Algebraic Attacks on Stream Ciphers with Linear Feedback

Extended Version of the Eurocrypt 2003 paper, August 24, 2003

Nicolas T. Courtois[1] and Willi Meier[2]

[1] Cryptography Research, Schlumberger Smart Cards, 36-38 rue de la Princesse, BP 45, F-78430 Louveciennes Cedex, France, courtois@minrank.org

[2] FH Aargau, CH-5210 Windisch, Switzerland, meierw@fh-aargau.ch

Abstract. A classical construction of stream ciphers is to combine several LFSRs and a highly non-linear Boolean function f. Their security is usually studied in terms of correlation attacks, that can be seen as solving a system of multivariate linear equations, true with some probability. At ICISC'02 this approach is extended to systems of higher-degree multivariate equations, and gives an attack in 2^{92} for Toyocrypt, a Cryptrec submission. In this attack the key is found by solving an overdefined system of algebraic equations. In this paper we show how to substantially lower the degree of these equations by multiplying them by well-chosen multivariate polynomials. Thus we are able to break Toyocrypt in 2^{49} CPU clocks, with only 20 Kbytes of keystream, the fastest attack proposed so far. We also successfully attack the Nessie submission LILI-128, within 2^{57} CPU clocks (not the fastest attack known). In general, we show that if the Boolean function uses only a small subset (e.g. 10) of state/LFSR bits, the cipher can be broken, whatever is the Boolean function used (worst case). Our new general algebraic attack breaks stream ciphers satisfying all the previously known design criteria in at most the square root of the complexity of the previously known generic attack.

Key Words: Algebraic attacks on stream ciphers, pseudo-random generators, nonlinear filtering, Boolean functions, factoring multivariate polynomials, multivariate equations, overdefined problems, XL algorithm, ciphertext-only attacks, Toyocrypt, Cryptrec, LILI-128, Nessie.

1 Introduction

In this paper we study stream ciphers with linear feedback. In such ciphers there is a linear part, producing a sequence with a large period, usually composed of one or several LFSRs, and a nonlinear combiner that produces the output, given the state of the linear part. The security of such stream ciphers received much attention. In [14], Golic gives a set of criteria that should be satisfied in order to resist to the known attacks on stream ciphers. For example, a stream cipher should resist to the fast correlation attack [16], the conditional correlation attack [1] and the inversion attack [14]. In order to resist different types of correlation attacks, many authors focused on proposing Boolean functions that will have no good linear approximation and that will be correlation immune with regard to a subset of several input bits, see for example [6]. Recently the scope of application of the correlation attacks have been extended. In [11], the author exploits rather correlation properties with regard to a non-linear low degree multivariate function that uses all of the variables, or in other words, non-linear low degree approximations. This kind of correlations is not new, see for example in [15]. However their application to cryptographic attacks did not receive sufficient attention, probably because only recently people became aware of the existence of efficient algorithms for solving systems of nonlinear multivariate equations of low degree [25, 9–11].

Following [11], stream ciphers with linear feedback are potentially very vulnerable to such algebraic attacks. If for one state we are able, by some method, to deduce from the output, a multivariate equation of low degree in the state bits, then it is also of low

degree in the initial state bits. Then the same can (probably) be done for many other states, and given many keystream bits, we inevitably obtain a very overdefined system of equations (i.e. many equations). Such systems can be solved efficiently by techniques such as XL [25, 9], adapted for this purpose in [11] or the simple linearization [25].
In [11], the equations of low degree are obtained by approximating the non-linear component f of the cipher by a function of low degree. If the probability that the approximation holds is close to 1, then it can be used simultaneously for many equations, and we obtain efficient attacks with XL method. For example in [11], an attack in 2^{92} against Toyocrypt[1] is proposed, that requires only some 2^{19} bits of the keystream. With more keystream, and if at least some 32 bits are consecutive, a better attack is possible, due to Mihaljevic and Imai [19].
In this paper we show that algebraic attacks on stream ciphers will apply even if there is no good low degree approximation. We propose a new method of generating low degree equations, basically by multiplying the initial equations by well-chosen multivariate polynomials. This method allows to cryptanalyse a large class of stream ciphers, satisfying all the previously known design criteria. For example, all very traditional designs using only a small subset of the state bits, are shown to be insecure, whatever is the Boolean function used.
The paper is organized as follows: in Section 2 we give a general view of algebraic attacks on stream ciphers. The main component of our new attack on stream ciphers is described in Section 2.4. In Section 3.1 we overview Toyocrypt and previously known attacks, then in Section 3.2 we apply our new attack for Toyocrypt. In Sections 4 and 5 we will study LILI-128 and apply our attack. Then in Section 6 we develop our general attack on stream ciphers using a small subset of state bits. Finally we present our conclusions about the design of stream ciphers.

2 Algebraic Attacks Against Stream Ciphers

In this part we overview and substantially extend the general strategy initially described in [11], that reduces an attack on a stream cipher, to solving a system of multivariate equations.

2.1 The Stream Ciphers that May be Attacked

We consider only synchronous stream ciphers, in which each state is generated from the previous state independently of the plaintext, see for example [18] for precise definitions. In principle also, we consider only regularly clocked stream ciphers, and also (it makes no difference) stream ciphers that are clocked in a known way. However this condition can sometimes be relaxed, cf. attacks on LILI-128 described in Sections 4-5.
For simplicity we restrict to binary stream ciphers in which the state and keystream are composed of a sequence of bits and that generate one bit at a time. We also restrict to the case when the "connection function" that computes the next state is linear over $GF(2)$. We call L this "connection function", and assume that L is public, and only the state is secret. We also assume that the function f that computes the output bit from the state is public and does not depend on the secret key of the cipher. This function f is called "a nonlinear filter". The ciphers described here include the very popular filter generator, in

[1] Toyocrypt has been accepted to the second evaluation phase of the Japanese Cryptrec call for primitives, and (apparently) rejected later.

which the state of a single LFSR[2] is transformed by a Boolean function, and also not less popular nonlinear function generators, in which outputs of several LFSRs are combined by a Boolean function [18]. There are also many other ciphers that enter this scenario.
The problem of cryptanalysis of such a stream cipher can be described as follows. Let $(k_0, \ldots, k_{n-1})$ be the initial state, then the output of the cipher (i.e. the keystream) is given by:

$$\begin{cases} b_0 = f \quad (k_0, \ldots, k_{n-1}) \\ b_1 = f\,(L\,(k_0, \ldots, k_{n-1})) \\ b_2 = f\,(L^2(k_0, \ldots, k_{n-1})) \\ \quad\quad \vdots \end{cases}$$

Our problem is to recover the key $k = (k_0, \ldots, k_{n-1})$ from some subset of keystream bits b_i.

2.2 The Attack Scenario

We are going to perform a partially known plaintext attack, i.e. we know some bits of the plaintext, and the corresponding ciphertext bits. The bits don't need to be consecutive. For example if the plaintext is written in the Latin alphabet, and does not use too many special characters, it is very likely that all the characters have their most significant bit equal to 0. This will be enough for us, if the text is sufficiently long. This would be (almost) a ciphertext-only attack.
In our attacks we just assume that we have some m bits of the keystream b_i at some known positions[3].

2.3 The Summary of the Attack

For easier reading we give here a brief summary of the attack developed later. At the time t, the current keystream bit gives an equation $f(s) = b_t$ with s being the current state. The main new idea consists of multiplying $f(s)$, that is usually of high degree, by a well chosen multivariate polynomial $g(s)$, such that fg is of substantially lower degree, denoted by d. Then, for example if $b_t = 0$, we get an equation of low degree $f(s)g(s) = 0$. This in turn, gives a multivariate equation of low degree d on the initial state bits k_i. If we get one such equation for each of sufficiently many keystream bits, we obtain a very overdefined system of multivariate equations that can be solved efficiently.
In the following Section 2.4 we describe in more details known and new methods to obtain overdefined systems of equations from a stream cipher and explain how to solve them.

2.4 Design Criteria on f and Known Attacks

Let f be the Boolean function [4] that is used to combine the outputs of the linear part of the cipher. For example the inputs to the function are some bits of the state of some LFSRs. The usual requirements on such functions can be summarised as follows. First, f should be balanced and have high algebraic degree. To prevent correlation attacks, f should be highly non-linear[5], and correlation immune at high order, see [6].

---

[2] A Linear Feedback Shift Register, see e.g. [18]. It is also possible to use a MLFSR, equivalent in theory [19] but having faster diffusion, as used in Toyocrypt cipher that we study later.

[3] For unknown positions, it would give no information whatsoever.

[4] In this paper we study mainly attacks on ciphers using one single Boolean function f, i.e. with an output in $GF(q)$ with $q = 2$. It is straightforward to extend them to the case when $q > 2$ or with an output in some ring. It is also straightforward to extend them to the case of several different filtering functions combined in parallel.

[5] But maybe not a perfectly non-linear (bent) function, see Section 4 in [11].

2.5 Criteria on f in the Recent Algebraic Attack on Stream Ciphers

An additional criterion on f is given in [11]: f should also not have a very good correlation with respect to low-degree non-linear multivariate functions, otherwise efficient attacks are possible, for example for Toyocrypt [11]. They are possible when:

S1 either the Boolean function f has a low algebraic degree D (classical criterion),

S2 or f can be approximated by such a function with a probability close to 1 (new criterion [11]). This probability is usually denoted $1-\varepsilon$ for some small ε.

The practical attacks on Toyocrypt presented in [11] use the second case S2. In this paper we use equations true with probability 1 (as in the first case) except that we relax the degree condition: it is no longer necessary that f has a low algebraic degree d:

2.6 New Assumptions on f and New Attack Scenarios

S3 The multivariate polynomial f has some multiple fg of low degree d, with g being some non-zero multivariate polynomial.

S4 It is also possible to imagine attacks in which f has some multiple fg, such that fg can be approximated by a function of low degree with some probability $(1-\varepsilon)$.

How to Use Low Degree Multiples

Our goal is to obtain an overdefined system of low degree equations. In scenarios S1 and S2, for each known keystream bit at position t: b_t, we obtain a concrete value of $b_t = f(s)$, and thus we get the equation $b_t = f\left(L^t(k_0,\ldots,k_{n-1})\right)$. For this, f has to be of low degree. In scenarios S3 and S4, for each known keystream bit $b_t = f(s)$ at position t, we get:

$$f(s)\cdot g(s) = b_t \cdot g(s),$$

and, since the state is at time t is $s = L^t(k_0,\ldots,k_{n-1})$, it boils down to:

$$f\left(L^t(k_0,\ldots,k_{n-1})\right)\cdot g\left(L^t(k_0,\ldots,k_{n-1})\right) = b_t \cdot g\left(L^t(k_0,\ldots,k_{n-1})\right).$$

This is the equation we are going to use in our attack(s). We get one multivariate equation for each keystream bit. This equation may be of very low degree, without f being of low degree, and without f having an approximation of low degree.

In the basic version of this attack S3, we also require that g is of low degree. There are other possibilities. In the basic version of the attack S3, that may be called S3a, we use the equation written above and we require the $fg \neq 0$ and fg is of low degree, and also we need g of low degree. There is another variant, in which we may admit that $\forall s\ f(s)g(s) = 0$, and the equation can still be used when $b_t \neq 0$. This is called the scenario S3b. Another variant, called S3c, allows to relax the degree condition on g: when $b_t = 0$, we can still use the equation, whatever is the degree of g, provided that $fg \neq 0$ and is of low degree. All the 3 sub-cases of the S3 attack scenario are summarized in the following table[6]:

[6] In this table, we see that S2 is obtained from S1 by adding "true with probability $(1-\varepsilon)$" everywhere. The same applies to S4 defined with respect to S3.

Attack scenario considered	Degree of f	Degree of g	Degree of fg	Use the equation	Only when	Number of equations for m keystream bits
S1 and[6] S2	low	$g = 1$	low	$f(s) = b_t$	always	m
S3a and[6] S4a	high	low, $g \neq 0$	low, $fg \neq 0$	$f(s) \cdot g(s) = b_t \cdot g(s)$	always	m
S3b and[6] S4b	high	low, $g \neq 0$	$fg = 0$	$g(s) = 0$	$b_t \neq 0$	$m/2$
S3c and[6] S4c	high	high	low, $fg \neq 0$	$f(s) \cdot g(s) = 0$	$b_t = 0$	$m/2$

Table 1. Different methods to obtain low degree equations from keystream bits

The important question is now, given a cipher, whether such polynomials g exist, and how to find them. For this, see Section 5 and 6.

Remark 1: It can be seen that the scenarios S1 (respectively S2) are a special case of the scenarios S3 (respectively S4). Indeed, if we put $g = 1$, the equation used in scenarios S3/S4 becomes the usual equation $f(s) = b_t$ of the previous scenarios S1/S2.

Example: In the scenario S3b, if g is a non-zero polynomial of low degree $d \in \mathbb{N}$ such that $fg = 0$, then we will obtain one equation $g(s) = 0$ of the same degree D for each of the keystream bits for which $b_t = 0$, i.e. given m keystream bits, we will obtain on average $m/2$ multivariate equations of degree D.

Remark 2: The equations obtained for each keystream bit b_t, from the 3 scenarios S3a-c may overlap: i.e. become linearly dependent. For this reason we will rather use only the equations produced by one of these scenarios, full version except for the scenarios S3b and S3c can be combined safely: only one of them is used, for each keystream bit b_t, depending whether $b_t = 0$ or 1.

2.7 Solving Overdefined Systems of Multivariate Equations

In our attack, given m keystream bits, let R be the number of multivariate equations of degree d, and with n variables k_i, that we obtain. With one equation, and in scenario S3a, we have $R = m$, but we may also combine several scenarios and several different g for the same f, and get, for example $R = 14 \cdot m$. We solve them as follows.

Linearization Method: There are about $T \approx \binom{n}{d}$ monomials of degree $\leq d$ in the n variables k_i (assuming $d \leq n/2$). We consider each of these monomials as a new variable V_j. Given $R \geq \binom{n}{d}$ equations, we get a system of $R \geq T$ linear equations with $T = \binom{n}{d}$ variables V_i that can be easily solved by Gaussian elimination on a linear system of size T.

XL Method: When as many as the required $m = \mathcal{O}(\binom{n}{d})$ keystream bits, are **not** available, it is still possible to use XL algorithm or Gröbner bases algorithms to solve the system, with less keystream bits, but with more computations, see [11] or Appendix D.1.

2.8 About the Complexity of Gaussian Reduction

Let ω be the exponent of the Gaussian reduction. , i.e. a linear system with T variables can be solved in time T^ω. In theory it is at most $\omega \leq 2.376$, see [7]. However the (neglected) constant factor in this algorithm is unknown to the authors of [7], and is expected to be very big. The fastest practical algorithm we are aware of, is Strassen's algorithm [30] that requires about $7 \cdot T^{log_2 7}$ operations. Since our basic operations are over $GF(2)$, we expect that a careful bitslice implementation of this algorithm on a modern CPU can handle 64 such operations in one single CPU clock. To summarize, in this paper we assume that the Gaussian reduction takes $7 \cdot T^{log_2 7}/64$ CPU clocks.

3 Cryptanalysis of Toyocrypt

3.1 Toyocrypt and Recent Algebraic Attacks

We look at the stream cipher called Toyocrypt, a submission to the Japanese government Cryptrec call for cryptographic primitives. It was, at the time of the design, believed to resist to all known attacks on stream ciphers. In Toyocrypt, we have one 128-bit LFSR, and thus $n = 128$. The Boolean function is of the form:

$$f(s_0, .., s_{127}) = s_{127} + \sum_{i=0}^{62} s_i s_{\alpha_i} + s_{10}s_{23}s_{32}s_{42} +$$
$$+ s_1 s_2 s_9 s_{12} s_{18} s_{20} s_{23} s_{25} s_{26} s_{28} s_{33} s_{38} s_{41} s_{42} s_{51} s_{53} s_{59} + \prod_{i=0}^{62} s_i.$$

with $\{\alpha_0, \dots, \alpha_{62}\}$ being some permutation of the set $\{63, \dots, 125\}$. This system is quite vulnerable to an attack using low-order approximations: there is only one monomial of degree 17, and one of degree 63. The higher-order monomials are almost always zero.

The Recent Higher Order Correlation Attack on Toyocrypt

This attack is described in [11] an attack following the scenario S2 is described. In this attack, f is approximated by a multivariate function of degree 4 with probability $1 - 2^{-17}$. The attack runs in 2^{92} CPU clocks. requires 2^{65} bits of memory, and and has very loose requirements on the keystream needed: only 51 kilobytes, that does not have to be consecutive. For more details see [11]. In this paper we present a much faster attack on Toyocrypt.

3.2 Our New Algebraic Attack on Toyocrypt

In our attack we need to find g such that fg is of low degree (or is such with high probability), following the assumption S3 (or S4) from Section 2.4. How do we find a function g such that fg is of low degree ? One method we present in this paper, is by factoring multivariate polynomials. We consider the terms of high degree in $f(s)$ (regardless the lower degree terms) and look if they are divisible by a common low degree factor $g'(s)$. Then (for polynomials over $GF(2)$) we observe that $f(s) \cdot g(s)$ with $g(s) = g'(s) - 1$ is of low degree.

Remark: Compared to the previous algebraic attacks on Toyocrypt from [11], that work for any field $GF(q)$, this approach seems to work well only for ciphers defined over $GF(2)$. Later in Sections 5 and 6, we will describe a different method to find polynomials g satisfying our requirements.

In the case of Toyocrypt, we observe that the combination of the parts of degree 4, 17 and 63, is divisible by a common factor $s_{23}s_{42}$. For each clock t, s_{23} and s_{42} are different known linear combinations of the k_i. For each clock t and the corresponding keystream bit b_t, we start from the equation $f(s) = b_t$, and multiply both sides of it by $g(s) = (s_{23} - 1)$. Then we get $f(s)s_{23} - f(s) = b_t(s_{23} - 1)$. The monomials divisible by s_{23} in f will cancel out, and what remains is an equation of degree 3 true with probability 1. We repeat the same trick for s_{42}, i.e. we put $g(s) = (s_{42} - 1)$. From this, we have a simple linearization attack following the scenario S3a. For each keystream bit, we obtain 2 equations of degree 3 in the s_i, and thus 2 equations of degree 3 in the k_i. The linearization will work as soon as $R > T$, and we have $T \approx \binom{128}{3} = 2^{18.4}$ monomials. We have $R = 2m$, and having $m = T/2 = 2^{17.4}$ keystream bits will be sufficient. Our new attack on Toyocrypt is in

$7/64 \cdot T^{log_2 7} = \mathbf{2^{49}}$ CPU clocks, requires 16 Gigabytes of memory and only about 20 kilobytes of (non-consecutive) keystream.

We programmed this attack to see if it works. Our simulations show that the number of linearly dependent equations is negligible and therefore the attack will be able to recover the key. Details are given in the Appendix A.

Comparison with Other Attacks

This attack is much better than the general purpose time/memory/data tradeoff attack described by Shamir and Biryukov in [24], that given the same number of keystream bits, about 2^{17}, will require about 2^{111} computations (in precomputation phase). Our attack is also always better than the Mihaljevic and Imai attack from [19]. For example in [19], given much more data, for example 2^{48} bits, and in particular at least some 32 consecutive bits of the keystream, and given much more memory 2^{64}, the key can be recovered with a precomputation of 2^{80} and processing time in 2^{32}. Moreover if the keystream does not contain 32 consecutive bits, only our attack will work. Similarly, if the keystream available is limited to 2^{17}, both the attack from [19] and [24] will require a precomputation of about 2^{111}.

4 Background on LILI-128 and Simple Attacks

In principle our algebraic attacks are designed only for regularly clocked stream ciphers (or ciphers clocked in a known way). However in some cases, this difficulty can be removed. This is the case for LILI-128, a submission to Nessie European call for cryptographic primitives.

4.1 Eliminating the First Component

LILI-128 is a stream cipher composed of two LFSR-based filter generators, the first being used to clock the second. There are two basic strategies to by-pass this.

A. Since the key length of the first component is only 39 bits, we may guess these 39 bits and attack the second component alone. In LILI-128, the first component advances the clock of the second component by 1, 2, 3 or 4. Given the state of the first component, we have access to some number of non-consecutive keystream bits of the second component, at known positions. This is sufficient for our attacks, and the complexity of the attack is multiplied by 2^{39}.

B. Given more keystream bits, it is possible to avoid repeating the whole attack 2^{39} times. For this, we use the Lemma 1 from [28]: after clocking the first LFSR of LILI-128 $2^{39}-1$ times, the second LFSR advances exactly $\Delta_d = 5 \cdot 2^{38} - 1$ times. Thus, we may, instead of guessing the state of the clock control subsystem, clock it $2^{39}-1$ at a time, and apply any of the XL attacks exactly as if the first generator did not exist.

In both cases, the bits the attacker has access to, are in some known places of the keystream of the second component (but not in chosen places). It is perfectly sufficient to apply directly, to the second component, all the algebraic attacks S1-S4 described in Section 2.4: 2.2-2.7, From the point of view of all our attacks, that write one equation for each keystream bit, the second component is attacked exactly as if it was a stand-alone filter generator: we will have access to some keystream bits at some known positions. This will give a system of equations we will solve.

Intermediate Solutions A-B. The period of the first component of LILI-128 is not a prime, and we have $2^{39} - 1 = 7 \cdot 79 \cdot 121369 \cdot 8191$. This suggests that one should be able to design a decimation attack, in which by clocking the generator t clocks at a time, for a suitable t, one could simulate a smaller LFSR, see [13] for more details. It could give a version of our later attacks, intermediate between A and B, in which both the keystream requirements and the attack complexity would be multiplied by some factor, but both factors should be smaller than 2^{39}. One should refer to [13] for more details on decimation.

The Boolean Function Used in LILI-128

We call f the output filtering function of LILI-128 (called f_d in [26]). It is a highly non-linear Boolean function of degree 6, with 10 variables, built following the paper [22]. It uses a subset of 10 variables:

$$(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) \stackrel{def}{=} (s_0, s_1, s_3, s_7, s_{12}, s_{20}, s_{30}, s_{44}, s_{65}, s_{80}).$$

We computed the algebraic normal form (ANF) of this LILI-128 function f. It gives:

$$\begin{aligned} f(x_1, .., x_{10}) = \\ x_2 + x_3 + x_4 + x_5 + x_6x_7 + x_1x_8 + x_2x_8 + x_1x_9 + x_3x_9 + x_4x_{10} + x_6x_{10} + \\ x_3x_7x_9 + x_4x_7x_9 + x_6x_7x_9 + x_3x_8x_9 + x_6x_8x_9 + x_4x_7x_{10} + x_5x_7x_{10} + \\ x_6x_7x_{10} + x_3x_8x_{10} + x_4x_8x_{10} + x_2x_9x_{10} + x_3x_9x_{10} + x_4x_9x_{10} + x_5x_9x_{10} + \\ x_3x_7x_8x_{10} + x_5x_7x_8x_{10} + x_2x_7x_9x_{10} + x_4x_7x_9x_{10} + x_6x_7x_9x_{10} + x_1x_8x_9x_{10} + x_3x_8x_9x_{10} + \\ x_4x_8x_9x_{10} + x_6x_8x_9x_{10} + x_4x_6x_7x_9 + x_5x_6x_7x_9 + x_2x_7x_8x_9 + x_4x_7x_8x_9 + \\ x_4x_6x_7x_9x_{10} + x_5x_6x_7x_9x_{10} + x_3x_7x_8x_9x_{10} + x_4x_7x_8x_9x_{10} + x_4x_6x_7x_8x_9 + x_5x_6x_7x_8x_9 + \\ x_4x_6x_7x_8x_9x_{10} + x_5x_6x_7x_8x_9x_{10} \end{aligned}$$

4.2 First Attacks on LILI-128 (Scenarios S1 and S2)

First, it is possible to apply to LILI-128 our scenario S1. We have $d = 6$, $\varepsilon = 0$ and $R = m$. Then $T \approx \binom{89}{6} \approx 2^{29.2}$ monomials, and in order to have $R > T$ we will need $m \approx 2^{29.2}$. It gives an attack in[7] about:

$$2^{39} \cdot 7/64 \cdot T^{\log_2 7} \approx 2^{118} < 2^{128}.$$

Remark: This is in fact a degenerated version of the XL attack with $D = d$, (known also as linearization [25]), and if $\varepsilon = 0$, there is no use take in XL a bigger D than the minimum $D = d = 6$. However, as always, if there is not enough keystream bits to perform this attack, XL may be used, see [11] and Appendix D of this paper.

Given the Boolean function used, it is not useful to extend the attack to the scenario S2: there is no good approximation of degree $d < 6$, as it would gives $\varepsilon > 2^{-6}$ which is by far too big: with LILI-128, an approximation of degree 5 holds only for about 2^6 keystream bits. There is no real possibility to get an overdefined system of equations, for example the probability to obtain a system of 128 equations of degree 5 that are correct is $(1-2^{-6})^{128} \approx 0.13$, and to obtain $m = 2^{12}$ equations it is about 2^{-93}. We cannot hope to obtain an overdefined system: for systems of degree 5, we need m of order of $\binom{128}{5} \approx 2^{28}$.

However we may improve the attack following the method B from Section 4.1. Clocking the first LFSR of LILI-128 $2^{39} - 1$ times advances the first LFSR exactly $5 \cdot 2^{38} - 1$ times, exactly as if the first generator did not exist. Thus instead of guessing the state of the

[7] This simple attack has already been described by Steve Babbage in [4].

clock control subsystem we clock it $2^{39}-1$ at a time and apply the simple linearization S1-type attack with $D=6$ and $\varepsilon=0$. Now the complexity is only $\frac{7}{64}T^{\log_2 7}\approx 2^{79}$ but this version requires much more keystream bits: 2^{68} instead of 2^{29} bits.

5 Better Attacks on LILI-128

First we try to apply to LILI-128 the idea of factoring multivariate polynomials from Section 3.2. For this, we consider the part of degree 5 and 6 of f. We used Maple function `factor()` to factor this multivariate polynomial. It gives:

$$x_7x_9\left(x_3x_8x_{10}+x_4x_6x_8+x_4x_6x_{10}+x_4x_8x_{10}+x_5x_6x_8+x_5x_6x_{10}+x_4x_6x_8x_{10}+x_5x_6x_8x_{10}\right)$$

It means that when we multiply f by either (x_7+1) or (x_9+1), the degree collapses from $6+1=7$ to $4+1=5$. We consider then the factoring of the part of degree 5 and 4 of respectively $f(x)\cdot(x_7+1)$ and $f(x)\cdot(x_9+1)$. Only the second function can still be factored and it gives:

$$x_{10}\left(x_3x_7x_8x_9+x_5x_7x_8x_9+x_3x_7x_8+x_3x_8x_9+x_4x_7x_9+x_4x_8x_9+x_5x_7x_8+x_5x_7x_9+x_6x_7x_9\right)$$

From this we deduce the remarkable fact that $f(x)(x_9+1)(x_{10}+1)$ is of degree **4**, instead of 8. We have done computer simulations to see if more low degree multiples of f exist.

5.1 Finding More Low Degree Multiples of f for LILI-128

We are trying to mount an attack following the scenario S3a or S3c, i.e. we are looking for the number of linearly independent polynomials g, such that fg is of low degree. In order to find them, we are looking for linear dependencies in the following set of multivariate polynomials (stopped at maximum degree for g and some maximum degree for fg).

$$\{f(x),f(x)\cdot x_1,f(x)\cdot x_2,\ldots,f(x)\cdot x_1x_2,\ldots;\quad 1,x_1,x_2,\ldots,x_1x_2,\ldots\}$$

We do not count polynomials g for which fg is a polynomial of low degree equal to 0. Results on such equations, corresponding rather to the scenario S3b, are given later in Section 5.2. We note that the maximum degrees cannot be higher than 10, as there are only 10 variables. Here are our results with $fg\neq 0$, (a.k.a. Scenario S3a) compared to a random Boolean function of the same size:

The function	LILI-128 f_d						Random Boolean					
Degree of g	10	1	**2**	3	4	10	10	1	2	3	4	10
Degree of fg	3	4	4	4	4	4	3	4	4	4	4	4
Nb. of g	0	0	**4**	8	**14**	14	0	0	0	0	0	0

Table 2. Simulations on the number of linearly independent g such that fg is of low degree

We have computed and tested all these solutions. For example, one can verify that:

$$f(x)\cdot x_8x_{10}=x_8x_{10}\left(x_2x_9+x_3x_7+x_4x_7+x_5x_9+x_1+x_4+x_5+x_6\right).$$

We see that the function f of LILI-128, behaves much worse than a random Boolean function. This shows that the design of LILI-128 is far from being optimal against algebraic attacks.

Remark on Scenario S3c vs. S3a: From our simulations we see that there is no more g of degree 10 with fg of degree 4, than when g are of degree ≤ 4. For this, particular function f, and for $d=4$, the scenario S3c cannot be used.

5.2 Finding Multiples $fg = 0$ with Low Degree of g (scenario S3b)

We may also try to mount an attack following the scenario S3b, i.e. we are looking for the number of linearly independent polynomials g of low degree, such that $fg = 0$.
We are interested in the total number of such g that are linearly independent. Here are the results of our simulations: Our simulations show that surprisingly many such orthogonal factors exist. In the following table we compare the number of such linearly independent g found for f used in LILI-128, compared to a random Boolean function $GF(2)^{10} \to GF(2)$.

The function	LILI-128 f_d						Random Boolean					
Given max. degree	1	2	3	4	5	6	1	2	3	4	5	6
Nb. of g	0	0	0	**14**	149	350	0	0	0	0	144	337

Table 3. S3b: Simulations on the number of linearly independent low degree g such that $fg = 0$

Unfortunately, in the scenario S3b, we did not find any g of degree 3, however we did find 14 linearly independent g of degree 4. For example, one can verify the following two examples:

$$f(x) \cdot x_{10} \left(x_1x_7x_8 + x_3x_7x_8 + x_5x_7x_8 + x_2x_7x_9 + x_3x_7x_9 + x_4x_7x_9 + x_5x_7x_9 + x_6x_7x_8 + x_2x_8x_9 + x_3x_8x_9 + x_1x_9 + x_3x_9 + x_4x_9 + x_6x_9 + x_7x_8 + x_9\right) = 0$$

$$f(x) \cdot x_8x_{10} \left(x_2x_9 + x_3x_7 + x_4x_7 + x_5x_9 + x_1 + x_4 + x_5 + x_6 + 1\right) = 0$$

Remark: The several g functions obtained here (scenario S3b) are not necessarily linearly independent of those of Section 5.1 (scenario S3a). Indeed, from the definitions of S3a and S3b we see that a linear combination of g that satisfy S3a can be in S3b. For this reason, we will not try to combine these two attacks from Sections 5.1 (scenario S3a) and 5.2 (scenario S3b).
LILI-128 vs. Random Boolean Functions: In comparison to random, the function f of LILI-128, behaves, again, worse than a random Boolean function. This (again) shows that the design of LILI-128 is not optimal.
Attacks for Any Boolean Function ?! However, in this (second) attack on LILI-128, we observe that even for a random Boolean function with 10 variables, there are still solutions g of degree 5. It can be seen that it is due to a small number of variables (10). From this, in Section 6 we develop a general attack on stream ciphers, whatever is the Boolean function used.

Consequences for LILI-128

Following the scenario S3a, and using the results of Section 5.1, given m keystream bits, we obtain $14 \cdot m$ multivariate equations of degree 4 in the key bits k_i of LILI-128. We will have to solve an overdefined system of multivariate equations of degree 4, true with probability 1. This is done by linearization. Following Section 4.1 there are two versions of the attack.

A In the first version (A) the state of the first generator is guessed and the complexity is multiplied by 2^{39}. For each keystream bit we obtain 14 equations of degree 4 in the k_i. For linearization we have $T = \binom{89}{4} = 2^{21}$ monomials, and we will need $m = T/14 = 2^{18}$ keystream bits in order to have $R > T$. Our best new attack on LILI-128 requires then $2^{39} \cdot 7 \cdot T^{log_2 7}/64 \approx 2^{96}$ CPU clocks. This first attack version works given access to some m stream bits, being at some known positions.

B In the second version (B), the first component is clocked $2^{39}-1$ clocks at a time (see Section 4.1) and thus the number of keystream bits is multiplied by 2^{39}. We have the same $T = \binom{89}{4} = 2^{21.2}$, and the complexity is now $\mathbf{2^{57}}$ CPU clocks. The attack requires 762 Gigabytes of memory and 2^{57} bits of consecutive keystream, or, it can be seen that we only need 2^{18} keystream bits that are on some positions of the form $\alpha + \beta \cdot (2^{39}-1)$, for a fixed α. Once the state at the time α of the second LFSR is recovered, the state of the first LFSR can be found very quickly within 2^{39} tries and a few additional keystream bits. The values of β does not have to be consecutive.

Remark: This is not the best attack known for LILI-128. In [28] it is shown that LILI 128 can be broken with 2^{46} bits of keystream, a lookup table of 2^{45} 89-bit words and computational effort which is roughly equivalent to 2^{48} DES operations. Our attack has however much more general consequences.

6 General Attack on Stream Ciphers Using a Subset of LFSR bits

In this section we show that the case of LILI-128 is not isolated. We will show that all very traditional stream ciphers, with linear feedback and a highly non-linear (stateless) filtering function are insecure, for example[8] when they use only a small subset of state bits.

We consider a stream cipher with n state bits, and using only a small subset of k state bits to derive the keystream bit. Thus we have:

$$\{x_1, x_2, \ldots, x_k\} \subset \{s_0, s_1, \ldots, s_{n-1}\}\,.$$

We assume that k is a small constant and n is the security parameter. For example for the second component of LILI-128 $k = 10, n = 89$. A good stream cipher design should have security that grows exponentially in n, for example as $2^{2n/3}$, see [24], as long as the amount m of stream bits available, is not too big.

We would like to mount an attack following the scenario S3a, and for this, we are looking for the number of linearly independent low-degree polynomials $g \neq 0$, such that fg is also of low degree. In order to find them, similarly as in Section 5.1, we check for linear dependencies in the set of polynomials $C = A \cup B$ defined below as follows.

In this proof we consider multi-sets (i.e. repetitions are allowed). First, we consider all the possible monomials up to some maximum degree d (this part will later compose fg).

$$A = \{1, x_1, x_2, \ldots, x_1x_2, \ldots\}$$

Then we consider all multiples of f, multiplied by monomials of the degree up to d (this degree corresponds to the degree of g). Thus we write the following polynomials:

$$B = \{f(x), f(x) \cdot x_1, f(x) \cdot x_2, \ldots, f(x) \cdot x_1x_2, \ldots\}$$

Let $C = A \cup B$. All elements of A,B and C, can be seen as multivariate polynomials in the x_i: for this we need to substitute f with its expression in the x_i. A set of multivariate polynomials with k variables cannot have a dimension greater than 2^k. If there are more than 2^k elements in our multi-set, linear dependencies will exist. Such combinations allow to find a function g such that $f \cdot g$ is of substantially lower degree than f. More precisely we have the following theorem:

[8] Though Toyocrypt does not satisfy this assumption, it is still broken by our attack, that will also work in many other interesting cases, see Section 7.

Theorem 6.0.1 (Low Degree Relations).
Let f be any Boolean function $f : GF(2)^k \to GF(2)$. Then there is a Boolean function $g \neq 0$ of degree at most $\lceil k/2 \rceil$ such that: $f(x) \cdot g(x)$ is of degree at most $\lfloor k/2 \rfloor$.

Proof: If we include in A all the monomials with degrees up to $\lfloor k/2 \rfloor$, and for B, we multiply f by all the monomials with degrees up to $\lceil k/2 \rceil$, then we have:

$$|C| = |A| + |B| = \sum_{i=0}^{\lfloor k/2 \rfloor} \sum_{i=0}^{\lceil k/2 \rceil} \binom{k}{i} + \sum_{i=0}^{\lceil k/2 \rceil} \binom{k}{i} = \sum_{i=0}^{k} \binom{k}{i} + \binom{k}{\lfloor k/2 \rfloor} > 2^k$$

The rank of $C = A \cup B$ cannot exceed 2^k. Since $|C| > 2^k$, some linear dependencies must exist. Moreover there are no linear dependencies in the part A of our multi-set C, and therefore all linear dependencies must combine either only the elements of B, or both A and B, which in turn means that $g \neq 0$. This ends the proof. □

Remark: If $fg \neq 0$, we obtain an attack following the scenario S3a or S3c, otherwise when $fg = 0$ we obtain an attack following the scenario S3b.

Consequences of the Theorem

We see that for any stream cipher with linear feedback, for which the non-linear filter uses k variables, it is possible to generate at least one equation of degree $\lceil k/2 \rceil$ in the n keystream bits. These equations will be solved, as usual, by linearization. We will need at most $\sum_{i=0}^{\lceil k/2 \rceil} \binom{n}{i} \approx \binom{n}{\lceil k/2 \rceil}$ keystream bits in order to obtain a complete saturated system solvable by linearization. To summarise, we get the following general attack for any Boolean function f with k inputs.

d	$\lceil k/2 \rceil$
ε	0
Data	$\binom{n}{\lceil k/2 \rceil}$
Memory	$\binom{n}{\lceil k/2 \rceil}^2$
Complexity	$\binom{n}{\lceil k/2 \rceil}^\omega$

The Complexity of the Attack. This attack is polynomial when k is fixed. This attack will only be exponential in n, if $k = \mathcal{O}(n)$. This means that, in order to achieve exponential security, The number of bits used in a non-linear filter can**not** be small.
In practice, talking about polynomial (or not-polynomial) time is misleading and should always be confronted with concrete results. Knowing that the maximum degree of the filtering function cannot exceed k, it can be seen that in the scenario S1, any stream cipher with linear feedback can be broken in about $\binom{n}{k}^\omega$, given $\binom{n}{k}$ keystream bits, by simple linearization. This simple attack is already polynomial when k is fixed, and well known, see for example [14] or [4]. Here precisely is the problem: many stream ciphers, some of them unpublished and proprietary, have been designed in such a way that, one has for example $\binom{n}{k}^\omega \approx 2^{80}$. In practice, since our complexity $\binom{n}{\lceil k/2 \rceil}^\omega$ is, very roughly the square root of the previous one, we can break all these ciphers in roughly 2^{40}.
Improving the Keystream Requirements. If for a given f, there are several linear dependencies in C, we will be able to use several linearly independent g, and for each keystream bit, we will obtain several equations. Then the keystream requirements will be divided accordingly. For example in Section 5.2 they are divided by 14.
About the Scenario S3c: We were able to show that, either the scenario $S3a$ or $S3b$ works for the degree $\lceil k/2 \rceil$. For some specific functions, the scenario S3c may be much

more powerful: if r' is very small, then we will obtain an attack following the scenario S3c, in which the degree of g is indeed high, but the degree of fg is very small. For example if $r' < n^2/2$, we obtain quadratic equations, and the cipher would be broken in $n^{2\omega}$ given only about $m = n^2/2$ keystream bits.

Worse Case vs Practice. This attack deals with the worst case. For specific functions the cipher may be much less secure. For example in LILI-128, $k = 10$ and, with a strict application of Theorem 6.0.1 we obtain the worst case complexity $\mathcal{O}(n^{5\omega})$ for any Boolean function. However, for the specific function used in LILI-128, our attack from Section 5.2 is in $\mathcal{O}(n^{4\omega})$.

Moreover, in many cases it will be sufficient to consider the case when the degrees are such that we have $|C| = |A| + |B| = 2^k$. In this case one linear dependency will still exist with a non-negligible probability.

Another Remark: In Section 5.2 we see that for a random Boolean function (or at least for most of them), there will be an equation of degree 5, as predicted by the Theorem, but with $fg = 0$. This is completely unexpected from the Theorem (because if $|A| = 1$, the degree should be in principle much bigger to have $|C| > 2^k$, but is confirmed by our simulations. This suggests that there might be a better Theorem on the average behaviour of the Boolean functions.

Resistance Criteria Against This Attack: By inspection we verify that the requirement to achieve the best possible resistance against our attack is the following. We need to make sure that no linear dependencies exist when both multi-sets A and B are generated up to any degree, strictly smaller than the degree for which dependencies must exist, due to the theorem. It can be seen that it boils down to assuring that:

Optimal Resistance Criterion: Each time that A and B are generated with degrees in the x_i such that their sum is exactly $k-1$, and consequently $|C| = 2^k$, all the equations in C should be linearly independent. It is easy to show that this criterion implies that the degree of f will be sufficiently large to prevent the attack S1. However, it cannot guarantee that attacks of type S2 (as in [11]) or S4, will not exist.

7 Consequences for the Design of Stream Ciphers

There are many interesting cases in which the attacks described in this paper will work. For example, it can be seen that they will work for any regularly clocked stream cipher with linear feedback and any Boolean function f such that:

1. either f uses a small subset of state bits, (e.g. 10), as in LILI-128, see Section 6,
2. or is very, very sparse, as in Toyocrypt, see [11],
3. or can be factored with a low degree factor, as in Toyocrypt, see Section 3.2.
4. or can be approximated by one of the above (see e.g. our Scenario S4),
5. or its part of high degree is one of the above, for example in Section 5 it has low degree factors.

We conclude that, in a stream cipher with linear feedback, the filtering function should use many state bits, for example at least[9] 32, and should not be too sparse, so it has also many terms of very high degree (for example at least 32). Moreover the part of high

[9] However, a too large number of state bits used in the filter function may conflict with certain design criteria introduced in [14] to render inversion attacks infeasible.

degree, should not have a low degree factor, and should itself also use many state bits. Then, no approximation of the part of high degree (for example obtained by removing or adding a few very high degree terms) should have a low degree factor, or should use a small number of state bits. Finally, we see in the example of LILI-128, that specific functions may behave worse than a random Boolean function of the same size, for no apparent reason.

What Happened to Stream Ciphers Using Boolean Functions ? In this paper we show that using LFSRs and so called "good" Boolean functions is by far insufficient to construct secure stream ciphers. Regularly clocked filter generators, and the nonlinear function generators (combinatorial function generators), fail to deliver the security claimed by some authors [13]. Algebraic attacks open multiple avenues for further research (see also the generalisations below). Recently, a filter combiner model for memoryless synchronous stream ciphers has been proposed in [23]. This model allows for more freedom to simultaneously satisfy various design criteria (e.g., correlation attacks and inversion attacks can be avoided by a suitable choice of parameters). However, this model still uses a stateless Boolean function to combine outputs of a few linear finite state machines (e.g., LFSRs or linear cellular automata). Hence our attacks are in principle also valid for the Boolean functions used in this recent proposal.

New Design Criteria for Boolean Functions. It can be seen that the attacks described in the present paper are possible when there exist h, g such that $fg + h = 0$, with either h is of low degree, or $h = 0$ and g is of low degree. In both cases the degree of fg becomes small. Hence, ideally, to resist algebraic attacks, given the function f, for every function g of "reasonable" size the degree of fg should always be lower-bounded by a sufficiently large degree (for example 10).

This is the new design criterion we propose for Boolean functions used in stream ciphers.

Remarks:

1. It is a bit imprecise. If the degree of fg is always at least 6, and when $n = 128$, we will have an attack in about 2^{88}. If the degree of fg is always at least 10, and when $n = 128$, the complexity of the attacks described in the present paper will exceed 2^{128}. For more detailed recommandations one needs to study the exact complexity of the attacks we described in this paper including possible variants and improvements.
2. If we require that the degree of fg is at least 10, than from Theorem 6.0.1, it implies that k should be at least 20. At any rate, $2 \cdot 6 = 12$ is a strict minimum. To have 20 inputs is not easy to satisfy in practical stream ciphers and the Boolean function cannot easily be stored as a table, as in LILI-18 [26]. This gets even much worse if we look at the improved attack described in Section 7.2. of [12]: from this attack we get that even $k = 50$ may not be enough for $n = 128$ and the security level of 2^{128}. See [12] for more details.
3. We would say that f is an "optimal Boolean function w.r.t. the attacks described in this paper", if for every function g the degree of fg should be always be at least $\min(\lfloor k/2 \rfloor, deg(f) + deg(g))$. Indeed, following Theorem 6.0.1, in general we cannot hope that it will be always bigger than $\lfloor k/2 \rfloor$.

7.1 General Algebraic Attacks and New Design Criteria for Stream Ciphers

More generally, if there are several filtering functions f_i in a stream cipher, there should be no algebraic combination of the f_i and of "reasonable" size, that would have an unusually low degree. By extension, this criterion also applies to stream ciphers that have only one filtering function. Indeed a cipher having only one filtering function f, can be seen as using several functions defined as: $f, f \circ L, f \circ L^2, \ldots$. It can be seen that, in all cases, our security criterion can be re-formulated as: there should be no non-trivial multivariate relations of low degree that relate the key bits and one or many output bits of the cipher. Otherwise, if only one such multivariate relation exists (for any reason), an algebraic attack as described in this paper will be possible. We may call this Scenario S5.

It can be seen that we obtain a design criterion that is basically identical to the notion of non-trivial equations defined in Section 2 of [8]. It is also very similar to the design criterion given in [10] for the S-boxes of block ciphers. Finally, it can also be seen as an interpretation of Shannon's prescription: In the famous paper from 1949, Claude E. Shannon states that breaking a good cipher should require "as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type", see [29].

Extension to Stateful Combiners It is important to see that our generalized attack scenario S5 applies potentially to all ciphers with linear feedback, even for filters with memory (and not only to ciphers using stateless Boolean functions), if the number of possible states is small. For example, we may consider these as a stream cipher with a stateless combiner chosen at random among a set of Boolean functions. From our general attack given in Section 6, it can be seen that such equations may still exist simultaneously for several Boolean functions. However if for example the state is uniformly distributed and is XORed to the output, this way to obtain equations will not work. This does not say that there are no equations, and they could still be found by some type of clever elimination of the state bits. There seems that there is no reason at all that such attacks should not exist, and in fact they do exist for some real stream ciphers with a stateful combiner, for example for the Bluetooth keystream generator E0, as shown by Frederik Armknecht [2, 3].

Probabilistic Version. Our general attack S5 will also have a probabilistic version:

S5 There exists a non-trivial multivariate relation of low degree that relates (only) the key bits and several output bits of the cipher.

S6 There exists a non-trivial multivariate relation of low degree true with some probability $(1 - \varepsilon)$ that relates the key bits and several output bits of the cipher.

Again, ε will have to be very small for such an attack to be practical.

8 Conclusion

In this paper we studied algebraic attacks on stream ciphers with linear feedback (e.g. using LFSRs), such that the only non-linear component is a filtering function. We reduce their cryptanalysis to solving a system of algebraic equations, namely an overdefined system of multivariate binary equations of low degree. We present a method to decrease the degree of the equations, by multiplying them by well chosen multivariate polynomials. Thus, we are able to cryptanalyse two well known stream ciphers.

For Toyocrypt, a Cryptrec submission, our best attack requires only 2^{49} CPU clocks and some 20 Kbytes of keystream, for a 128-bit cipher. Compared to the best known attack on Toyocrypt so far by Mihaljevic and Imai [19], our new attack has simultaneously a much lower complexity, much lower memory, and much looser requirements on the keystream. We also attacked LILI-128, a Nessie submission, and obtained an attack in 2^{57} CPU clocks for a 128-bit cipher, unfortunately far from being practical (requires 2^{57} keystream bits). This attack can be seen as the best known, if only speed is regarded, but requires a lot of keystream.

The main contribution of the present paper is the following. If the non-linear function of a cipher with linear feed-back uses only a small subset of state bits, the cipher will be insecure, though satisfying all the previously known design criteria. If only k keystream bits are used out of n, it is widely known that an attack in $\binom{n}{k}^{\omega}$ exists, whatever is the Boolean function, see [14] or [4]. Thus many stream ciphers, have been designed in such a way that, one has for example $\binom{n}{k}^{\omega} \approx 2^{80}$. In practice, since the worst-case complexity of our attack is $\binom{n}{k/2}^{\omega}$, roughly the square root of the previous one, we can break these ciphers in about 2^{40}. This attack works for any Boolean function (the worst-case). The examples of Toyocrypt and LILI-128 show that for specific ciphers, the resistance against algebraic attacks may be substantially worse: it is not clear how to make sure that some similar algebraic attacks will not break them.

It has long been known that for stateless Boolean functions used in stream ciphers one is confronted with design criteria that may conflict each other. Our attacks impose even stronger restrictions on the choice of such functions. Extrapolating from our general attack S5, we proposed a very general security criterion for stream ciphers: the non-existence of multivariate relations of low degree relating the key bits and the output bits. It turns out to be basically identical to the security criterion defined in Section 2 of [8] for multivariate trapdoor functions, and also to the requirements advocated in [10] for S-boxes of block ciphers. Moreover, it turns out that this security criterion is also very important for combiners with memory, see [2, 3, 12].

Important Note: The attacks described in the present paper work given any subset of keystream bits. In [12] it is shown that if the keystream bits are consecutive, the attack complexity can be substantially reduced, while using exactly the same initial multivariate equations.

Acknowledgments: Many thanks to Philip Hawkes, Josef Pieprzyk and the anonymous referees of Eurocrypt for their helpful comments.

References

1. Ross Anderson: *Searching for the Optimum Correlation Attack,* FSE'94, LNCS 1008, Springer, pp 137-143.
2. Frederik Armknecht: *A Linearization Attack on the Bluetooth Key Stream Generator,* Available on `http://eprint.iacr.org/2002/191/`.
3. Frederik Armknecht, Matthias Krause: *Algebraic Atacks on Combiners with Memory,* Crypto 2003, LNCS 2729, pp. 162-176, Springer.
4. Steve Babbage: *Cryptanalysis of LILI-128,* Nessie project internal report, available at `https://www.cosic.esat.kuleuven.ac.be/nessie/reports/`.
5. Eli Biham: *A Fast New DES Implementation in Software,* FSE'97, Springer, LNCS 1267, pp. 260-272.
6. Paul Camion, Claude Carlet, Pascale Charpin and Nicolas Sendrier: *On Correlation-immune Functions,* Crypto'91, LNCS 576, Springer, pp. 86-100.

7. Don Coppersmith, Shmuel Winograd: *Matrix multiplication via arithmetic progressions,* J. Symbolic Computation (1990), 9, pp. 251-280.
8. Nicolas Courtois: *The security of Hidden Field Equations (HFE)*, Cryptographers' Track Rsa Conference 2001, LNCS 2020, Springer, pp. 266-281.
9. Nicolas Courtois and Jacques Patarin: *About the XL Algorithm over* $GF(2)$, Cryptographers' Track RSA 2003, LNCS 2612, pages 141-157 ,Springer 2003.
10. Nicolas Courtois and Josef Pieprzyk: *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations,* Asiacrypt 2002, LNCS 2501, pp.267-287, Springer, A preprint with a different version of the attack is available at `http://eprint.iacr.org/2002/044/`.
11. Nicolas Courtois: *Higher Order Correlation Attacks, XL algorithm and Cryptanalysis of Toyocrypt,* ICISC 2002, November 2002, Seoul, Korea, LNCS 2587, pp. 182-199, Springer. An updated version is available at `http://eprint.iacr.org/2002/087/`.
12. Nicolas Courtois: *Fast Algebraic Attacks on Stream Ciphers with Linear Feedback,* Crypto 2003, LNCS 2729, pp: 177-194, Springer.
13. Eric Filiol: *Decimation Attack of Stream Ciphers,* Indocrypt 2000, LNCS 1977, pp. 31-42, 2000. Available on `eprint.iacr.org/2000/040`.
14. Jovan Dj. Golic: *On the Security of Nonlinear Filter Generators,* FSE'96, LNCS 1039, Springer, pp. 173-188.
15. Jovan Dj. Golic: *Fast low order approximation of cryptographic functions,* Eurocrypt'96, LNCS 1070, Springer, pp. 268-282.
16. Willi Meier and Othmar Staffelbach: *Fast correlation attacks on certain stream ciphers,* Journal of Cryptology, 1(3):159-176, 1989.
17. Willi Meier and Othmar Staffelbach: *Nonlinearity Criteria for Cryptographic Functions,* Eurocrypt'89, LNCS 434, Springer, pp.549-562, 1990.
18. Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone: *Handbook of Applied Cryptography,*CRC Press.
19. M. Mihaljevic, H. Imai: *Cryptanalysis of Toyocrypt-HS1 stream cipher*, IEICE Transactions on Fundamentals, vol. E85-A, pp. 66-73, Jan. 2002. Available at `http://www.csl.sony.co.jp/ATL/papers/IEICEjan02.pdf`.
20. Jacques Patarin: *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88,*Crypto'95, Springer, LNCS 963, pp. 248-261, 1995.
21. Rainer A. Rueppel: *Analysis and Design of Stream Ciphers,* Springer, New York, 1986.
22. Palash Sarkar, Subhamoy Maitra: *Nonlinearity Bounds and Constructions of Resilient Boolean Functions,* Crypto 2000, LNCS 1880, Springer, pp. 515-532.
23. Palash Sarkar: *The Filter-Combiner Model for Memoryless Synchronous Stream Ciphers,* Crypto 2002, LNCS 2442, Springer, pp. 533-548.
24. Alex Biryukov, Adi Shamir: *Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers,* Asiacrypt 2000, LNCS 2248, Springer, pp. 1-13.
25. Adi Shamir, Jacques Patarin, Nicolas Courtois and Alexander Klimov: *Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations*, Eurocrypt'2000, LNCS 1807, Springer, pp. 392-407.
26. L. Simpson, E. Dawson, J. Golic and W. Millan: *LILI Keystream Generator,* SAC'2000, LNCS 2012, Springer, pp. 248-261, Cf. `www.isrc.qut.edu.au/lili/`.
27. Fredrik Jonsson, Thomas Johansson: *A Fast Correlation Attack on LILI-128,* Can be found at `http://www.it.lth.se/thomas/papers/paper140.ps`
28. Markku-Juhani Olavi Saarinen: *A Time-Memory Tradeoff Attack Against LILI-128,* FSE 2002, LNCS 2365, Springer, pp. 231-236, available at `http://eprint.iacr.org/2001/077/`.
29. Claude Elwood Shannon: *Communication theory of secrecy systems,* Bell System Technical Journal 28 (1949), see in patricular page 704.
30. Volker Strassen: *Gaussian Elimination is Not Optimal,* Numerische Mathematik, vol 13, pp 354-356, 1969.

A Computer Simulations

Algebraic attacks may be controversial, because in most cases, all the equations generated are **not** linearly independent. In the algebraic attack on block ciphers proposed in [10], the number of equations generated is limited, and if there are too many linear dependencies, the attack will fail. The situation with stream ciphers is very different: for each keystream bit, we get one equation and thus can obtain as many equations as we want. Then, given a sufficient quantity of keystream, that in practice turns out to be still quite realistic, the attack should always work.
Let T be the number of monomials in these equations, and R be the number of equations. We conjecture that, for all the attacks described in the present paper, the number of linear dependencies among these equations is negligible. From this we expect the following.

- When $R < T$ most of the equations are linearly independent, and $rank \approx R$.
- Then when $R \approx T$ we have $rank \approx T$,
- Finally when, for example $R = 1.01 \cdot T$, the rank should be exactly equal to the number of non-constant monomials, i.e. we should obtain $rank = T - 1$. Such a system of full rank can always be solved Gaussian reduction.

All our simulations always confirmed this conjecture.
Example: Simulations on Toyocrypt We programmed our attack on Toyocrypt exactly as described in Section 3.2, except that we replaced the connection polynomial of the MLFSR that is not published by Cryptrec, by the commonly used irreducible polynomial $X^{128} + X^{29} + X^{27} + X^2 + 1$,
An attack in 2^{49} is a bit too slow to be experimented on a PC. In order to make the attack slightly faster, we will fix the values of some of the key bits. Let $0 \leq n_a \leq 128$ denote the number of variables that are still unknown. For example if we fix 48 bits to 0, we have $n_a = 128 - 40 = 80$ and we are still breaking a version of Toyocrypt in which the initial state has 80 bits. This cipher cannot be broken neither by exhaustive search, nor by any other attack previously known.
In the following table we computed the rank of the system of equations generated for the attack on Toyocrypt described in Section 3.2. This is done for different values of n_a, and for different choices for R: when $R < T$, then when $R \approx T$, and finally when R slightly exceeds T. We did simulations up to $n_a = 80$, which with our (non-optimized) implementation, was the maximum that could be done on a PC in one day of CPU time. We note that the equations are cubic and we have $T = \binom{n_a}{3} + \binom{n_a}{2} + \binom{n_a}{1} + \binom{n_a}{0}$.
Results: We observe that the number of linear dependencies is becoming negligible when $n_a \to 128$, as expected. We also observed that not only the proportion but even the (absolute, not relative) number of linear dependencies is decreasing when $n_a \to 128$. This implies that, for the full Toyocrypt, at most about $R = T + 10$ equations should be sufficient to obtain a solvable system of a maximum rank $= T - 1$. Then the complexity of our attack will be exactly 2^{49}.

n_a/n	T	R	$rank$	$\min(R,T-1) - rank$	$\frac{rank}{\min(R,T-1)}$
20/128	1351	$0.5 \cdot T$	558	117	0.82605
20/128	1351	T	1181	170	0.87417
20/128	1351	$1.5 \cdot T$	1316	34	0.97409
20/128	1351	$2 \cdot T$	1350	0	1.00000
40/128	10701	$0.5 \cdot T$	5316	34	0.99355
40/128	10701	T	10667	33	0.99682
40/128	10701	$1.1 \cdot T$	10700	0	1.00000
80/128	85401	T	85391	9	0.99988
80/128	85401	$T+50$	85400	0	1.00000
80/128	85401	$1.01 \cdot T$	85400	0	1.00000

Table 4. Algebraic Attack on Toyocrypt - Computer Simulations

B A sequel to the Section 2.6

How to avoid linearly dependent equations. Given a Boolean function f we denote by $\mathrm{S3a}^d(f)$ the number of linearly independent polynomials g of degree d, satisfying S3a. In the same way we define $\mathrm{S3b}^d(f)$ and $\mathrm{S3c}^d(f)$.

We see that, by definition, these three scenarios S3a, S3b and S3c exclude each other, i.e. given f, there is no g that satisfies several scenarios at the same time.

we observe that the scenarios S3b is invariant under a non-zero linear combination of the possible g functions. The scenarios S3a and S3c are not invariant: some linear combinations may have lower degree or become 0. Therefore, if we have for example $\mathrm{S3a}^d(f) = 14$ $\mathrm{S3b}^d(f) = 10$ and $\mathrm{S3c}^d(f) = 10$ this does not mean that we have necessarily $14 \cdot m + 2 \cdot 10 \cdot m/2$ linearly independent equations. For example for the 10 linearly independent equations in S3c, some of their linear combinations may be in S3a. How to solve this problem ?

In general the number of linearly independent equations obtained in our attacks is at most:

$$\mathrm{S3a}^d(f) \cdot m + \mathrm{S3b}^d(f) \cdot m/2 + \mathrm{S3c}^d(f) \cdot m/2$$

In practice it is probably less. An alternative, probably better way to find out the number of independent equations is to consider the following:

• $\mathrm{S3}_0^d$ will be the number of linearly independent equations g of degree $\leq d$ that are such that $f(s) = 0 \Rightarrow g(s) = 0$.

• $\mathrm{S3}_1^d$ will be the number of linearly independent equations g of degree $\leq d$ that are such that $f(s) = 1 \Rightarrow g(s) = 0$.

Attack scenario considered	Degree of f	Degree of g	Use g such that	Use the equation	Only when	Number of equations for m keystream bits
S1 and[6] S2	low	0	$g = 1$	$f(s) = b_t$	always	m
$\mathrm{S3}_0$ and[6] $\mathrm{S4}_0$	high	low, $g \neq 0$	$f(s) = 0 \Rightarrow g(s) = 0$	$g(s) = 0$	$b_t = 0$	$m/2$
$\mathrm{S3}_1$ and[6] $\mathrm{S4}_0$	high	low, $g \neq 0$	$f(s) = 1 \Rightarrow g(s) = 0$	$g(s) = 0$	$b_t = 1$	$m/2$

Table 5. The alternative classification of ways to obtain low degree equations from keystream bits

Now we estimate the number of linearly independent equations obtained in our attacks to be:

$$\mathrm{S3}_0^d(f) \cdot m/2 + \mathrm{S3}_1^d(f) \cdot m/2$$

It is expected that all, or great majority of these equations will be linearly independent.

C A different proof of Theorem 6.0.1

We prove a (very slightly) weaker theorem, by a different method.

Theorem C.0.1 (Low Degree Relations Slightly Weaker Version).
Let f be any Boolean function $f : GF(2)^k \rightarrow GF(2)$. Then there is a Boolean function $g \neq 0$ of degree at most $\lceil k/2 \rceil$ such that $f(x) \cdot g(x)$ is of degree at most $\lceil k/2 \rceil$.

Proof: We consider a Boolean function f with k inputs. The number of possible inputs is 2^k and if we fix the output to $b = 0$ or 1, for at least one of these possibilities we have:

$$\exists\ b \in \{0,1\}, s.t. \quad |\{x|f(x) = b\}| \leq \frac{1}{2}2^k$$

Let I_b^k be this set of inputs. We will create a following matrix: lines are all the possible elements of I_b^k. The columns are all the monomials of degree up to $\lceil k/2 \rceil$. The number of columns is:

$$\sum_{i=0}^{\lceil k/2 \rceil} \binom{k}{i} > \frac{1}{2}2^k$$

Each entry in the matrix is the value $\in \{0,1\}$ of the column monomial for the entry corresponding to the current line.
The number of columns is strictly greater than the number of lines, therefore a non-trivial linear combination of columns must be zero. This combination of columns (i.e.) monomials of degree $\leq \lceil k/2 \rceil$, amounts to having a multivariate function g of degree $\leq \lceil k/2 \rceil$, such that:

$$\exists b \in \{0,1\}, s.t. \quad \forall x \in I_b^k,\ g(x) = 0.$$

Therefore:

$$\exists b \in \{0,1\}, s.t. \quad \forall x, \quad f(x) = b \ \Rightarrow\ g(x) = 0.$$

By inspection we verify that the following equation is true and that fg satisfies all the requirements of the Theorem:

$$\forall x \in \{0,1\}^k, \quad f(x) \cdot g(x) = (1 + b) \cdot g(x) \qquad \square$$

Remarks: Apart from the degree that will be sometimes slightly lower (only in very special cases), we showed here a stronger result than Theorem 6.0.1: the equation not only satisfies the two degree conditions, but is also very special:

1. It can be written in a very special form:

$$(f(x) + 1 + b) \cdot g(x) = 0$$

2. There are solutions such that fg is only of small degree, but it is either 0 or equal to g. This is surprising and is difficult to expect from our previous proof.

D Attacks on Toyocrypt and LILI-128 with XL

In this part we will give some additional algebraic attacks on Toyocrypt and LILI-128. They are motivated by a situation in which the keystream amount available is not sufficient to apply the main linearization attack. In this case one should apply the XL algorithm, that generalizes the linearization technique.

D.1 Solving Overdefined Systems of Multivariate Equations

In all our attacks, as in [11], after clocking the keystream generator t times, the state is a known linear combination of the key bits. Hence if we know some m bits of the keystream, we have m equations with n variables, as follows:

$$\begin{cases} b_{t_1} = f\left(L^{t_1}(k_0, \ldots, k_{n-1})\right) \\ \quad\vdots \\ b_{t_m} = f\left(L^{t_m}(k_0, \ldots, k_{n-1})\right) \end{cases}$$

We recall that f, and all the L^{t_i} are public, and only the k_j are secret. Then we derive other equations, of lower degree, as explained in Section 2.6.

A Simple Linearization Attack

It is already known that when f has just a low algebraic degree d, the system can be easily broken given $m = \binom{n}{d}$ keystream bits. This method is called linearization [25]:
Linearization Method: There are about $T = \binom{n}{d}$ terms of degree d in the k_i, and we consider each of them as a new variable V_i. Then given $R \geq \binom{n}{d}$ keystream bits we get a system of $R \geq \binom{n}{d}$ linear equations with $T = \binom{n}{d}$ variables V_i that can be easily solved by Gaussian elimination.
An example of a successful application of this attack to LILI-128 can be found in [4].

The XL Algorithm

The XL algorithm will work when we do not have as many as $\binom{n}{d}$ keystream bits, but still the system is largely overdefined. It is proposed in [25] and studied in detail in [9, 11] for the cases relevant here (i.e. higher degree multivariate equations over $GF(2)$). We recall quickly how it works over $GF(2)$. Given a system of m equations with n unknowns over $GF(2)$. For each D we consider all the multiples of the given equations by a monomial, that are of the total degree D. Let R be the number of such equations that are generated, we have $R \approx m \cdot \binom{n}{D-d}$. Let $T \approx \binom{n}{D}$ the total number of terms in these equations. For each D we take the smallest m for which $R/T > 1.1$. Then the we add a new variable for each of T monomials and solve the resulting linear system. It is not obvious if this always works, as the equations may not always be linearly independent, however the detailed analysis given in [11] and the computer simulations shows that when $R/T > 1.1$, we expect the XL attack to work exactly as predicted, at least for random systems of equations.

Do We Need XL ?

In this paper, we add an important preliminary step (see methods S3 and S4 defined above), in which from the initial equations, we generate other equations of substantially lower degree. In the scenario S3, equations are true with probability 1, and thus we may obtain as many such equations as required, given enough keystream bits. Then for XL, we will have a largely overdefined system, and it turns out the best way to solve such a system by XL is to put $D = d$ in XL. In this case XL boils down to the simple linearization method as described in the main paper.

Only when, for some reasons the amount of the keystream is restricted, it makes sense to use XL. The idea is that, When the number of equations obtained from the keystream is insufficient for linearization, we still get a very overdefined system of equations. Below we give some examples of application of XL to Toyocrypt and LILI-128.

D.2 Application of XL to Toyocrypt

In practice, in order to apply XL to our system of equations (the same system that in the main paper, except that it has less equations), we proceed as follows. For each $D = d, d+1, \ldots$ we consider all the multiples of the given equations by a monomial, that are of the total degree D. Then for each D we take the smallest m for which $R/T > 1.1$ and from [11] we know that in this case the XL attack always works exactly as described. Thus, for Toyocrypt, we get the following results:

d	3	3	3	3	3	3
ε	0	0	0	0	0	0
D	3	4	5	6	7	8
Data	$1.5 \cdot 2^{17}$	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}
Memory	2^{37}	2^{47}	2^{56}	2^{65}	2^{73}	2^{81}
Complexity	$\mathbf{2^{49}}$	2^{62}	2^{76}	2^{88}	2^{99}	2^{110}

Table 6. Our attack on Toyocrypt extended by XL, possible trade-offs

D.3 Application of XL to LILI-128

Here, as explained in Section 4.1, we have two versions A and B. In the first version (A) the state of the first generator is guessed and the complexity is multiplied by 2^{39}, In the second version (B), the first component is clocked $2^{39} - 1$ clocks at a time and it is the number of keystream bits that is multiplied by 2^{39}. The attacks of type B requires a lot of keystream, but it can be seen that we only need s very small subset of it, namely only bits that are on some positions of the form $\alpha + \beta \cdot (2^{39} - 1)$, for a fixed α. Once the state at the time α of the second LFSR is recovered, the state of the first LFSR can be found very quickly within 2^{39} tries and a few additional keystream bits.

Intermediate variants between A and B are also possible, see Section 4.1.

Here are the results that can be obtained with XL:

Version	A	A	A	B	B	B
d	4	4	4	4	4	4
ε	0	0	0	0	0	0
D	4	5	6	4	5	6
Data	2^{18}	2^{15}	2^{14}	2^{57}	2^{54}	2^{53}
Memory	2^{43}	2^{51}	2^{58}	2^{43}	2^{51}	2^{58}
Complexity	2^{96}	2^{107}	2^{118}	$\mathbf{2^{57}}$	2^{68}	2^{79}

Table 7. Our attack on LILI-128 extended by XL, possible trade-offs

Remark: Again, for both Toyocrypt and LILI-128, our fastest attack happens to be when $D = d$ and in this case XL boils down to the simple linearization as described in Section 2.7: we just consider each monomial as a variable and solve a system of linear equations. This attack however requires the biggest amount of the keystream. We observe also that the complexity of the attacks increases substantially for a moderate decrease in the amount of keystream.