Algebraic Immunity of S-boxes and Augmented Functions

Simon Fischer and Willi Meier

FHNW, CH-5210 Windisch, Switzerland
{simon.fischer, willi.meier}@fhnw.ch

Abstract. In this paper, the algebraic immunity of S-boxes and augmented functions of stream ciphers is investigated. Augmented functions are shown to have some algebraic properties that are not covered by previous measures of immunity. As a result, efficient algebraic attacks with very low data complexity on certain filter generators become possible. In a similar line, the algebraic immunity of the augmented function of the eSTREAM candidate Trivium is experimentally tested. These tests suggest that Trivium has some immunity against algebraic attacks on augmented functions.

Key words: S-box, Stream Cipher, Augmented Function, Algebraic Attack, Filter Generator, Trivium

1 Introduction

Algebraic attacks can be efficient against stream ciphers based on LFSR's [12], and potentially against block ciphers based on S-boxes [13]. In the case of stream ciphers, the algebraic immunity $\mathcal{AI}$ of the filter function is a measure for the complexity of conventional algebraic attacks. However, it turned out in some cases that large $\mathcal{AI}$ did not help to prevent fast algebraic attacks (FAA's). It is an open question if immunity against FAA's is a sufficient criterion for any kind of algebraic attacks on stream ciphers. In the case of block ciphers, the algebraic immunity of S-boxes is a measure for the complexity of a very general type of algebraic attacks, considering implicit or conditional equations [2, 13]. Present methods for computation of $\mathcal{AI}$ of S-boxes are not very efficient, only about $n = 20$ variables are computationally feasible (except for power mappings, see [11, 20]).

In this paper, we integrate the general approach for S-boxes in the context of stream ciphers and generalise the concept of algebraic immunity of stream ciphers (see Open Problem 7 in [7]). More precisely, we investigate conditional equations for augmented functions of stream ciphers and observe some algebraic properties (to be used in an attack), which are not covered by the previous definitions of $\mathcal{AI}$. As a consequence, immunity against FAA's is not sufficient to prevent any kind of algebraic attack: Depending on the Boolean functions used in a stream cipher, we demonstrate that algebraic properties of the augmented function allow for attacks which need much less known output than established algebraic attacks. This induces some new design criteria for stream ciphers. Time complexity of our attacks is derived by intrinsic properties of the augmented function. Our framework can be applied to a large variety of situations. We present two applications (which both have been implemented). First, we describe efficient attacks on some filter generators. For example, we can efficiently recover the state of a filter generator based on certain Boolean functions when an amount of output data is available which is only linear in the length of the driving LFSR. This should be compared to the data complexity of conventional algebraic attacks, which is about $\binom{n}{e}$, where n is the length of the LFSR and e equals the algebraic immunity of the filter function. Our investigation of the augmented function allows to contribute to open problems posed in [15], and explains why algebraic attacks using Gröbner bases against filter generators are in certain cases successful even for a known output segment only slightly larger than the LFSR length. In a second direction, a large scale experiment carried out with the eSTREAM focus candidate Trivium suggests some immunity of this cipher against algebraic attacks on augmented functions. This experiment becomes feasible as for Trivium with its 288-bit state one can find preimages of 144-bit outputs in polynomial time.

Augmented functions of LFSR-based stream ciphers have previously been studied, *e.g.* in [1], [16] and [19], where it had been noticed that the augmented function can be weaker than a single output function, with regard to (conditional) correlation attacks as well as to inversion attacks. However, for the first time, we analyse the $\mathcal{AI}$ of (sometimes quite large) augmented functions. Surprisingly, augmented functions did not receive much attention in this context yet.

The paper is organised as follows: In Sect. 2, we investigate some algebraic properties of S-boxes. Our general ideas of algebraic attacks on augmented functions (which are some special S-boxes) are presented in Sec. 3. In Sect. 4, this framework is discussed for filter generators. Sect. 5 and Sect. 6 contain applications of our method, namely for some specific filter generators and for eSTREAM candidate Trivium.

2 Algebraic Properties of S-boxes

Let $\mathbb{F}$ denote the finite field GF(2), and consider the vectorial Boolean function (or S-box) $S : \mathbb{F}^n \to \mathbb{F}^m$ with $S(x) = y$, where $x := (x_1, \ldots, x_n)$ and $y := (y_1, \ldots, y_m)$. In the case of $m = 1$, the S-box reduces to a Boolean function, and in general, the S-box consists of m Boolean functions $S_i(x)$. These functions give rise to the explicit equations $S_i(x) = y_i$. Here, we assume that y is known and x is unknown.

2.1 Implicit Equations

The S-box can hide implicit equations, namely $F(x, y) = 0$ for each $x \in \mathbb{F}^n$ and with $y = S(x)$. The algebraic normal form of such an equation is denoted

$F(x, y) = \sum c_{\alpha,\beta} x^\alpha y^\beta = 0 \mod 2$, with coefficients $c_{\alpha,\beta} \in \mathbb{F}$, multi-indices $\alpha, \beta \in \mathbb{F}^n$ (which can likewise be identified by their integers) and the notation $x^\alpha := (x_1^{\alpha_1} \cdots x_n^{\alpha_n})$. In the context of algebraic attacks, it is of interest to focus on implicit equations with special structure, *e.g.* on sparse equations or equations of small degree. Let the degree in x be $d := \max\{|\alpha|, c_{\alpha,\beta} = 1\} \leq n$ with the weight $|\alpha|$ of α, and consider an unrestricted degree for the known y, hence $\max\{|\beta|, c_{\alpha,\beta} = 1\} \leq m$. The maximum number of monomials (or coefficients) in an equation of degree d corresponds to $2^m D$, where $D := \sum_{i=0}^{d} \binom{n}{i}$. In order to determine the existence of an implicit equation of degree d, consider a matrix M in $\mathbb{F}$ of size $2^n \times 2^m D$. Each row corresponds to an input x, and each column corresponds to an evaluated monomial (with some fixed order). If the number of columns in M is larger than the number of rows, then linearly dependent columns (*i.e.* monomials) exist, see [9,13]. The rank of M determines the number of solutions, and the solutions correspond to the kernel of M^T. Any non-zero implicit equation (which holds for each input x) may then depend on x and y, or on y only. If it depends on x and y, then the equation may degenerate for some values of y. For example, $x_1 y_1 + x_2 y_1 = 0$ degenerates for $y_1 = 0$.

2.2 Conditional Equations

As the output is assumed to be known, one could investigate equations which are conditioned by the output y, hence $F_y(x) = 0$ for each preimage $x \in S^{-1}(y)$ and of degree d in x. The number of preimages is denoted $U_y := |S^{-1}(y)|$, where $U_y = 2^{n-m}$ for balanced S and $m \leq n$. Notice that conditional equations for different outputs y need not be connected in a common implicit equation, and one can find an optimum equation (*i.e.* an equation of minimum degree) for each output y. Degenerated equations are not existing in this situation, and the corresponding matrix M_y has a reduced size of $U_y \times D$. Similar to the case of implicit equations, one obtains:

Proposition 1. *Consider an S-box $S : \mathbb{F}^n \to \mathbb{F}^m$ and let $S(x) = y$. Then, the number of (independent) conditional equations of degree at most d for some y is $R_y = D - \mathrm{rank}(M_y)$. A sufficient criterion for the existence of a non-zero conditional equation is $0 < U_y < D$.*

The condition $R_y > 0$ requires some minimum value of d, which can depend on y. As already proposed in [2], this motivates the following definition of algebraic immunity for S-boxes:

Definition 1. *Consider an S-box $S : \mathbb{F}^n \to \mathbb{F}^m$. Given some fixed output y, let d be the minimum degree of a non-zero conditional equation $F_y(x) = 0$ which holds for all $x \in S^{-1}(y)$. Then the algebraic immunity $\mathcal{AI}$ of S is defined by the minimum of d over all $y \in \mathbb{F}^m$.*

The $\mathcal{AI}$ can be bounded, using the sufficient condition of Prop. 1. Let d_0 be the minimum degree such that $D > 2^{n-m}$. If the S-box is surjective, then there exists at least one y with a non-zero conditional equation of degree at most d_0, hence $\mathcal{AI} \leq d_0$. In addition, the block size m of the output could be considered as a parameter (by investigating truncated S-boxes S_m, corresponding to partial conditioned equations for S). Let $m_0 := \lfloor n - \log_2 D + 1 \rfloor$ for some degree d. Then, the minimum block size m to find non-zero conditional equations of degree at most d is bounded by m_0. See Tab. 1 for some numerical values of m_0.

Table 1. Theoretical block size m_0 for different parameters n and d.

d \ n	16	18	20	32	64	128
1	12	14	16	27	58	121
2	9	11	13	23	53	115
3	7	9	10	20	49	110

A single output y is called *weak*, if non-zero conditional equations of degree d exist for $U_y \gg D$ (or if the output is strongly imbalanced). This roughly corresponds to the condition $d \ll d_0$, or $m \ll m_0$.

2.3 Algorithmic Methods

As already mentioned in [7], memory requirements to determine the rank of M are impractical for about $n > 20$. In the case of conditional equations, the matrix M_y can be much smaller, but the bottleneck is to compute an exhaustive list of preimages, which requires a time complexity of 2^n. However, one could use a probabilistic variant of this basic method: Instead of determining the rank of M_y which contains all U_y inputs x, one may solve for a smaller matrix M'_y with $V < U_y$ random inputs. Then, one can determine the *non-existence* of a solution: If no solution exists for M'_y, then no solution exists for M_y either. On the other hand, if one or more solutions exist for M'_y, then they hold true for the subsystem of V inputs, but possibly not for all U_y inputs. Let the probability p be the fraction of preimages that satisfy the equation corresponding to such a solution. With the heuristical argument $(1-p)V < 1$, we expect that $p > 1-1/V$. However, this argument holds only for $V > D$, because otherwise, there are always at least $D - V$ solutions (which could be balanced). Consequently, if V is a small multiple of D, the probability can be quite close to one. For this reason, all solutions of the smaller system can be useful in later attacks. Determining only a few random preimages can be very efficient: In a naive approach, time complexity to find a random preimage of an output y is about $2^n/U_y$ (which is 2^m for balanced S), and complexity to find D preimages is about $2^n D/U_y$. This is an improvement compared to the exact method if $U_y \gg D$, *i.e.* equations can be found efficiently for weak outputs. Memory requirements of the probabilistic algorithm are about $C_M = D^2$, and time complexity is about $C_T = D2^m + D^3$. Computation of $\mathcal{AI}$ requires about $C_T = D2^m + D^3 2^m = \mathcal{O}(D^3 2^m)$.

3 Algebraic Attacks based on the Augmented Function

In this section, we focus on algebraic cryptanalysis of S-boxes in the context of stream ciphers. Given a stream cipher, one may construct an S-box as follows:

Definition 2. *Consider a stream cipher with internal state x of n bits, an update function L, and an output function f which outputs one bit of keystream in a single iteration. Then, the augmented function S_m is defined by*

$$\begin{aligned} S_m : \mathbb{F}^n &\to \mathbb{F}^m \\ x &\mapsto (f(x), f(L(x)), \dots, f(L^{m-1}(x)) \ . \end{aligned} \tag{1}$$

The update L can be linear (*e.g.* for filter generators), or nonlinear (*e.g.* for Trivium). The input x correspond to the internal state at some time t, and the output y corresponds to an m-bit block of the known keystream. Notice that m is a very natural parameter here. The goal is to recover the initial state x by algebraic attacks, using (potentially probabilistic) conditional equations $F_y(x) = 0$ of degree d for outputs y of the augmented function S_m. This way, one can set up equations for state variables of different time steps t. In the case of a linear update function L, each equation can be transformed into an equation of degree d in the initial state variables x. In the case of a nonlinear update function L, the degree of the equations is increasing with time. However, the nonlinear part of the update is sometimes very simple, such that equations for different time steps can be efficiently combined. Finally, the system of equations in the initial state variables x is solved.

If the augmented function has some weak outputs, then conditional equations can be found with the probabilistic algorithm of Sect. 2.3, which requires about D preimages of a single m-bit output. One may ask if there is a dedicated way to compute random preimages of m-bit outputs in the context of augmented functions. Any stream cipher as in Def. 2 can be described by a system of equations. Nonlinear systems of equations with roughly the same number of equations as unknowns are in general NP-hard to solve. However, due to the special (simple) structure of some stream ciphers, it may be easy to partially invert the nonlinear system. For example, given a single bit of output of a filter generator, it is easy to find a state which gives out this bit. Efficient computation of random preimages for m-bit outputs is called *sampling*. The maximal value of m for which states can be sampled without trial and error is called sampling resistance of the stream cipher. Some constructions have very low sampling resistance, see [4, 5].

The parameters of our framework are the degree d of equations, and the block-size m of the output. An optimal tradeoff between these parameters depends on the algebraic properties of the augmented function. The attack is expected to be efficient, if:

1. There are many low-degree conditional equations for S_m.
2. Efficient sampling is possible for this block size m.

This measure is well adapted to the situation of augmented functions, and can be applied to sometimes quite large augmented functions, see Sect. 5 and 6. This way, we intend to prove some immunity of a stream cipher, or present attacks with reduced complexity.

4 Generic Scenarios for Filter Generators

Our framework is investigated in-depth in the context of LFSR-based stream ciphers (and notably for filter generators), which are the main target of conventional and fast algebraic attacks (see also Appendix A). We describe some elementary conditional equations induced by annihilators. Then, we investigate different methods for sampling, which are necessary to efficiently set up conditional equations. We suggest a basic scenario and estimate data complexity of an attack, the scenario is refined and improved.

4.1 Equations Induced by Annihilators

Let us first discuss the existence of conditional equations of degree $d = \mathcal{AI}$, where $\mathcal{AI}$ is the ordinary algebraic immunity of f here. With $m = 1$, the number of conditional equations for $y = 0$ (resp. $y = 1$) corresponds to the number of annihilators of $f + 1$ (resp. f) of degree d. If one increases m, then all equations originating from annihilators are involved: For example, if there is 1 annihilator of degree d for both f and $f + 1$, then the number of equations is expected to be at least m for any m-bit output y. Notice that equations of fast algebraic attacks are not involved if m is small compared to n.

4.2 Sampling

Given an augmented function S_m of a filter generator, the goal of sampling is to efficiently determine preimages x for fixed output $y = S_m(x)$ of m bits. Due to the special structure of the augmented function, there are some efficient methods for sampling:

Filter Inversion One could choose a fixed value for the k input bits of the filter, such that the observed output bit is correct (using a table of the filter function). This can be done for about n/k successive output bits, until the state is unique. This way, preimages of an output y of n/k bits can be found in polynomial time, and by partial search, preimages of larger outputs can be computed. Time complexity to find a preimage of $m > n/k$ bits is about $2^{m-n/k}$, *i.e.* the method is efficient if there are only few inputs k.

Linear Sampling In each time step, a number of l linear conditions are imposed on the input variables of f, such that the filter becomes linear. The linearised filter gives one additional linear equation for each keystream bit. Notice that all variables can be expressed by a linear function of the n variables of the initial state. Consequently, for an output y of m bits, one obtains $(l+1)m$ (inhomogeneous) linear equations for n unknowns, *i.e.* we expect that preimages can be found in polynomial time if $m \leq n/(l+1)$. To find many different preimages, one should have several independent conditions (which can be combined in a different way for each clock cycle).

In practice, sampling should be implemented carefully in order to avoid contradictions (*e.g.* with appropriate conditions depending on the keystream), see [5].

4.3 Basic Scenario

We describe a basic scenario for algebraic attacks on filter generators based on the augmented function: With C_D bits of keystream, one has $C'_D = C_D - m + 1$ (overlapping) windows of m bits. Assume that there are $R := \sum_y R_y$ equations of degree d for m-bit outputs y. For each window, we have about $r := R/2^m$ equations, which gives a total of $N = rC'_D$ equations.[1] Each equation has at most D monomials in the initial state variables, so we need about the same number of equations to solve the system by linearisation. Consequently, data complexity is $C_D = D/r + m - 1$ bits. The initial state can then be recovered in $C_T = D^3$. This should be compared with the complexity of conventional algebraic attacks $C_D = 2E/R_A$ and $C_T = E^3$, where $e := \mathcal{AI}$, $E := \sum_{i=0}^{e} \binom{n}{i}$, and R_A the number of annihilators of degree e. Notice that the augmented function may give low-degree equations, which are not visible for single-bit outputs; this increases information density and may reduce data complexity. Our approach has reduced time complexity if $d < e$, provided that sampling (and solving the matrix) is efficient.

4.4 Refined Basic Scenario

The basic scenario for filter generators should be refined in two aspects, concerning the existence of dependent and probabilistic equations: First, with overlapping windows of m bits, it may well happen that the same equation is counted several times, namely if the equation already exists for a substring of $m' < m$ bits (*e.g.* in the case of equations produced by annihilators). In addition, equations may be linearly dependent by chance. If this is not considered in the computation of R, one may have to enlarge data complexity a little bit. Second, one can expect to obtain probabilistic solutions. However, depending on the number of computed preimages, the probability p may be large and the corresponding equations can still be used in our framework, as they increase R and reduce data complexity, but potentially with some more cost in time. As we need about D (correct and linearly independent) equations to recover the initial state, the probability p should be at least $1 - 1/D$ (together with our estimation for p, this justifies that the number of preimages should be at least D). In the case

[1] From a heuristical point of view, the parameter r is only meaningful if the conditional equations are approximately uniformly distributed over all outputs y.

of a contradiction, one could complement a few equations in a partial search and solve again, until the keystream can be verified. Depending on the actual situation, one may find an optimal tradeoff in the number of computed preimages. Notice that our probabilistic attack deduced from an algebraic attack with equations of degree 1 is a powerful variant of a conditional correlation attack, see [19]. A probabilistic attack with nonlinear equations is a kind of higher order correlation attack, see [8].

4.5 Substitution of Equations

It is possible to further reduce data complexity in some cases. Consider the scenario where one has $N = rC'_D$ linear equations. On the other hand, given an annihilator of degree $e := \mathcal{AI}$, one can set up a system of degree e as in conventional algebraic attacks. The N linear equations can be substituted into this system in order to eliminate N variables. This results in a system of $D' := \sum_{i=0}^{e} \binom{n-N}{i}$ monomials, requiring a data complexity of $C_D = D'$ and time complexity $C_T = D'^3$. Notice that data can be reused in this case, which gives the implicit equation in C_D. Obviously, a necessary condition for the success of this method is $rE > 1$. A similar improvement of data complexity is possible for nonlinear equations of degree d. One can multiply the equations by all monomials of degree $e - d$ in order to obtain additional equations of degree e, along the lines of XL [14] and Gröbner bases algorithms.

5 First Application: Some Specific Filter Generators

Many conventional algebraic attacks on filter generators require about $\binom{n}{e}$ output bits where e equals the algebraic immunity of the filter function. On the other hand, in [15], algebraic attacks based on Gröbner bases are presented, which in a few cases require only $n+\varepsilon$ data. It is an open issue to understand such a behavior from the Boolean function and the tapping sequence. We present attacks on the corresponding augmented functions, requiring very low data complexity. This means, we can identify the source of the above behavior, and in addition, we can use our method also for other functions.

5.1 Existence of Equations

In this subsection, we give extensive experimental results for different filter generators. Our setup is chosen as follows: The filter functions are instances of the CanFil family (see [15]) or the Majority functions. These instances all have five inputs and algebraic immunity 2 or 3. Feedback taps correspond to a random primitive feedback polynomial, and filter taps are chosen randomly in the class of full positive difference sets, see Tab. 4 in Appendix B for an enumerated specification of our setups. Given a specified filter generator and parameters d and m, we compute the number R_y of independent conditional equations $F_y(x) = 0$ of degree d for each output $y \in \mathbb{F}^m$. The overall number of equations $R := \sum_y R_y$

for $n = 20$ is recorded in Tab. 2. Thereby, preimages are computed by exhaustive search in order to exclude probabilistic solutions.

Table 2. Counting the number of linear equations R for the augmented function of different filter generators, with $n = 20$ bit input and m bit output.

Filter	m	R for setups $6 - 10$				
CanFil1	14	0	0	0	0	0
	15	3139	4211	3071	4601	3844
CanFil2	14	0	0	0	0	0
	15	2136	2901	2717	2702	2456
CanFil5	6	0	0	0	2	0
	7	0	0	0	8	0
	8	0	0	0	24	0
	9	0	0	0	64	0
	10	6	0	0	163	0
	11	113	0	2	476	0
	12	960	16	215	1678	29
Majority5	9	0	0	0	2	0
	10	1	10	1	18	1
	11	22	437	40	148	56

In the case of CanFil1 and CanFil2, linear equations exist only for $m \geq m_0 - 1$, independent of the setup. On the other hand, for CanFil5 and Majority5, there exist many setups where a large number of linear equations already exists for $m \approx n/2$, see Ex. 1. We conclude that the number of equations weakly depends on the setup, but is mainly a property of the filter function. The situation is very similar for other values of n, see Appendix C. This suggests that our results can be scaled to larger values of n. Let us also investigate existence of equations of higher degree: CanFil1 and CanFil2 have $\mathcal{AI} = 2$ and there is 1 annihilator for both f and $f + 1$, which means that at least m quadratic equations can be expected for an m-bit output. For each setup and $m < m_0 - 1$, we observed only few additional equations, whereas the number of additional equations is exploding for larger values of m. This was observed for many different setups and different values of n.

Example 1. Consider CanFil5 with $n = 20$ and setup 9. For the output $y =$ 000000 of $m = 6$ bits, there are exactly 2^{14} preimages, hence the matrix M_y has 2^{14} rows and $D = 21$ columns for $d = 1$. Evaluation of M_y yields a rank of 20, *i.e.* a nontrivial solution exists. The explicit solution is $F_y(x) = x_2 + x_4 + x_5 + x_6 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{17} = 0$. □

5.2 Probabilistic Equations

In the previous subsection, the size n of the state was small enough to compute a complete set of preimages for some m-bit output y. However, in any practical situation where n is larger, the number of available preimages is only a small multiple of D, which may introduce probabilistic solutions. Here is an example with $n = 20$, where the probability can be computed exactly:

Example 2. Consider again CanFil5 with $n = 20$ and setup 9. For the output $y = \mathtt{000000}$ of $m = 6$ bits, there are 2^{14} preimages and one exact conditional equation of degree $d = 1$. We picked 80 random preimages and determined all (correct or probabilistic) linear conditional equations. This experiment was repeated 20 times with different preimages. In each run, we obtained between 2 and 4 independent equations with probabilities $p = 0.98, \ldots, 1$. For example, the (probabilistic) conditional equation $F_y(x) = x_2 + x_3 + x_4 + x_7 + x_{10} + x_{16} + x_{17} + x_{18} = 0$ holds with probability $p = 1 - 2^{-9}$. □

In the above example, there are only few probabilistic solutions and they have impressively large probability, which makes the equations very useful in an attack. Notice that experimental probability is in good agreement with our estimation $p > 1 - 1/80 = 0.9875$. The situation is very similar for other parameters. With the above setup and $m = 10$, not only $y = \mathtt{000 \ldots 0}$ but a majority of outputs y give rise to linear probabilistic equations. In the case of CanFil1 and CanFil2, we did not observe linear equations of large probability for $m < m_0 - 1$. It is interesting to investigate the situation for larger values of n:

Example 3. Consider CanFil5 with $n = 40$ and setup 11. For the output $y = \mathtt{000 \ldots 0}$ of $m = 20$ bits, we determine 200 random preimages. With $d = 1$, evaluation of M_y yields a rank of 30, *i.e.* 11 (independent) solutions exist. With 2000 random preimages, we observed a rank of 33, *i.e.* only 3 solutions of the first system were detected to be merely probabilistic. An example of an equation is $F_y(x) = x_1 + x_8 + x_{10} + x_{14} + x_{15} + x_{18} + x_{19} + x_{26} + x_{31} + x_{34} = 0$. □

The remaining 8 solutions of the above example may be exact, or probabilistic with very high probability. By sampling, one could find (probabilistic) conditional equations for much larger values of n. For example, with CanFil5, $n = 80$, $m = 40$ and filter inversion, time complexity to find a linear equation for a weak output is around 2^{32}.

5.3 Discussion of Attacks

Our experimental results reveal that some filter functions are very vulnerable to algebraic attacks based on the corresponding augmented function. For CanFil5 with $n = 20$ and setup 9, we observed $R = 163$ exact equations using the parameters $m = 10$ and $d = 1$, which gives a ratio of $r = 0.16$. Including probabilistic equations, this ratio may be even larger. Here, preimages of any y can be found efficiently by sampling: using filter inversion, a single preimage can be found in $2^{m-n/k} = 2^6$ steps, and a single equation in around 2^{13} steps. Provided that equations are independent and the probability is large, data complexity is about $C_D = (n + 1)/r + m - 1 = 140$. The linear equations could also be substituted into the system of degree $\mathcal{AI} = 2$, which results in a data complexity of about $C_D = 66$. Notice that conventional algebraic attacks would require $C_D = E = 211$ bits (and time complexity E^3). As we expect that our observation can be scaled, (*i.e.* that r remains constant for larger values of n and $m = n/2$), data complexity is a linear function in n. Considering time complexity for variable n, the matrix M and the final system of equations can be solved in polynomial time, whereas sampling is subexponential (and polynomial in some cases, where linear sampling is possible).

In [15], CanFil5 has been attacked experimentally with $n + \varepsilon$ data, where $n = 40, \ldots, 70$ and $\varepsilon < 10$. Our analysis gives a conclusive justification for their observation. Other functions such as Majority5 could be attacked in a similar way, whereas CanFil1 and CanFil2 are shown to be much more resistant against this general attack: No linear equations have been found for $m < m_0 - 1$, and only few quadratic equations.

6 Second Application: Trivium

Trivium [6] is a stream cipher with a state of 288 bits, a nonlinear update and a linear output. It has a simple algebraic structure, which makes it an interesting candidate for our framework. We consider the S-box $S_m(x) = y$, where S is the augmented function of Trivium, x the state of $n = 288$ bits, and y the output of m bits. We will first analyse the sampling of S_m, which is very similar to linear sampling of filter generators.

6.1 Sampling

The state consists of the 3 registers $R_1 = (x_1, \ldots, x_{93})$, $R_2 = (x_{94}, \ldots, x_{177})$ and $R_3 = (x_{178}, \ldots, x_{288})$. In each clock cycle, a linear combination of 6 bits of the state (2 bits of each register) is output. Then, the registers are shifted to the right by one position, with a nonlinear feedback to the first position of each register. In the first 66 clocks, each keystream bit is a linear function of the input, whereas the subsequent keystream bit involves a nonlinear expression. Consequently, given any output of $m = 66$ bits, one can efficiently determine some preimages by solving a linear system. It is possible to find preimages of even larger output size. Observe that the nonlinear function is quadratic, where the two factors of the product have subsequent indices. Consequently, one could fix some alternating bits of the state, which results in additional linear equations for the remaining variables. Let `c`, `l`, `q` denote constant, linear, and quadratic dependence on the initial state. Let all the even bits of the initial state be `c`, see Tab. 3. After update 83, bits 82 and 83 of R_2 are both `l`. Variable t_2 takes bits 82 and 83 of R_2 to compute the nonlinear term. So after update 84, $t_2 = x_{178}$ is `q` (where nonlinear terms in t_1 and t_3 appear somewhat later).
After 65 more updates, x_{243} is quadratic, where x_{243} is filtered out from R_3 in the next update (after 84 updates, other bits are also `q` and are filtered out from registers R_1 and R_2, but on a later point in time). Consequently, keystream bit

Table 3. Evolution of states with partially fixed input

Initial state	After 1 update	After 84 updates
$R_1 = \mathtt{lclcl}\ldots$	$R_1 = \mathtt{llclcl}\ldots$	$R_1 = \mathtt{lllll}\ldots$
$R_2 = \mathtt{clclc}\ldots$	$R_2 = \mathtt{lclclc}\ldots$	$R_2 = \mathtt{lllll}\ldots$
$R_3 = \mathtt{clclc}\ldots$	$R_3 = \mathtt{lclclc}\ldots$	$R_3 = \mathtt{qllll}\ldots$

number $66 + 84 = 150$ (counting from 1) is $\mathtt{q}$, and the first 149 keystream bits are linear in the remaining variables.

The number of remaining variables in the state (the degree of freedom) is 144. Consequently, for an output of size $m = 144$ bits, we can expect to find one solution for the remaining variables; this was verified experimentally. The solution (combined with the fixed bits) yields a preimage of y. Notice that we do not exclude any preimages this way. In addition, m can be somewhat larger with partial search for the additional bits.

Example 4. Consider the special output $y = \mathtt{000}\ldots\mathtt{0}$ of $m = 160$ bits. By sampling and partial exhaustive search, we find the nontrivial preimage

$$x = \begin{array}{l} \mathtt{1000101111000101110011000101010011010000010010010} \\ \mathtt{0001001001001100111110110111011000010011001010000} \\ \mathtt{1100000001010110011100001111110110011000011010100} \\ \mathtt{0111000000101010011001101111010101011111110100001} \\ \mathtt{0000010000011010001000011110011010101000101011111} \\ \mathtt{1010000011101001010100110001001110010100101011010} \end{array}$$

□

6.2 Potential Attacks

The nonlinear update of Trivium results in equations $S_m(x) = y$ of increasing degree for increasing values of m. However, for any output y, there are at least 66 linear equations in the input variables. It is an important and security related question, if there are additional linear equations for some fixed output y. A linear equation is determined by $D = 289$ coefficients, thus we have to compute somewhat more than 289 preimages for this output. By sampling, this can be done in polynomial time. Here is an experiment:

Example 5. Consider a prescribed output y of 144 bits, and compute 400 preimages x such that $S_m(x) = y$ (where the preimages are computed by a uniform random choice of 144 fixed bits of x). Given these preimages, set up and solve the matrix M of linear monomials in x. For 30 uniform random choices of y, we always observed 66 linearly independent solutions. □

Consequently, Trivium seems to be immune against additional linear equations, that might help in an attack. Because of the lack of probabilistic solutions, Trivium is also supposed to be immune against equations of large probability (compare with CanFil1 and CanFil2). As pointed out in [17], there are some states resulting in a weak output: If R_1, R_2 and R_3 are initialised by some period-3 states, then the whole state (and hence the output) repeats itself every 3 iterations. Each of these states results in $y = \mathtt{000}\ldots\mathtt{0}$. Here is an extended experiment (with partial exhaustive search) for this special output:

Example 6. Consider the output $y = \mathtt{000}\ldots\mathtt{0}$ of 150 bits, and compute 400 random preimages x such that $S_m(x) = y$. By solving the matrix M of linear monomials in x, we still observed 66 linearly independent solutions. □

7 Conclusions

Intrinsic properties of augmented functions of stream ciphers have been investigated with regard to algebraic attacks. Certain properties of the augmented function enable efficient algebraic attacks with lower data complexity than established algebraic attacks. In order to assess resistance of augmented functions against such improved algebraic attacks, a prespecified number of preimages of outputs of various size of these functions have to be found. For a random function, the difficulty of finding preimages increases exponentially with the output size. However, due to a special structure of the augmented function of a stream cipher, this can be much simpler than in the random case. For any such stream cipher, our results show the necessity of checking the augmented function for algebraic relations of low degree for output sizes for which finding preimages is feasible. In this paper, this has been successfully carried out for various filter generators as well as for the eSTREAM candidate Trivium.

Acknowledgments

This work is supported in part by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center of the Swiss National Science Foundation under grant number 5005-67322. The second author is supported by Hasler Foundation `www.haslerfoundation.ch` under project number 2005. We would like to thank Steve Babbage for encouraging us to study algebraic immunity of large S-boxes.

References

1. R. J. Anderson. Searching for the Optimum Correlation Attack. In *Fast Software Encryption - FSE 1994*.
2. F. Armknecht and M. Krause. Constructing Single- and Multi-Output Boolean Functions with Maximal Algebraic Immunity. In *ICALP 2006*.
3. F. Armknecht, C. Carlet, P. Gaborit, S. Künzli, W. Meier, and O. Ruatta. Efficient Computation of Algebraic Immunity for Algebraic and Fast Algebraic Attacks. In *Advances in Cryptology - EUROCRYPT 2006*.

4. S. Babbage. A Space/Time Tradeoff in Exhaustive Search Attacks on Stream Ciphers. In *European Convention on Security and Detection, IEE Conference Publication No. 408, 1995.*
5. A. Biryukov and A. Shamir. Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. In *Advances in Cryptology - ASIACRYPT 2000.*
6. C. de Cannière and B. Preneel. Trivium - A Stream Cipher Construction Inspired by Block Cipher Design Principles. In *eSTREAM, ECRYPT Stream Cipher Project, Report 2005/030.*
7. A. Canteaut. Open Problems Related to Algebraic Attacks on Stream Ciphers. In *Workshop on Coding and Cryptography - WCC 2005.*
8. N. Courtois. Higher Order Correlation Attacks, XL algorithm and Cryptanalysis of Toyocrypt. In *Cryptology ePrint Archive, Report 2002/087.*
9. N. Courtois. Algebraic Attacks on Combiners with Memory and Several Outputs. In *Cryptology ePrint Archive, Report 2003/125.*
10. N. Courtois. How Fast can be Algebraic Attacks on Block Ciphers? In *Cryptology ePrint Archive, Report 2006/168.*
11. N. Courtois, B. Debraize, and E. Garrido. On Exact Algebraic (Non-)Immunity of S-boxes Based on Power Functions. In *Cryptology ePrint Archive, Report 2005/203.*
12. N. Courtois and W. Meier. Algebraic Attacks on Stream Ciphers with Linear Feedback. In *Advances in Cryptology - EUROCRYPT 2003.*
13. N. Courtois and J. Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In *Advances in Cryptology - ASIACRYPT 2002.*
14. N. Courtois, A. Shamir, J. Patarin, and A. Klimov. Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations. In *Advances in Cryptology - EUROCRYPT 2000.*
15. J.-C. Faugère and G. Ars. An Algebraic Cryptanalysis of Nonlinear Filter Generators using Gröbner Bases. In *Rapport de Recherche de l'INRIA, 2003.*
16. J. Dj. Golić. On the Security of Nonlinear Filter Generators. In *Fast Software Encryption - FSE 1996.*
17. J. Hong. Some Trivial States of Trivium. In *eSTREAM Discussion Forum, 2005.*
18. M. Krause. BDD-Based Cryptanalysis of Keystream Generators. In *Advances in Cryptology - EUROCRYPT 2002.*
19. B. Löhlein. Attacks based on Conditional Correlations against the Nonlinear Filter Generator. In *Cryptology ePrint Archive, Report 2003/020.*
20. Y. Nawaz, K. C. Gupta, and G. Gong. Algebraic Immunity of S-boxes Based on Power Mappings: Analysis and Construction. In *Cryptology ePrint Archive, Report 2006/322.*

A Algebraic and Fast Algebraic Attacks

The algebraic immunity $\mathcal{AI}$ of a Boolean function f is defined by the minimum degree d of a function g, such that $fg = 0$ or $(f+1)g = 0$. In the case $fg = 0$, one can multiply $y^t = f(L^t(x))$ by g and obtains $g(L^t(x)) \cdot y^t = 0$. For $y^t = 1$, this is an equation of degree d. Similarly, for $(f+1)g = 0$, one obtains $g(L^t(x)) \cdot (y^t + 1) = 0$. With R_A linearly independent annihilators of degree d for f and $f+1$, a single output bit can be used to set up (in average) $R_A/2$ equations in x at time t. The number of monomials in these equations is at most $D := \sum_{i=0}^{d} \binom{n}{i}$, hence by linearisation, data complexity of conventional algebraic attacks becomes about $2D/R_A$, and time complexity $C_T = D^3$.

In fast algebraic attacks, one considers equations of type $fg = h$ for $\deg h \geq \mathcal{AI}$ and $\deg g < \mathcal{AI}$. The equation $y^t = f(L^t(x))$ is multiplied by g such that $g(L^t(x)) \cdot y^t = h(L^t(x))$. One can precompute then a linear combination $\sum_i c_i \cdot h(L^{t+i}(x)) = 0$ for all t, such that $\sum_i c_i \cdot g(L^{t+i}(x)) \cdot y^{t+i} = 0$ of lower degree $\deg g$. The linear combination utilises the structure of the LFSR, and helps to cancel out all monomials of degree larger than $\deg g$. However, the equation depends on several output bits y^t. It is a special case of implicit equation, where the degree in y is 1. Depending on the degrees of g and h, time complexity can be smaller than in algebraic attacks, and data complexity is about $C_D = D + E$, where $E := \sum_{i=0}^{e} \binom{n}{i}$. This is not much larger than in algebraic attacks (with the same asymptotic complexity). See [3] for an efficient computation of annihilators and low-degree multiples.

B Experimental Setup for Filter Generators

In Tab. 4, we collect the setups of our experiments with filter generators, where n is the size of the LFSR, and k the number of inputs to the filter function. The feedback taps are chosen such that the LFSR has maximum period (*i.e.*, the corresponding polynomial is primitive), and filter taps are chosen according to a full positive difference set (*i.e.*, all the positive pairwise differences are distinct). Tap positions are counted from the left (starting by 1), and the LFSR is shifted to the right.

Table 4. Different setups for our experiments with filter generators.

Setup	n	k	feedback taps	filter taps
1	18	5	$[2, 3, 5, 15, 17, 18]$	$[1, 2, 7, 11, 18]$
2	18	5	$[1, 2, 5, 7, 9, 14, 15, 16, 17, 18]$	$[1, 3, 7, 17, 18]$
3	18	5	$[3, 5, 7, 15, 17, 18]$	$[1, 5, 8, 16, 18]$
4	18	5	$[4, 5, 6, 10, 13, 15, 16, 18]$	$[1, 6, 7, 15, 18]$
5	18	5	$[2, 3, 5, 7, 11, 15, 17, 18]$	$[1, 3, 6, 10, 18]$
6	20	5	$[7, 10, 13, 17, 18, 20]$	$[1, 3, 9, 16, 20]$
7	20	5	$[1, 2, 4, 7, 8, 10, 11, 12, 13, 15, 19, 20]$	$[1, 5, 15, 18, 20]$
8	20	5	$[2, 3, 4, 5, 6, 7, 8, 11, 13, 14, 19, 20]$	$[1, 4, 9, 16, 20]$
9	20	5	$[1, 2, 3, 4, 5, 8, 10, 11, 12, 13, 15, 17, 19, 20]$	$[1, 2, 15, 17, 20]$
10	20	5	$[1, 2, 6, 7, 9, 11, 15, 20]$	$[1, 5, 13, 18, 20]$
11	40	5	$[3, 8, 9, 10, 11, 13, 14, 15, 18, 19,$ $23, 24, 25, 26, 27, 30, 33, 34, 36, 40]$	$[1, 3, 10, 27, 40]$

C Additional Experimental Results

In Tab. 5, we present the number of conditional equations for different filters and different parameters, where the size of the LFSR is $n = 18$.

Table 5. Counting the number of linear equations R for the augmented function of different filter generators, with $n = 18$ bit input and m bit output.

Filter	m	R for setups 1-5				
CanFil1	12	0	0	0	0	0
	13	625	288	908	335	493
CanFil2	12	0	0	0	0	0
	13	144	346	514	207	418
CanFil3	12	0	0	4	0	0
	13	1272	1759	2173	2097	983
CanFil4	7	0	0	0	0	0
	8	19	4	0	0	0
	9	102	17	1	0	12
	10	533	69	9	20	167
CanFil5	6	1	0	0	0	0
	7	4	0	0	0	0
	8	15	0	0	0	1
	9	55	1	0	0	39
	10	411	61	3	0	360
	11	2142	1017	166	10	1958
CanFil6	8	0	0	0	0	0
	9	0	10	64	0	0
	10	0	97	256	0	0
	11	0	517	1024	0	0
	12	0	2841	3533	1068	0
	13	152	19531	17626	12627	9828
CanFil7	11	0	2	0	0	6
	12	68	191	36	26	178
Majority5	8	1	0	0	0	0
	9	8	3	42	27	14
	10	97	94	401	282	158