Information Security and Cryptology – ICISC 2002. LNCS, vol. 2587, pp. 182–199. Springer (2002)
10. Courtois, N.: Fast algebraic attacks on stream ciphers with linear feedback. In: Advances in Cryptology—CRYPTO 2003. LNCS, vol. 2729, pp. 176–194. Springer (2003)
11. Courtois, N.: Algebraic attacks on combiner with memory and several outputs. In: International Conference on Information Security and Cryptology – ICISC 2004. LNCS, vol. 3506, pp. 3–20. Springer (2005)
12. Courtois, N., Meier, W.: Algebraic attacks on stream ciphers with linear feedback. In: Advances in Cryptology—EUROCRYPT 2003. LNCS, vol. 2656, pp. 346–359. Springer (2003)
13. Dalai, D.K., Gupta, K.C., Maitra, S.: Cryptographically significant Boolean functions: construction and analysis in terms of algebraic immunity. In: Fast Software Encryption 2005. LNCS, vol. 3557, pp. 98–111. Springer (2005)
14. Dalai, D.K., Maitra, S., Sarkar, S.: Basic theory in construction of Boolean functions with maximum annihilator immunity. Des. Codes Cryptogr. **40**(1), 41–58 (2006)
15. ECRYPT: Call for stream cipher primitives. http://www.ecrypt.eu.org/stream/
16. Forré, R.: Advances in Cryptology—EUROCRYPT'89, LNCS, vol. 434, pp. 586–595 (1990)
17. Gouget, A., Sibert, H.: Revisiting correlation-immunity in filter generators. In: Selected Areas in Cryptography 2007. LNCS, vol. 4876, pp. 378–395 (2008)

18. Hawkes, P., Rose, G.G.: Rewriting variables: the complexity of fast algebraic attacks on stream ciphers. In: Advances in Cryptology—CRYPTO 2004. LNCS, vol. 3152, pp. 390–406. Springer (2004)
19. Li, N., Qi, W.F.: Construction and analysis of Boolean functions of $2t + 1$ variables with maximum algebraic immunity. In: Advances in Cryptology—ASIACRYPT 2006, LNCS, vol. 4284, pp. 84–98. Springer (2006)
20. Meier, W., Pasalic, E., Carlet, C.: Algebraic attacks and decomposition of Boolean functions. In: Advances in Cryptology—EUROCRYPT 2004. LNCS, vol. 3027, pp. 474–491. Springer (2004)
21. Meier, W., Staffelbach, O.: Fast correlation attacks on certain stream ciphers. J. Cryptol. **1**, 159–176 (1989)
22. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
23. Ronjom, S., Helleseth, T.: A new attack on the filter generator. IEEE Trans. Inf. Theory **IT-53**(5), 1752–1758 (2007)
24. Sarkar, P., Maitra, S.: Efficient implementation of cryptographically useful large Boolean functions. IEEE Trans. Comput. **52**(4), 410–417 (2003)
25. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27**, 379–423 (Part I); 623–656 (Part II) (1948)
26. Siegenthaler, T.: Correlation-immunity of nonlinear combining functions for cryptographic applications. IEEE Trans. Inf. Theory **IT-30**, 776–780 (1984)
27. Siegenthaler, T.: Decrypting a class of stream ciphers using ciphertext only. IEEE Trans. Comput. **C-34**, 81–85 (1985)
28. Siegenthaler, T.: Cryptanalysts representation of nonlinearly filtered M-sequences. In: Advances in Cryptology—EUROCRYPT'85. LNCS, vol. 219, pp. 103–110. Springer (1986)
29. Tang, X., Tang, D., Zeng, X., Hu, L.: Balanced Boolean functions with (almost) optimal algebraic immunity and very high nonlinearity. Cryptology ePrint Archive, Report 2010/443. http://eprint.iacr.org/ (2010)
30. Tu, Z., Deng, Y.: A class of 1-resilient function with high nonlinearity and algebraic immunity. Cryptology ePrint Archive, Report 2010/179 (2010)
31. Tu, Z., Deng, Y.: A conjecture on binary strings and its application on constructing Boolean functions of optimal algebraic immunity. Designs, Codes and Cryptography, Online First Articles (2010). doi:10.1007/s10623-010-9413-9
32. Wang, Q., Peng, J., Kan, H., Xue, X.: Constructions of cryptographically signicant Boolean functions using primitive polynomials. IEEE Trans. Inf. Theory **IT-56**, 3048–3053 (2010)
33. Zeng, X., Carlet, C., Shan, J., Hu, L.: Balanced Boolean functions with optimum algebraic immunity and high nonlinearity. Cryptology ePrint Archive, Report 2010/534. http://eprint.iacr.org/ (2010)