Evaluating the Resistance of Stream Ciphers with Linear Feedback Against Fast Algebraic Attacks

An Braeken, Joseph Lano, and Bart Preneel

Katholieke Universiteit Leuven
Dept. Elect. Eng.-ESAT/SCD-COSIC,
Kasteelpark Arenberg 10, 3001 Heverlee, Belgium
{an.braeken,joseph.lano,bart.preneel}@esat.kuleuven.be

Abstract. In this paper we evaluate the resistance of stream ciphers with linear feedback against fast algebraic attacks. We summarize the current knowledge about fast algebraic attacks, develop new and more efficient algorithms to evaluate the resistance against fast algebraic attacks, study theoretical bounds on the attacks, and apply our methodology to the eSTREAM candidates SFINKS and WG as an illustration.

1 Introduction

Algebraic attacks (AAs) and fast algebraic attacks (FAAs), proposed in 2002 and 2003 respectively, are a very interesting and powerful development in the cryptanalysis of stream ciphers. So far, most papers on this subject (see e.g. [1, 5, 3, 6]) present ways to improve the algorithms to perform these attacks. Another interesting problem in this context is how to *evaluate* the resistance of stream ciphers against these attacks. Meier et al. [7] discuss how to evaluate the resistance of stream ciphers with linear feedback and a Boolean output function against AA. Therefore they define the property of *algebraic immunity* of a Boolean function.

In this paper, we will discuss how the resistance of such ciphers against *fast* algebraic attacks can be evaluated. We will approach this problem both from a theoretical and a practical perspective. As FAAs are very often the most efficient attacks against such designs, the evaluation of the resistance against FAAs is a very important tool when analyzing a stream cipher.

The outline of this paper is as follows. We first introduce some preliminaries in Sect. 2. In Sect. 3, we give an explicit description of the current state of the art in (fast) algebraic attacks. In Sect. 4, a new algorithm to evaluate the resistance of a Boolean function against a class of FAAs is presented. Section 5 studies some theoretical bounds on the resistance of Boolean functions against FAAs. We discuss the resistance of eSTREAM candidates SFINKS and WG against FAAs in Sect. 6 as an illustration and conclude in Sect. 7. Due to page restrictions, we could not include all results in this proceedings version. An extended version containing more results and a comparison with recent similar results will be made available.

2 Preliminaries

The framework we develop applies to stream ciphers with a linear state update mechanism and a Boolean output function. The k -bit internal state is loaded with an initial state $\bar{s}_0 = (s_0, s_1, \dots, s_{k-1})$, updated by a linear function L at each time t , i.e., $\bar{s}_t = L^t(\bar{s}_0)$ where $L^t = L \circ \dots \circ L$. The Boolean output function f takes φ bits from the internal state and produces one key stream bit z_t .

We now introduce briefly some concepts on Boolean functions. A *Boolean function* f is a mapping from $\mathbb{F}_2^\varphi$ into $\mathbb{F}_2$. The *support* of f is defined as $\text{sup}(f) = \{\bar{x} \in \mathbb{F}_2^\varphi : f(\bar{x}) = 1\}$. The cardinality of $\text{sup}(f)$ represents the *weight* $\text{wt}(f)$ of the function. A Boolean function f can be uniquely represented by means of its *algebraic normal form (ANF)*:

$$f(\bar{x}) = f(x_0, \dots, x_{\varphi-1}) = \bigoplus_{(a_0, \dots, a_{\varphi-1}) \in \mathbb{F}_2^\varphi} c_f(a_0, \dots, a_{\varphi-1}) x_0^{a_0} \dots x_{\varphi-1}^{a_{\varphi-1}},$$

with $c_f(\bar{a}) = \bigoplus_{\bar{x} \preceq \bar{a}} f(\bar{x})$,

(1)

where $\bar{x} \preceq \bar{a}$ means that $x_i \leq a_i$ for all $i \in \{0, \dots, n-1\}$. The *algebraic degree* of f , denoted by d_f , is defined as the highest number of variables in the terms $x_0^{a_0} \dots x_{\varphi-1}^{a_{\varphi-1}}$ in the ANF of f .

Two functions f_1, f_2 on $\mathbb{F}_2^\varphi$ are said to be *affine equivalent* if

$$f_1(\bar{x}) = f_2(\bar{x}A \oplus \bar{b}) \oplus \bar{x} \cdot \bar{c} \oplus d, \quad (2)$$

where A is an invertible $\varphi \times \varphi$ matrix over $\mathbb{F}_2$, $\bar{b}, \bar{c} \in \mathbb{F}_2^\varphi$, $d \in \mathbb{F}_2$. Two functions are said to be *affine equivalent in the input variables* if and only if $\bar{c} = 0, d = 0$ in the above equation.

The lowest degree of a function g from $\mathbb{F}_2^\varphi$ into $\mathbb{F}_2$ for which $f(\bar{x}) \cdot g(\bar{x}) = 0$ for all $\bar{x} \in \mathbb{F}_2^\varphi$ or $(f(\bar{x}) \oplus 1) \cdot g(\bar{x}) = 0$ for all $\bar{x} \in \mathbb{F}_2^\varphi$ is called the *algebraic immunity (AI)* [7] of the function f . A function g is said to be an *annihilator* of f if $f(\bar{x}) \cdot g(\bar{x}) = 0$ for all $\bar{x} \in \mathbb{F}_2^\varphi$. It has been shown [5] that any function f with φ inputs has algebraic immunity at most $\lceil \frac{\varphi}{2} \rceil$.

We now introduce a similar notion that will be of relevance for FAAs. A Boolean function f on $\mathbb{F}_2^\varphi$ *satisfies a (d_g, d_h) -relation* (shortly $(d_g, d_h) - \mathcal{R}$) if a tuple of Boolean functions (g, h) of degree (d_g, d_h) exists such that $f(\bar{x}) \cdot g(\bar{x}) = h(\bar{x})$ for all $\bar{x} \in \mathbb{F}_2^\varphi$. This notion will only be relevant here when $d_g < d_h$.

In the rest of this paper, we denote the number of combinations with cardinality less than or equal to d in a set of u elements by $\mathcal{M}_d^u = \sum_{i=0}^d \binom{u}{i}$. Note that this number also represents the number of linearly independent functions (e.g. monomials) of degree less than or equal to d in $\mathbb{F}_2^u$.

3 Algebraic Attacks and Fast Algebraic Attacks

In this section we will describe AAs and FAAs, as developed in [5, 3] by Courtois and Meier. AAs are very basic and straightforward attacks. We write out the system of nonlinear equations between the initial state $\bar{s}_0$ and the key stream $(z_t)_{t \geq 0}$ and try to solve this. These attacks turn out to be very powerful. The

system of equations we are trying to solve consists of equations $f(L^i(\bar{s}_0)) = z_i$. Clearly, this original system of equations has k unknowns (corresponding with the k bits of the initial state) and degree d_f .

3.1 Algebraic Attacks

In a standard AA we will try to reduce the degree of the above system of equations before we solve it. This will lower the attack complexity considerably.

First, we need to perform a *relation search step*. Here we search for nonzero functions g and/or h of lower degree d_g and/or d_h both smaller than d_f , such that the following holds for all $\bar{x} \in \mathbb{F}_2^{\mathcal{C}}$:

$$\begin{cases} f(\bar{x}) \cdot g(\bar{x}) = 0 \\ (f(\bar{x}) \oplus 1) \cdot h(\bar{x}) = 0. \end{cases} \quad (3)$$

It is easy to see that we can then replace each equation in by a lower degree equation as follows:

$$\begin{aligned} g(L^i(\bar{s}_0)) &= 0, \forall i \text{ where } z_i = 1 \\ h(L^i(\bar{s}_0)) &= 0, \forall i \text{ where } z_i = 0. \end{aligned} \quad (4)$$

This new system of equations can be solved much faster because of its lower degree. To facilitate the complexity analysis, we use linearization (replacing each monomial by a new variable) followed by Gaussian elimination of this overdetermined system of equations. Consequently, the time complexity of solving the system of equations (4) is of order $(\mathcal{M}_{d_g}^k)^\omega$, with $\omega = \log_2(7) \approx 2.807$ (Strassen's exponent [9]). The data complexity is about $\mathcal{M}_{d_g}^k$ but can be lowered if multiple functions g and/or h exist.

The complexity of the AA clearly decreases rapidly with the degree of the multiples of f and $f \oplus 1$. Hence the efficiency of an AA depends on the lowest degree of any multiple. This corresponds to the algebraic immunity of the Boolean function, as defined above. So in essence to determine the resistance of a design against the AA, all that needs to be done is to make sure that the AI of its output function is sufficiently high. We will explain how this can be computed in Sect. 4.

3.2 Fast Algebraic Attacks

FAAs can be much more efficient than the standard AAs. In the FAA introduced by Courtois [3], the attacker tries to decrease the degree d of the system of equations even further by searching for relations between the initial state and several bits of the output function simultaneously, *i.e.*, equations of the form

$$F(\bar{s}_0, z_t, z_{t+1}, \dots, z_{t+T}) = 0, \forall t, \quad (5)$$

with $F(\bar{s}_0, \bar{z}) = g(\bar{s}_0, \bar{z}) \oplus h(\bar{s}_0)$ and $d_g < d_h$. See [6] for a general description. Let us now restrict to the case $T = 1$ in (5) and thus obtain the equations

$$z_i g(L^i(\bar{k})) \oplus h(L^i(\bar{k})) = 0, i \geq 0, \quad (6)$$

where (g, h) represents the tuple of functions with low degrees (d_g, d_h) and $d_g < d_h$ for which $f \cdot g = h$.

So far, FAAs mounted on actual stream ciphers with linear update mechanism all use equations of type (6). These are the FAAs we are going to consider here. It is an open problem whether better attacks of type (5) can be found for actual stream ciphers.

The considered attack can be described by the following simple framework, as briefly described by Courtois in [3] and more explicitly in [4].

1. *Relation search step* : We need to find $(d_g, d_h) - \mathcal{R}$ for our Boolean function. As defined above, f satisfies $(d_g, d_h) - \mathcal{R}$ if an equation exists of the following form, where g and h are nonzero and $d_g < d_h$:

$$f(\bar{x}) \cdot g(\bar{x}) = h(\bar{x}) \quad \forall \bar{x} \in \mathbb{F}_2^\varphi. \quad (7)$$

Multiplying the original equations by g , our system of equations now becomes of degree d_h :

$$\begin{cases} h(\bar{s}_0) &= z_0 \cdot g(\bar{s}_0) \\ h(L(\bar{s}_0)) &= z_1 \cdot g(L(\bar{s}_0)) \\ h(L^2(\bar{s}_0)) &= z_2 \cdot g(L^2(\bar{s}_0)) \\ \dots \end{cases} \quad (8)$$

2. *Precomputation step* : We now reduce the degree of (8) further from d_h to d_g in a precomputation step. Because the combination of linear recurring sequences is again a linear recurring sequence, there *always* exists a single linear combination of every $\mathcal{M}_{d_h}^k + 1$ consecutive bits such that for every $t \geq 0$:

$$\bigoplus_{i=0}^{\mathcal{M}_{d_h}^k} \alpha_i \cdot h(L^{i+t}(\bar{s}_0)) = 0, \quad (9)$$

and therefore our system of equations (8) becomes of degree d_g and has the following form:

$$\begin{cases} \bigoplus_{i=0}^{\mathcal{M}_{d_h}^k} \alpha_i \cdot z_i \cdot g(L^i(\bar{s}_0)) &= 0 \\ \bigoplus_{i=0}^{\mathcal{M}_{d_h}^k} \alpha_i \cdot z_{i+1} \cdot g(L^{i+1}(\bar{s}_0)) &= 0 \\ \bigoplus_{i=0}^{\mathcal{M}_{d_h}^k} \alpha_i \cdot z_{i+2} \cdot g(L^{i+2}(\bar{s}_0)) &= 0 \\ \dots \end{cases} \quad (10)$$

The complexity of this precomputation step has been shown to be $\mathcal{M}_{d_h}^k \cdot (\log_2 \mathcal{M}_{d_h}^k)^2$ (see Hawkes and Rose, [6]).

3. *Substitution step* : As we obtain an actual key stream and use it in a FAA, we need to substitute this key stream into (10). The best algorithm to do this so far has been described in [6] and has complexity $2 \cdot \mathcal{M}_{d_g}^k \cdot \mathcal{M}_{d_h}^k \cdot \log_2 \mathcal{M}_{d_h}^k$.
4. *Solving step* : We solve the system (10) by linearization. The time complexity to do so is of order $(\mathcal{M}_{d_g}^k)^\omega$. The data needed is $\mathcal{M}_{d_g}^k + \mathcal{M}_{d_h}^k \approx \mathcal{M}_{d_h}^k$. This data complexity can be reduced by a factor if we find several linear combinations, but then the substitution complexity will increase by the same factor.

The following observation is very important : to evaluate the resistance of a stream cipher with linear feedback and Boolean output function against this particular FAA, all we need to do is to perform step 1 (the relation search) for its Boolean function. Namely, we need to find out which $(d_g, d_h) - \mathcal{R}$ the Boolean function satisfies. In Sect. 4, we will show a new algorithm which performs this step much faster than previously thought possible, and which hence makes it possible to evaluate the resistance of stream cipher with linear feedback using relatively large Boolean functions against such a FAA (see Sect. 6 for some examples).

4 Algorithms for Computing AI and $(d_g, d_h) - \mathcal{R}$

As we explained in Sect. 3, to evaluate the resistance of a generator against the considered classes of (fast) AAs, all that needs to be done is performing the corresponding relation search step for its Boolean function. We now describe the algorithms to do so. In Sect. 4.1 we describe the algorithms from [7] to find relations for mounting AAs, but explain them in a more elegant way. This will help to understand the new algorithm we will develop in Sect. 4.2, and why this algorithm is much more efficient than previously thought possible.

4.1 Computing the AI of a Boolean Function

The aim is to find or to disprove the existence of a function $g(\bar{x})$ of degree d_g such that $f(\bar{x}) \cdot g(\bar{x}) = 0$ for all inputs $\bar{x}$. The unknowns we need to find are the coefficients $c_g(\bar{a})$ with $wt(\bar{a}) \leq d_g$ in the ANF of g :

$$g(\bar{x}) = \bigoplus_{\substack{\bar{a} \in \mathbb{F}_2^\varphi \\ wt(\bar{a}) \leq d_g}} c_g(\bar{a}) x_0^{a_0} \dots x_{\varphi-1}^{a_{\varphi-1}}. \quad (11)$$

Hence the number of unknown coefficients is $\mathcal{M}_{d_g}^\varphi$. The equations we can use to find these unknowns are found as follows: the requirement $f(\bar{x}) \cdot g(\bar{x}) = 0$ can be easily seen to correspond to the requirement that $g(\bar{x}) = 0$ whenever $f(\bar{x}) = 1$. As a good cryptographic Boolean function should be balanced, we have exactly $2^{\varphi-1}$ such equations. As we are looking for $d_g < \lceil \frac{\varphi}{2} \rceil$, the number of equations $2^{\varphi-1}$ is always larger than the number of unknowns $\mathcal{M}_{d_g}^\varphi$.

Solving the above system of equations corresponds to algorithm 1 in [7]. The complexity of solving this system naively is of order $2^{\varphi-1} \cdot (\mathcal{M}_{d_g}^\varphi)^2$. However, we can also just solve a subsystem of $\mathcal{M}_{d_g}^\varphi$ equations and check whether its solution fits the other equations. Then we get a complexity in the order of $(\mathcal{M}_{d_g}^\varphi)^3$.

Algorithm 2 of [7] improves this complexity. In this algorithm, a specific choice is made for the subsystem of $\mathcal{M}_{d_g}^\varphi$ equations that we will solve. Denote the system we are solving by

$$M \cdot \bar{c}_g = \bar{0}, \quad (12)$$

where $\bar{c}_g$ is the $\mathcal{M}_{d_g}^\varphi \times 1$ column matrix containing the ANF coefficients $c_g(\bar{a})$ ordered lexicographically on $\bar{a}$, and M is a $\mathcal{M}_{d_g}^\varphi \times \mathcal{M}_{d_g}^\varphi$ matrix of which we choose the rows as follows: the first row corresponds with the equation $g(\bar{x}) = 0$ with $wt(\bar{x}) = 0$ (if it exists, i.e., if $f(\bar{0}) = 1$). The following rows correspond to the equations (if they exist) where the Hamming weight of the input vector is 1, 2, ..., until Hamming weight d_g , all ordered lexicographically. By now we have about $\frac{1}{2}\mathcal{M}_{d_g}^\varphi$ rows of our matrix M . We fill up the remaining rows of M by choosing other values $\bar{x}$ at random.

Now we look at the upper part of this matrix, corresponding to the rows up to Hamming weight d_g . It is easy to see that this part of the matrix is in *echelon form* and its size is of order $\frac{1}{2}\mathcal{M}_{d_g}^\varphi \times \mathcal{M}_{d_g}^\varphi$. From the structure of the problem presented, it is easy to see that this echelon form has order of $\frac{1}{2}\mathcal{M}_{d_g}^\varphi$ pivot columns, $\frac{1}{2}\binom{\varphi}{d_g}$ zero columns (which we can ignore) and $\frac{1}{2}\mathcal{M}_{d_g-1}^\varphi$ nonzero nonpivot columns. The solution of this underdetermined system of equations can be found efficiently, with a complexity of order $\frac{1}{4}\mathcal{M}_{d_g}^\varphi \cdot (\mathcal{M}_{d_g}^\varphi + \mathcal{M}_{d_g-1}^\varphi)$ operations.

We now substitute these solutions into the remaining $\frac{1}{2}\mathcal{M}_{d_g}^\varphi$ equations. Only the nonzero pivot columns need to be substituted into the system, resulting in a substitution complexity of order $\frac{1}{8}(\mathcal{M}_{d_g}^\varphi)^2 \cdot \mathcal{M}_{d_g-1}^\varphi$, and we obtain a system of $\frac{1}{2}\mathcal{M}_{d_g}^\varphi$ equations in $\frac{1}{2}\mathcal{M}_{d_g}^\varphi$ unknowns, which can be solved by Gaussian elimination in order of $\frac{1}{8}(\mathcal{M}_{d_g}^\varphi)^3$ operations.

This approach corresponds to algorithm 2 in [7]. The last step will be the dominating step in terms of time complexity. To make the algorithm memory efficient we will process the matrix row by row and only store the nonzero nonpivot elements for later use, resulting in a memory complexity of order $\frac{1}{4}\mathcal{M}_{d_g}^\varphi \mathcal{M}_{d_g-1}^\varphi$.

4.2 Computing the $(d_g, d_h) - \mathcal{R}$ for a Boolean Function

The gain of algorithm 2 compared to algorithm 1 for computing the AI is only modest because the final system we need to solve still has half the number of unknowns as the original system. We will now show that for the search of $(d_g, d_h) - \mathcal{R}$, we can do much better.

Our aim is to find out whether the Boolean function satisfies $(d_g, d_h) - \mathcal{R}$ for certain values $d_g < d_h$ at which the complexity of a FAA would be less than exhaustive search. The unknowns we need to find are the coefficients $c_g(\bar{a})$ with $wt(\bar{a}) \leq d_g$ and $c_h(\bar{a})$ with $wt(\bar{a}) \leq d_h$ in the ANFs of g and h :

$$\begin{aligned} g(\bar{x}) &= \bigoplus_{\substack{\bar{a} \in \mathbb{F}_2^\varphi \\ wt(\bar{a}) \leq d_g}} c_g(\bar{a}) x_0^{a_0} \dots x_{\varphi-1}^{a_{\varphi-1}} \\ h(\bar{x}) &= \bigoplus_{\substack{\bar{a} \in \mathbb{F}_2^\varphi \\ wt(\bar{a}) \leq d_h}} c_h(\bar{a}) x_0^{a_0} \dots x_{\varphi-1}^{a_{\varphi-1}} . \end{aligned} \tag{13}$$

Hence the number of unknown coefficients is $\mathcal{M}_{d_h}^\varphi + \mathcal{M}_{d_g}^\varphi$.

Previously, it was thought that the complexity of solving this system of equations is of order $(\mathcal{M}_{d_h}^\varphi)^3$, making it hard to evaluate the resistance against FAAs

even for functions with modest input dimension. We will now show a new algorithm that can solve the problem in time complexity of order $\mathcal{M}_{d_h}^\varphi \cdot (\mathcal{M}_{d_g}^\varphi)^2 + (\mathcal{M}_{d_h}^\varphi)^2$. This makes evaluation of the resistance against FAAs achievable for larger Boolean functions, as shown by the examples in Sect. 6.

The equations used to find the unknown coefficients of the ANFs of g and h are as follows: the requirement $f(\bar{x}) \cdot g(\bar{x}) = h(\bar{x})$ for all $\bar{x} \in \mathbb{F}_2^\varphi$ gives us a condition on the ANF coefficients of g and h for *every* value of $\bar{x}$. This results in an equation that can be of one of the following two types:

$$\begin{aligned} g(\bar{x}) &= h(\bar{x}) && \text{whenever } f(\bar{x}) = 1 \\ h(\bar{x}) &= 0 && \text{whenever } f(\bar{x}) = 0. \end{aligned} \quad (14)$$

The number of equations obtained, 2^φ , will always be larger than the number of unknowns $\mathcal{M}_{d_h}^\varphi + \mathcal{M}_{d_g}^\varphi$ in situations of practical interest.

The naive solution is again to solve this system of equations, which has complexity of order $2^\varphi \cdot (\mathcal{M}_{d_h}^\varphi + \mathcal{M}_{d_g}^\varphi)^2$. However, we can also just solve a subsystem of $\mathcal{M}_{d_h}^\varphi + \mathcal{M}_{d_g}^\varphi$ equations and see if its solution fits the other equations. Then we get a complexity in the order of $(\mathcal{M}_{d_h}^\varphi + \mathcal{M}_{d_g}^\varphi)^3$.

It is possible to significantly reduce the complexity further by a good choice of the subsystem to be solved and by making use of its structure. Denote again the system we are solving by

$$M \cdot \bar{c}_{g,h} = \bar{0}, \quad (15)$$

where $\bar{c}_{g,h}$ is the $(\mathcal{M}_{d_h}^\varphi + \mathcal{M}_{d_g}^\varphi) \times 1$ column matrix containing the ANF coefficients $c_h(\bar{a})$ ordered lexicographically, where we intercalate the coefficients $c_g(\bar{a})$ right before the corresponding $c_h(\bar{a})$ as long as such coefficients exist - recall that $d_g < d_h$. The matrix M is a $(\mathcal{M}_{d_h}^\varphi + \mathcal{M}_{d_g}^\varphi) \times (\mathcal{M}_{d_h}^\varphi + \mathcal{M}_{d_g}^\varphi)$ matrix of which we choose the rows as follows: the first row corresponds to the equation (14) with $wt(\bar{x}) = 0$. The following φ rows correspond to the equations (14) with $wt(\bar{x}) = 1$ ordered lexicographically, followed by the $\binom{\varphi}{2}$ rows corresponding to the equations (14) with $wt(\bar{x}) = 2$. We continue this procedure until the $\binom{\varphi}{d_h}$ rows corresponding to the equations (14) with $wt(\bar{x}) = d_h$. The remaining $\mathcal{M}_{d_g}^\varphi$ rows of M are then filled up by choosing other values $\bar{x}$ at random for obtaining additional equations (14).

Again, we can reduce the complexity significantly by looking at the structure of this M . Let's look at the $\mathcal{M}_{d_h}^\varphi$ first rows of this matrix. It is easy to see that this part of the matrix is in *echelon form* and its size is exactly $\mathcal{M}_{d_h}^\varphi \times (\mathcal{M}_{d_h}^\varphi + \mathcal{M}_{d_g}^\varphi)$. The solution of this underdetermined system of equations can be found efficiently, requiring order of $(\mathcal{M}_{d_h}^\varphi)^2 + \mathcal{M}_{d_h}^\varphi \mathcal{M}_{d_g}^\varphi \approx (\mathcal{M}_{d_h}^\varphi)^2$ operations. The $\mathcal{M}_{d_h}^\varphi$ pivot columns correspond to the coefficients $c_h(\bar{a})$ with $wt(\bar{a}) \leq d_h$, the $\mathcal{M}_{d_g}^\varphi$ nonpivot columns correspond to the coefficients $c_g(\bar{a})$ with $wt(\bar{a}) \leq d_g$.

We now substitute this into the remaining equations. We need to substitute order of $\mathcal{M}_{d_h}^\varphi$ coefficients by $\mathcal{M}_{d_g}^\varphi$ coefficients and do this in $\mathcal{M}_{d_g}^\varphi$ rows, resulting in a substitution complexity of order $\mathcal{M}_{d_h}^\varphi \cdot (\mathcal{M}_{d_g}^\varphi)^2$. We can then solve this

system to obtain the coefficients $c_g(\bar{a})$ through Gaussian elimination, requiring order of $(\mathcal{M}_{d_g}^\varphi)^3$ operations. Finally, we introduce the solution for the coefficients $c_g(\bar{a})$ with $wt(\bar{a}) \leq d_g$ into the upper part of the matrix and can efficiently find the solution for the coefficients $c_h(\bar{a})$ with $wt(\bar{a}) \leq d_h$.

The gain in complexity is much more important here than in the case of the previous section. Adding up the complexities of the steps and discarding terms that will not be relevant for the analysis, the time complexity will be dominated by either finding the solution of the echelon matrix or by the substitution complexity. Our algorithm hence has a time complexity of order $\mathcal{M}_{d_h}^\varphi \cdot (\mathcal{M}_{d_g}^\varphi)^2 + (\mathcal{M}_{d_h}^\varphi)^2$. To make the algorithm memory efficient, we will process the matrix row by row and only store the nonpivot elements for later use, resulting in a memory complexity of order $\mathcal{M}_{d_h}^\varphi \mathcal{M}_{d_g}^\varphi$.

5 Theoretical Bounds on the Existence of $(d_g, d_h) - \mathcal{R}$

Relation between φ and $(d_g, d_h) - \mathcal{R}$. We will first look into the relation between the existence of $(d_g, d_h) - \mathcal{R}$ and the number of inputs φ of the Boolean function. To do so, we first investigate the affine invariance of $(d_g, d_h) - \mathcal{R}$. For AI, it holds that the AI of affine equivalent functions can differ with one. Only for functions that are affine equivalent in the input variables, the AI is invariant. We now show that $(d_g, d_h) - \mathcal{R}$ satisfies a stronger invariance:

Theorem 1. *The $(d_g, d_h) - \mathcal{R}$ (with $d_g < d_h$) of Boolean functions is an affine invariant property.*

Proof. Let $f(\bar{x})$ satisfy a $(d_g, d_h) - \mathcal{R}$ with $d_g < d_h$:

$$f(\bar{x}) \cdot g(\bar{x}) = h(\bar{x}). \quad (16)$$

Then for any function f' that is affine equivalent to f expressed by Equation (2), the following relation holds:

$$\begin{aligned} f'(\bar{x}) \cdot g(\bar{x}A \oplus \bar{b}) &= (f(\bar{x}A \oplus \bar{b}) \oplus \bar{x} \cdot \bar{c} \oplus d) \cdot g(\bar{x}A \oplus \bar{b}) \\ &= [f(\bar{x}A \oplus \bar{b}) \cdot g(\bar{x}A \oplus \bar{b})] \oplus [(\bar{x} \cdot \bar{c} \oplus d)g(\bar{x}A \oplus \bar{b})] \end{aligned} \quad (17)$$

In the right hand side, the first term is equal to $h(\bar{x}A \oplus \bar{b})$ because of (16), and the second term has degree $\leq d_h$ because $d_g < d_h$. So (17) shows that f' satisfies $(d_g, d_h) - \mathcal{R}$. $\square$

A bound on the relation between $(d_g, d_h) - \mathcal{R}$ and φ has already been given in [3, Theorem 7.2.1] by Courtois:

Theorem 2. *A Boolean function f always satisfies $(d_g, d_h) - \mathcal{R}$ for any d_g and d_h such that $d_g + d_h \geq \varphi$*

Relation between d_f and $(d_g, d_h) - \mathcal{R}$. Second, we investigate the bounds posed by the degree of the Boolean function on the existence of $(d_g, d_h) - \mathcal{R}$. The following straightforward relation exists between $(d_g, d_h) - \mathcal{R}$ and the degree (also mentioned in [3, Theorem 7.1.1]) .

Theorem 3. *If f has degree d_f , then f satisfies $(i, d_f + i) - \mathcal{R}$ for every $i < d_f$.*

Relation between AI and $(d_g, d_h) - \mathcal{R}$. We now show that if f satisfies $(d_g, d_h) - \mathcal{R}$, the parameter d_h is greater than or equal to $\text{AI}(f)$, due to the following relation with the annihilators of f and $f \oplus 1$.

Lemma 1. *Let f, g, h be three Boolean functions on $\mathbb{F}_2^\varphi$. The equation $f \cdot g = h$ is satisfied if and only if both $f \cdot (g \oplus h) = 0$ and $(f \oplus 1) \cdot h = 0$ hold simultaneously. In other words, if and only if $g \oplus h$ is an annihilator of f and h is an annihilator of $f \oplus 1$.*

Proof. Let $f \cdot g = h$. Multiplying both sides of the equation with f leads to $f^2 \cdot g = f \cdot h$. Consequently, $f \cdot g = f \cdot h$ and $f \cdot h = h$. The other side of the implication is trivial. $\square$

As a consequence, for every d_h greater than or equal to $\text{AI}(f)$, the lowest possible d_g such that f satisfies $(d_g, d_h) - \mathcal{R}$ could be found by searching the lowest degree function which is a linear combination of functions belonging to the set of annihilators with degree d_h of f and $f \oplus 1$. We have implemented an algorithm that uses this methodology. This algorithm is less efficient than the one described in Sect. 4.2, but the implementation effort of this new algorithm is more complex and is work in progress.

6 Application to Concrete Designs

6.1 SFINKS

SFINKS [2] is a filter generator with 80-bit secret key and 256-bit internal state. The filter function has 17 inputs and degree 15 ($\varphi = 17$ and $d_f = 15$). The resistance against AAs was measured using the algorithm of [7], resulting in an AI of 6. To analyze the resistance against FAAs, we need to check whether the function satisfies $(d_g, d_h) - \mathcal{R}$ in all cases where the corresponding attack complexity is smaller than 2^{80} . Table 1 presents the total complexity order (i.e., the precomputation, substitution and solving complexities together) of the FAA for an arbitrary filter generator with internal size 256 and filter function which satisfies $(d_g, d_h) - \mathcal{R}$, only when $d_h \geq 6$ because of Lemma 1.

Theoretical bounds. From Theorem 2, it follows that f satisfies $(1, 16) - \mathcal{R}$, $(2, 15) - \mathcal{R}$, $(3, 14) - \mathcal{R}$, ... but none of these are a threat. From Theorem 3, it follows that f satisfies $(1, 16) - \mathcal{R}$, $(2, 17) - \mathcal{R}$, ... which is not a problem as well.

Table 1. $\log_2$ of order of complexities of FAA for a 256-bit state if f satisfies $(d_g, d_h) - \mathcal{R}$.

$d_g \backslash d_h$	6	7	8	9	10	11	12
1	52.73	58.10	63.20	68.12	72.88	77.47	81.93
2	59.73	65.10	70.20	75.12	79.88	84.47	88.93
3	66.14	71.48	76.61	81.53	86.28	90.88	95.34
4	76.94	77.47	82.60	87.52	92.27	96.87	101.33
5	92.83	92.83	92.83	93.18	97.94	102.53	106.99

Searching for relations. It is possible that we can improve on the theoretical bounds by actually searching for relations. With our algorithm of Sect. 4.2, this can be done easily. We only need to verify the existence of relations for which the attack complexity is less than 2^{80} . The complexity of doing so can be calculated to be of order 2^{39} . Courtois [4] already checked for the existence of such relations, using the old algorithm but which found the relations in a few days. We repeated his experiments with the old algorithm and confirm his simulations.

In particular, Courtois found that f satisfies $(4, 6) - \mathcal{R}$, $(3, 7) - \mathcal{R}$ and $(2, 8) - \mathcal{R}$. Note that if one assumes that the number of key stream bits is limited to 2^{40} (as stated in [2]), not all of these attacks are possible. But still the $(4, 6) - \mathcal{R}$ and $(3, 7) - \mathcal{R}$ (using several relations simultaneously for the latter) attacks are practical and can be mounted faster than exhaustive key search.

Remark 1. Due to the particular form of the filter function f in SFINKS, i.e., $f(\bar{x}, x_{16}) : \mathbb{F}_2^{17} \rightarrow \mathbb{F}_2 : \bar{x} \mapsto f_1(\bar{x}) \oplus x_{16}$, where f_1 is a Boolean function on $\mathbb{F}_2^{16}$ affine equivalent to the trace function of the inverse function on $\mathbb{F}_2^{16}$, we can restrict us to the computation of the $(d_g, d_h) - \mathcal{R}$ with $d_g < d_h$ of f_1 . This follows from the observation that

$$\begin{aligned} f_1 \cdot g = h &\Rightarrow (f_1 \oplus x_{16}) \cdot g = h \oplus x_{16} \cdot g \\ f_1 \cdot g = h \oplus x_{16} \cdot g &\Leftarrow (f_1 \oplus x_{16}) \cdot g = h. \end{aligned}$$

In general, we have the following theorem.

Theorem 4. *Let f_i be a Boolean function on $\mathbb{F}_2^i$ that satisfies $(e_i, d_i) - \mathcal{R}$ for $i = 1, 2$. Then $f_1 \oplus f_2$ on $\mathbb{F}_2^{n_1+n_2}$ satisfies also $(e_1, \max\{\deg(f_2) + e_1, d_1\}) - \mathcal{R}$ and $(e_2, \max\{\deg(f_1) + e_2, d_2\}) - \mathcal{R}$.*

6.2 WG

The stream cipher WG [8] is also a simple filter generator with 80-bit secret key, 319-bit internal state and a filter function with 29 inputs and degree 11. The AI could not be measured yet. For the FAA, we should again check whether f satisfies $(d_g, d_h) - \mathcal{R}$ for all cases where the attack complexity is less than 2^{80} . Table 2 presents these complexity orders.

Table 2. $\log_2$ of order of complexities of FAA for a 319-bit state if f satisfies $(d_g, d_h) - \mathcal{R}$.

$d_g \backslash d_h$	1	2	3	4	5	6	7	8	9	10	11
1	23.36	28.93	36.17	42.84	49.09	55.03	60.70	66.15	71.40	76.48	81.41
2	0	43.91	43.91	50.16	56.41	62.35	68.02	73.47	78.72	83.80	88.73
3	0	0	62.79	62.79	63.14	69.07	74.75	80.19	85.45	90.53	95.45
4	0	0	0	80.50	80.50	80.50	81.05	86.50	91.76	96.84	101.76

Theoretical bounds. From Theorem 2, it follows that f satisfies $(1, 28) - \mathcal{R}$, $(2, 27) - \mathcal{R}$, $(3, 26) - \mathcal{R}$, ... but none of these are a threat. From Theorem 3, it follows that f satisfies $(1, 12) - \mathcal{R}$, $(2, 13) - \mathcal{R}$, ...

Let us investigate this $(1, 12) - \mathcal{R}$ into more detail. We know that there exist at most 2^{30} equations of this type. These equations are simply found by multiplying the function with an arbitrary affine function which increases the degree with at most one. The required key stream length is equal to $2^{70.73}$ and the attack complexities are as follows.

- Precomputation complexity: $2^{81.958}$
- Substitution complexity: $2^{86.19}$
- Solving complexity: $2^{23.36}$

In the proposal of the stream cipher WG [8], the key stream has been restricted to sequences of length 2^{45} . A FAA may still exist if at least $2^{25.73}$ out of the 2^{30} equations are linearly independent. In this situation, the substitution complexity will increase with a factor of $2^{25.73}$, leading to a complexity of $2^{111.4}$.

Searching for relations. It may also be that better FAAs can be performed on WG. To check this, we need to run our algorithm and search for relations. Table 3 presents the complexity of our algorithm for determining if f is $(d_g, d_h) - \mathcal{R}$. We

Table 3. $\log_2$ of order of complexities for the Relation Search Step for WG with new algorithm

$d_g \backslash d_h$	1	2	3	4	5	6	7	8	9	10	11
1	14.77	19.15	24.28	29.58	34.33	38.49	42.12	45.25	47.95	50.24	0
2	0	26.31	29.57	32.50	35.52	38.88	42.24	45.30	47.97	0	0
3	0	0	36.00	38.76	41.17	43.29	45.23	0	0	0	0

expect that most of these values are attainable when implementing the algorithm on a modern PC. On the contrary, the old algorithm would have an infeasible complexity of order 2^{75} .

7 Conclusion

In this paper, we have presented a framework to evaluate the resistance of memoryless nonlinear filters and combiners against FAAs, using the (non-)existence of (d_g, d_h) -relations to assess the security. We developed a new algorithm to efficiently calculate these relations, and presented some bounds that relate the resistance against FAAs to some properties of the Boolean function. Finally, we have illustrated the application of our approach on two eSTREAM candidates, SFINKS (as previously done by Courtois) and WG. Further study of FAAs is still required, especially towards general classes of these attacks.

Acknowledgements. This work was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and by the European Commission through the IST Programme ECRYPT. An Braeken is an FWO Research Assistant, sponsored by the Fund for Scientific Research - Flanders (Belgium), Joseph Lano is financed by a PhD grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). The authors also would like to thank Nicolas Courtois and the anonymous referees for their useful comments.

References

1. F. Armknecht and M. Krause. Algebraic attacks on combiners with memory. In D. Boneh, editor, *Crypto 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 162–175. Springer-Verlag, 2003.
2. A. Braeken, J. Lano, N. Mentens, B. Preneel, and I. Verbauwhede. SFINKS: A synchronous stream cipher for restricted hardware environments. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/035, 2005. <http://www.ecrypt.eu.org/stream>.
3. N. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In D. Boneh, editor, *Crypto 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 176–194. Springer-Verlag, 2003.
4. N. Courtois. Cryptanalysis of SFINKS. In M. Rhee and B. Lee, editors, *Information Security and Cryptology - ICISC 2005*, number 3935 in *Lecture Notes in Computer Science*. Springer-Verlag, 2005.
5. N. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. In E. Biham, editor, *Eurocrypt 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer-Verlag, 2003.
6. P. Hawkes and G. Rose. Rewriting variables: The complexity of fast algebraic attacks on stream ciphers. In M. Franklin, editor, *Crypto 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 390–406. Springer-Verlag, 2004.
7. W. Meier, E. Pasalic, and C. Carlet. Algebraic attacks and decomposition of Boolean functions. In C. Cachin and J. Camenisch, editors, *Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 474–491. Springer-Verlag, 2004.
8. Y. Nawaz and G. Gong. The WG stream cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/033, 2005. <http://www.ecrypt.eu.org/stream>.
9. V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.