Fast Algebraic Attacks on Stream Ciphers with Linear Feedback

Nicolas T. Courtois

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

Abstract. Many popular stream ciphers apply a filter/combiner to the state of one or several LFSRs. Algebraic attacks on such ciphers [10, 11] are possible, if there is a multivariate relation involving the key/state bits and the output bits. Recent papers by Courtois, Meier, Krause and Armknecht [1, 2, 10, 11] show that such relations exist for several well known constructions of stream ciphers immune to all previously known attacks. In particular, they allow to break two ciphers using LFSRs and completely "well designed" Boolean functions: Toyocrypt and LILI-128, see [10, 11]. Surprisingly, similar algebraic attacks exist also for the stateful combiner construction used in Bluetooth keystream generator E0 [1]. More generally, in [2] it is proven that they can break in polynomial time, any combiner with a fixed number of inputs and a fixed number of memory bits.

In this paper we present a method that allows to substantially reduce the complexity of all these attacks. We show that when the known keystream bits are consecutive, an important part of the equations will have a recursive structure, and this allows to partially replace the usual sub-cubic Gaussian algorithms for eliminating the monomials, by a much faster, essentially linear, version of the Berlekamp-Massey algorithm. The new method gives the fastest attack proposed so far for Toyocrypt, LILI-128 and the keystream generator that is used in E0 cipher. Moreover we present two new fast general algebraic attacks for stream ciphers using Boolean functions, applicable when the degree and/or the number of inputs is not too big.

Key Words: Algebraic attacks, stream ciphers, multivariate equations, nonlinear filters, Boolean functions, combiners with memory, LFSR synthesis, Berlekamp-Massey algorithm, Toyocrypt, Cryptrec, LILI-128, Nessie, E0, Bluetooth.

1 Introduction

In this paper we study stream ciphers with linear feedback. In such ciphers there is a linear component, and a stateful or stateless nonlinear combiner that produces the output, given the state of the first part. For stateless combiners - using a Boolean function, most of the general attacks known are correlation attacks, see for example [20, 16, 15, 9]. In order to resist such 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 [9]. Unfortunately there is a tradeoff between these two properties. One of the proposed remedies is to use a stateful combiner, as for example in the Bluetooth cipher E0 [6].

Recently the scope of application of the correlation attacks have been extended to consider higher degree correlation attacks with respect to non-linear low degree multivariate functions, or in other words, allowing to exploit low degree approximations [10]. The paper [10], proposes a novel algebraic approach to the cryptanalysis of stream ciphers. It will reduce the problem of key recovery, to solving an overdefined system of algebraic equations (i.e. many equations). Following [10] and [11], all stream ciphers with linear feedback are potentially vulnerable to algebraic attacks. If for one state we are able, by some method, to deduce from the output bits, only one multivariate equation of low degree in the state bits, then the same can (probably) be done for many other states. Each equation remains also linear with respect to any other state, and given many keystream bits, we inevitably obtain a very overdefined system of equations. Then we may apply the XL algorithm from Eurocrypt 2000 [25], adapted for this purpose in [10], or the simple linearization as in [11], to efficiently solve the system.

In the paper [11], the scope of algebraic attacks is substantially extended, by showing new non-trivial methods to obtain low degree equations, that are not low degree approximations. The method to reduce the degree of the equations, is analogous to the method proposed by Courtois and Pieprzyk to attack some block ciphers [13], and the basic idea goes back to [12] and [23]. Instead of considering outputs as functions of inputs, one should rather study multivariate relations between the input and output bits. They turn out to have a substantially lower degree. In this paper we take this idea further, and consider more general equations that include potentially many output bits, instead of one (or few) considered in [11, 1]. Then we propose a new fast method to find and exploit such equations, mainly due to an application of a non-trivial asymptotically fast algorithm. This method allows to obtain equations that cannot be obtained by any other known method (due to their size), and to get much faster algebraic attacks.

The paper is organized as follows: in Section 2 we give a general view of algebraic attacks on stream ciphers. In Sections 2.2, 2.4 and 3 we discuss in details the types of equations that will be used in our attacks. In Section 4 we design our pre-computation attack and in Section 5 we show how to speed-up this attack. Then in Section 6 we apply it to Toyocrypt, LILI-128 and E0. In Section 7.1 we present a new general fast attack on ciphers with Boolean functions of low degree. Finally in Section 7.2 we apply our fast method to substantially speed up the general attack on stream ciphers using (arbitrary) Boolean functions with a small number of inputs from [11].

2 Algebraic Attacks on Stream Ciphers

For simplicity we restrict to binary stream ciphers defined over $GF(2)$. We consider only synchronous stream ciphers [21], in which there is a state $s \in GF(2)^n$. At each clock t the state s is updated by a "connection function" $s \mapsto L(s)$ that is assumed to be linear over $GF(2)$. Then a combiner f is applied to s, to produce the output bit $b_t = f(s)$. In principle also, we consider only regularly clocked stream ciphers. However this condition can sometimes be relaxed, see the attacks on LILI-128 described in [11]. Then the successive keystream bits $b_0, b_1, \ldots$ are XORed with the plaintext.

We assume that L and f are public, and only the state is secret. In many cases we assume that f is stateless, i.e. a Boolean function. Then our description covers "filter generators", in which the state of a single LFSR 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 [21], and also many other known constructions. We will also apply our fast algebraic attacks to the case when the combiner f is not a Boolean function, and contains memory bits, as for example in E0. Then if if the number of memory bits is not too big, efficient algebraic attacks will still exist.

2.1 The General Framework of the Attack

The problem we want to solve is the following: find the initial state given some keystream bits. In principle, it is a known plaintext attack. However in some cases, ciphertext-only attacks are possible: For example if the plaintext is written in the Latin alphabet, and does not use too many special characters, we expect all the bytes to have their most significant bit equal to 0. Then all our attacks will work, only multiplying the amount of ciphertext needed by 8. Indeed this is equivalent to knowing all consecutive keystream bits for the same cipher, in which we replaced L by its eight-fold composition (L^8).

The goal of our attacks is to recover the initial state $(k_0, \ldots, k_{n-1})$ from some m consecutive keystream bits $b_0 \ldots b_{m-1}$, by solving multivariate equations. (Unlike in [11], our fast algebraic attacks do require consecutive bits.)

There are many different closely related approaches to algebraic attacks on stream ciphers and they differ mainly by the types of equations they use. This usually determines the methods used to generate, and to solve these equations.

Attacks Based on Direct Equations Following [10, 11]

This approach applies only for stateless combiners. The problem of cryptanalysis of such a stream cipher is described as follows in [11].

Let $(k_0, \ldots, k_{n-1})$ be the initial state. Then the output of the cipher (i.e. the keystream) gives the following system of equations:

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

In [10] such equations are solved directly. In [11] they are first multiplied by well chosen multivariate polynomials, which decreases their degree, and then solved. In this paper we describe a more advanced method of generating (better) derived equations.

2.2 More General Attacks Using "Ad-hoc" Equations

Following [11], multivariate equations/relations that relate (only) key bits and output bits, may exist, for very different reasons, for (potentially) any cipher:

$$\begin{cases} 0 = \alpha + \sum \beta_i k_i + \sum \gamma_i b_i + \sum \delta_{ij} k_i b_j + \sum \epsilon_{ijk} k_i b_j b_k + \\ \quad\quad \vdots \end{cases}$$

In this paper we restrict ourselves to equations that are true with probability 1.

Types of Equations: Following the notations introduced in Section 5.1. of [12], the equations given above may be called of type $1 \cup k \cup b \cup kb \cup kb^2$. This notation is obvious to understand, for example 1 denotes the presence of a constant monomial and kb^2 of all monomials of type $k_i b_j b_k$. Following another convention used in [12], we use capital letters to denote types of equations that also contain lower degree monomials, for example $K^2 = k^2 \cup k \cup 1$ and $B = b \cup 1$. Therefore the equations given above can also be said of type $KB \cup kb^2$ but they are not of type KB^2 because they do not contain monomials in $b_i b_j$ (a.k.a. monomials of type b^2).

Using Ad-hoc Such Equations: Our attacks will proceed in three steps:

- **Step A1.** Pre-computation stage: Find these equations.
- **Step A2.** Given some keystream, substitute the b_i in the equations to get an overdefined system of multivariate equations in the k_i.
- **Step A3.** Solve this (very overdefined) system of equations. Given sufficiently many keystream bits, we apply the simple linearization technique [25, 11], which consists of adding one new variable for each monomial that appears in the system, and then solves a big linear system. If less keystream is available, one should use a version of the XL algorithm [25] for equations of any small degree over $GF(2)$, described and studied in [10].

2.3 Remarks on Ad-hoc Equations

The idea of ad-hoc equations follows closely the idea called scenario S5 in Section 7 of [11]. The word emphasises their unexpected character: they may be found be clever elimination by hand as in [1, 2] or by an indirect unexpectedly fast method as in the present paper. The idea itself can be seen as a higher-degree generalisation of the concept of "augmented function" proposed by Anderson in [3] and recently exploited in [18]. It is also, yet another application in cryptography of looking for multivariate relations of low degree, main method for attacking numerous multivariate asymmetric schemes [12, 23], and recently proposed also for attacking block ciphers [13].

Obviously, if such equations involving, say, T monomials exist, they can be found in time T^ω, with $\omega < 3$ being the exponent of the Gaussian reduction. The important contribution of this paper is to show that in some cases such equations can be found in a time linear or even sub-linear[1] in T.

[1] It will be essentially linear in the number of equations involved in elimination, denoted later by L. This, assuming that the equations are written in a compressed form, can indeed be sub-linear in their size T.

A different view of "ad-hoc" equations is adopted in [2]. Instead of considering the equations with no limitation in the degree in the b_i, and study their degree when we substitute for b_i their respective values, it is possible to look at the equations in the state/key bits that are true each when several consecutive outputs are fixed to some fixed values. This approach used in [2] allows to study equations that contain much less monomials, which allows to find them faster, yet cannot be applied to systems in which the number of outputs is very big, as in the present paper.

Another way of looking at "ad-hoc" equations will be to consider that the stream cipher having one output, is in fact using several output functions $f_i \stackrel{def}{=} f \circ L^i$. Then "ad-hoc" equations boil down to look for algebraic combinations of type $\sum_i f_i g_i$ that would be of unusually low degree, exactly as in Section 2 of [12]. Then, if the g_i are also of low degree, the attack will exploit, for $j = 0, 1, 2, \ldots$, the following equation of low degree:

$$\sum_i f_i(L^j(s)) \cdot g_i(s) = \sum_i b_{i+j} \cdot g_i(s).$$

This approach is much more powerful than the S3 attack scenario [11]. Such equations may (and will) exist, without a low degree product fg to exist. They may (and do) exist even for stateful combiners f [1, 2].

2.4 Equations Used in the S3 Attack [11] and Additional Properties

Let f be a Boolean function. We assume that the multivariate polynomial f has some multiple fg of low degree, with g being some non-zero multivariate polynomial. Let $deg(fg) = d$ and let d be not too big, then following [11], efficient attacks exist. For each known keystream bit at position t, we obtain a concrete value of $b_t = f(s)$ and this gives the following equation:

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

which, for the current $s = L^t(k_0, \ldots, k_{n-1})$, rewrites as:

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

If the degree of g is also $\leq d$, this equation can be used for any value of b_t, otherwise it can still be used when $b_t = 0$, i.e. for half of the time. We get one multivariate equation for each one or about two keystream bits. Each of these equations will be of the same degree and we inevitably obtain a very overdefined system of equations (number of equations $\gg n$), that can be efficiently solved, see [10, 11, 25].

Important Remark: Following [11], the chief advantage of the equations explained above may be of very low degree, without f being of low degree. In this paper we explore an additional property of these equations. Let e the degree of g. In [11], no distinction is made[2] between d and e. In this paper, on the contrary, we will exploit equations in which $e < d$, and show that for the same n and d, a smaller e leads allows to compute another type of "ad-hoc" equations, that will only be of degree e in the k_i, leading to much faster attacks than in [11].

[2] In the extended version of the paper [11], two versions of the main attack are studied, called S3a and S3b, that simply require that $e \leq d$. The complexity of the attack does not depend on e, only on $\binom{n}{d}$. For another proposed version S3c, successful attacks could be possible even when e is very big.

More generally, in the next section we define a subclass of "ad-hoc" equations (including the example above with $e < d$) that will be subsequently used in the next sections to compute efficiently other (much better) "ad-hoc" equations.

3 The Double-Decker Equations

Definition 3.0.1 (Double-Decker Equations). For any $e < d$, we call "double-decker equations" with degrees (d, e, f), any set of multivariate equations of type $K^d \cup K^e B^f$, with $d, e, f \in \mathbb{N}$. In other words, equations with a maximum degree d in the monomials containing only the k_i, and the maximum degree e in the k_i among the monomials being divisible by one of the b_j.

The Double-decker equations will be used to find even better "ad-hoc" equations of type $K^e B^f$, with $f \in \mathbb{N}$, and finally used in our attacks. These equations cannot be obtained directly due to their size, and will be found indirectly in two major steps that will be described later:

- **Step A1.1.** Find some "double-decker equations" with degrees (d, e, f).
- **Step A1.2.** Then eliminate (at least) all the monomials of types $k^{e+1} \dots k^d$, leaving only monomials of type $K^e B^f$ with a maximum degree e in the k_i.

3.1 Step A1.1. - Find the Basic Double-Decker Equations

The exact method to find the "double-decker equations" we will use later varies from one cipher to another.

Toyocrypt: Toyocrypt is a stream cipher, submitted to the Japanese government call for cryptographic primitives Cryptrec and accepted to the second phase. An impractical attack on Toyocrypt has been proposed [22] and it has been rejected by Cryptrec. A description can be found in [22]. From Section 5.1. of [11] we know that there are 2 equations of type $f(s) \cdot g(s) = b_t \cdot g(s)$ in which $e = deg(g) = 1$ and $d = deg(fg) = 3$. For example, anyone can verify that when $g(s) = (s_{23} - 1)$, then in $f(s)g(s)$, all the terms of degrees ≥ 4 cancel out and what remains is of degree 3.

LILI-128: LILI-128 is a stream cipher, that was submitted to the European evaluation effort Nessie, and was subsequently rejected due to attacks [27]. A description can be found in [26]. From Section 5.1. of [11] we know that there are 4 equations of type $f(s) \cdot g(s) = b_t \cdot g(s)$ in which $e = deg(g) = 2$ and $d = deg(fg) = 4$. For example, when $g(s) = s_{44}s_{80}$, anyone can verify that $f(s)g(s)$ is equal to the following multivariate polynomial of degree 4:

$$f(s) \cdot s_{44}s_{80} = s_{44}s_{80}\left(s_1 s_{65} + s_3 s_{30} + s_7 s_{30} + s_{12} s_{65} + s_0 + s_7 + s_{12} + s_{20}\right).$$

E0: EO is the keystream generator used in the Bluetooth wireless interface [6]. We will look for some equations of the following type:

Table 1. The equations that will be used in our attacks

stream cipher	the degrees d	e	f	equation type	number of equations	successive b_i per equation
Toyocrypt	3	1	1	$K^3 \cup bk^1$	2	1
LILI-128	4	2	1	$K^4 \cup bK^2$	4	1
E0	4	3	2	$K^4 \cup B^2K^3$	1	4

$$h(s) = \sum_i b_i \cdot g_{i\ 0}(s) + \sum_i b_i b_j \cdot g_{i\ j}(s),$$

in which $e = \max(deg(g_{i\ j})) < d = deg(h)$. They may be called of type $K^d \cup B^2K^e$ following notation of Section 2.2 or [12]. Such an equation of type $K^4 \cup B^2K^3$, combining only 4 successive states, and eliminating all the state bits has been found by careful study of the cipher and successive elimination done by hand in $[1,2]$. In this equation we have $d = 4$ and $e = 3$. Our simulations confirmed that this equation exists, and is always true. We also found that it was unique (when combining only 4 consecutive states).

3.2 Summary: The Equations that will be Used in Our Attacks

For all the three ciphers Toyocrypt, LILI-128 and E0 there are "double-decker equations" (i.e. multivariate equations of type $K^d \cup K^eB^f$), with the following degrees:

Important Remark: Nothing proves that there are no better equations. For any pair (d,e), even very small, when the number of b_i used grows, the simulations are becoming quickly impractical and cannot detect all equations that would lead to a (fast) algebraic attack.

4 The Pre-Computation Attack on Stream Ciphers

This idea has been suggested to us by Philip Hawkes. Starting from the above "double-decker equations" with degrees (d,e,f), we will be able to compute "ad-hoc" equations of degree only e, by eliminating the monomials of type K^d, as defined in Step A1.2. First, it will be done by Gaussian elimination, then by a much faster method.

4.1 The General (Slow) Pre-computation Algebraic Attack

We assume that for a given stream cipher, we have a system of "double-decker equations" being of type $K^d \cup K^eB^f$, with f being small, for example in all our attacks $f \leq 2$. We also assume that the size of these initial equations is a small constant $\mathcal{O}(1)$ (for example, they are sparse or/and use a small subset of state bits). We will write the equations in the following form

$$\mathrm{Left}(L^t(k)) = \mathrm{Right}(L^t(k), b),$$

and we will put all the monomials of type K^d on the left side, and all the other monomials of type K^eB^f on the right side. For monomials of type K^e we may place them indifferently on one or the other side. Then we do the following:

1. We assume that we have at least $R = \binom{n}{d} + \binom{n}{e} \approx \binom{n}{d}$ equations, this can be achieved given about at most[3] about $\binom{n}{d}$ keystream bits.
2. We will do a complete Gaussian elimination of all monomials that appear on the left sides, i.e. of all monomials of degree up to d in the k_i. This is possible because R is chosen to exceed the number of monomials. This gives at least one linear combination α of the left-hand sides that is 0:
$$0 = \sum_t \alpha_t \cdot \text{Left}(L^t(k)).$$
Since $R = \binom{n}{d} + \binom{n}{e}$, we are able to produce at least $\binom{n}{e}$ such linearly independent linear combinations. Each of them involves about $\binom{n}{d}$ left sides.
3. The complexity of this first step is about $\binom{n}{d}^{\omega}$ to find several solutions α (we need about $\binom{n}{e}$ solutions, and $\binom{n}{e} \ll \binom{n}{d}$.
4. We apply these linear combinations to the right sides. We get a system of equations of type $B^f K^e$.
$$0 = \sum_t \alpha_t \cdot \text{Right}(L^t(k)).$$
For example, for Toyocrypt or LILI-128, we get:
$$0 = \sum_t \alpha_t \cdot b_t \cdot g\left(L^t(k)\right).$$
Similarly for E0 we get equations of the type B^2K^2 using up to $\binom{n}{4}$ consecutive b_i.
5. Now we obtain a pre-computed information (a sort of trapdoor) that consists of $\binom{n}{e}$ equations. Each of these equations is of size $\mathcal{O}(\binom{n}{d})$, this is because we assume that the size of the initial equations is a small constant $\mathcal{O}(1)$. This trapdoor information allows, given a sequence b_i at the previously determined positions t that were in the sequence, to compute the secret key k by solving a system of equations of degree e instead of d.
6. Given a sequence b_t, we substitute it to all the equations. This step about takes $\mathcal{O}(\binom{n}{d} \cdot \binom{n}{e})$ computations.
7. Then we have a system of $\binom{n}{e}$ equations of degree e, that can be solved by linearization.
8. The complexity of the last linearization step is about $\binom{n}{e}^{\omega}$.

Remark: If all the equations are not linearly independent, the attack still works. Simulations done in the extended version of [11] suggest that the number of linearly dependent equations should always be negligible. Moreover, it is possible to see that even if we had to produce, say 10 times more equations, it would not greatly increase the complexity of the attack.

[3] As a matter of fact, here for many ciphers there will be several, say M, equations that exist for each keystream bit. Then given about $\binom{n}{d}/M$ keystream bits, we still get about $\binom{n}{d}$ equations as required. Unfortunately, for the fast attack described in Section 5 below, we only know how to use one of them and will assume $M = 1$.

4.2 Summary: The General (Slow) Pre-Computation Attack

To summarize we have a pre-computation attack that, given $\binom{n}{d}$ initial equations, allows to compute a trapdoor information, that later for any values of b_t obtained from the cipher, will allow to compute the key using only roughly $\binom{n}{d} \cdot \binom{n}{e} + \binom{n}{e}^{\omega}$ operations. For now, when applied with equations from Section 3.2, it does not improve on the attacks from [11] and [1, 2], except that with the same complexity we may do a pre-computation once, and then break the cipher again and again with a much lower complexity.

5 The Fast Pre-Computation Attack

In this (fast) attack there are three additional requirements (or limitations):

1. The equation used has to be exactly the same for each keystream bit, modulo a variable change resulting from the fact that $s = L^t$. For example in the previous attack we could use, for each keystream bit b_t, one or several equations of type $f(L^t(k)) \cdot g(L^t(k)) = b_t \cdot g(L^t(k))$ with different linearly independent functions g. In this attack we may only use one g that has always to be the same.
2. The slow pre-computation attack given above, as all attacks given in [11], will work given any subset of keystream bits. The improved (fast) attack will require consecutive keystream bits. (The attack will also work if they are taken at regular intervals. This amounts simply to taking a different linear feedback function L and without loss of generality we will assume that all the keystream bits are consecutive.)
3. Moreover we assume that the linear feedback L is non-singular, in a sense that the sequence $k, L(k), L^2(k), \ldots$ is always periodic, and we will also assume that this sequence has only one single cycle. This is true for all known stream ciphers with linear feedback, in particular when they are based on one or several maximum-length LFSRs.

When all the equations are of the form Equation$(L^t(k))$ and if all the keystream bits are consecutive, then the system has a very regular recursive structure:

$$\begin{cases} \text{Left}\,(k_0, \ldots, k_{n-1}) = \text{Right}\,(k_0, \ldots, k_{n-1}; b_0 \ldots b_{m-1}) \\ \qquad \vdots \\ \text{Left}\,(L^t(k_0, \ldots, k_{n-1})) = \text{Right}\,(L^t(k_0, \ldots, k_{n-1}); b_t \ldots b_{m+t-1}) \\ \qquad \vdots \end{cases}$$

Now we are going (for some time) to ignore the right sides of these equations. Their left sides do not depend on the b_i, they are just multivariate polynomials of type K^d. If we consider at least $\binom{n}{d}$ consecutive equations, a linear dependency α must exist. This linear dependency does not depend on the outputs b_i. Let S be the size of the smallest such linear dependency α. We have $S \leq \binom{n}{d}$.

One dependency is enough. Due to the recursive structure of the equations, this dependency α can be applied at any place. Indeed we have:

$$\forall k \quad 0 = \sum_{t=0}^{S-1} \alpha_t \cdot \text{Left}\left(L^t(k)\right) \qquad \Rightarrow \qquad \forall k\ \forall i \quad 0 = \sum_{t=0}^{S-1} \alpha_t \cdot \text{Left}\left(S^{t+i}(k)\right).$$

Moreover, since we assumed that the sequence $k, L(k), L^2(k), \ldots$ has one single cycle, this dependency α does not depend on the secret key of the cipher, and is the same for all k.

In the previous (slow) pre-computation attack we had to find $\binom{n}{e}$ linear dependencies α. Here we find one α and re-use it on $\binom{n}{e} \ll S$ successive windows of $S \leq \binom{n}{d}$ equations.

5.1 Finding α Faster - LFSR Synthesis

We have shown a (well known) fact, that for every initial k, the values of the left sides of our equations can be obtained from some LFSR of length at most S and defined by α. We can also do the reverse: recover α from the sequence (LFSR synthesis). It can be done given $2S$ bits of the sequence, see [19, 21]. For this we choose a random key k', (α does not depend on k), and we will compute $2S$ output bits of this LFSR $c_t = \text{Left}\left(L^t(k')\right)$ for $t = 0 \ldots 2S-1$. Then we apply the well known Berlekamp-Massey algorithm [19, 21] to find the connection polynomial of this LFSR that will be essentially α. Done in this way, both these steps (computing the sequence c_i and the LFSR synthesis) will take $\mathcal{O}(S^2)$ operations.[4] However both can be improved to become essentially linear in S.

Finding α Faster:

- ◇ First of all we observe that all the $L^i(k')$ can be computed in time about $\mathcal{O}(Sn^2)$ and even in $\mathcal{O}(Sn)$ if L is composed only of a combination of LFSRs and MLFSRs. Then, since $n \ll S \approx \binom{n}{d}$, the time is indeed essentially linear in S. Moreover, in many interesting cases in practice, the equations $\text{Left}\left(S^t(k')\right)$ are sparse or/and have a known structure that allows to compute them very fast, in a time that can be assumed constant, being less than n and therefore much smaller than S. It is the case for Toyocrypt, LILI-128 and E0. We exploit here the fact that these ciphers have been designed to be very fast.
- ◇ To recover α takes $\mathcal{O}(S^2)$ computations using Berlekamp-Massey Algorithm, but it will take only $\mathcal{O}(S\log(S))$ operations using improved asymptotically fast versions of the Berlekamp-Massey Algorithm, see [5, 14, 7]. However we do not know how fast are these algorithms for the concrete values of S used in this paper.

How to Use α: We recall that the same linear dependency will be used $\binom{n}{e}$ times:

[4] A quadratic time is already much faster than doing this by Gaussian elimination in $\mathcal{O}(S^\omega)$, and only this would already give the fastest attack known on all the three stream ciphers studied in this paper.

$$\forall i \quad 0 = \sum_{t=i}^{S+i-1} \alpha_t \cdot \text{Right}\left(S^t(k); b_t \ldots b_{m+t-1}\right).$$

Since $Right(t)$ does depend on many b_i, in general a sliding subset, and the output sequence b_i is not likely to have any periodic structure, all these equations are expected to be very different for different i. From simulations on a simpler but similar algebraic attack on Toyocrypt done in [11], we also expect that only a negligible number of these equations will be redundant (i.e. linearly dependent).

5.2 Summary - the Fast Pre-Computation Attack

1. Given about $m = \binom{n}{d} + \binom{n}{e} \approx \binom{n}{d}$ consecutive keystream bits.
2. We compute the linear dependency α using an improved version of the Berlekamp-Massey Algorithm. This step is expected to be essentially linear in S and take at most $\mathcal{O}(S\log(S) + Sn)$ steps with $S = \binom{n}{d}$. This α is our pre-computed information (or the trapdoor). It allows, given a sequence b_i of consecutive keystream bits, to recover the secret key k by solving a system of equations of degree only $e < d$.
3. The second step takes (as before) about $\mathcal{O}(\binom{n}{d} \cdot \binom{n}{e}) + \binom{n}{e}^{\omega}$ operations.

Memory Requirements: In this attack, the most memory-consuming operation will be to store the $\binom{n}{e}$ equations of size at most $\binom{n}{e}$ (we only store them after substituting the b_i by the output bits). It will be at most $\binom{n}{e}^2$ bits.

6 Application to Toyocrypt, LILI-128 and E0

In this paper, write $\mathcal{O}(2^{20})$ to say that the complexity is at most $C \cdot 2^{20}$ for some constant C, yet for simplicity we did not evaluate the exact value of C. All our results given in such form should be regarded as rough (and optimistic) approximations.

6.1 Application to Toyocrypt

For Toyocrypt we have $n = 128$, $d = 3$, $e = 1$. We obtain the following attack:
- ♢ With a pre-computation step in $\mathcal{O}(2^{23})$.
- ♢ Given $2^{18.4}$ consecutive keystream bits.
- ♢ The secret key can be computed in about $\mathcal{O}(2^{20})$ CPU clocks and with about 2^{14} bits of memory.

6.2 Application to LILI-128

In the second component of LILI-128 we have $n = 89$, $d = 4$, $e = 2$. The sequence will only become regularly clocked if we clock the first LFSR of LILI-128 $2^{39} - 1$ steps at a time, see [11]. We get the following attack:
- ♢ With a pre-computation step in $\mathcal{O}(2^{26})$.
- ♢ Given $2^{21.3+39} \approx 2^{60}$ consecutive keystream bits.
- ♢ The state of the second component can be computed in about $\mathcal{O}(2^{31})$ CPU clocks and with about 2^{24} bits of memory.
- ♢ Once the initial state of the second LFSR is recovered, the state of the first LFSR can be found easily in less than about 2^{20} (many thanks to Philip Hawkes for remarking this). Indeed, once the second LFSR is known, we can

Table 2. The results of this paper vs. the best previous attacks

cryptosystem	Toyocrypt			LILI-128				E0	
n	128	128	128	89	89	89	89	128	128
d		3	3	4	4		4	4	4
e			1				2		2
Attack	[22]	[11]	new	[11]	[11]	[27]	new	[1, 2]	new
Data	2^{48}	2^{18}	2^{18}	2^{18}	2^{57}	2^{46}	2^{60}	2^{24}	2^{24}
Memory	2^{48}	2^{37}	2^{14}	2^{43}	2^{43}	2^{51}	2^{24}	2^{48}	2^{37}
Pre-computation	2^{80}		$\mathbf{O(2^{23})}$			2^{56}	$\mathbf{O(2^{26})}$		$\mathbf{O(2^{28})}$
Attack Cxty	2^{64}	2^{49}	$\mathbf{O(2^{20})}$	2^{96}	2^{57}	2^{56}	$\mathbf{O(2^{31})}$	2^{68}	$\mathbf{O(2^{49})}$

predict any number of consecutive bits a_i of the second component (without decimation), then we may guess several (for example 20) consecutive output bits c_i of the first component, which determines a subsequence of the a_i that should be equal to the observed output sequence fragment b_i. Our choice will be confirmed only for on average 1 or 2 strings of 20 bits of c_i. For each of these (very few) cases, we will find the remaining $39 - 20 = 19$ bits by the exhaustive search.

6.3 Application to E0

For E0 we have $n = 128$, $d = 4$, $e = 3$. Then we obtain:

- ◇ With a pre-computation step in $\mathcal{O}(2^{28})$.
- ◇ Given $2^{23.4}$ consecutive keystream bits.
- ◇ The secret key can be computed in about $\mathcal{O}(2^{49})$ CPU clocks and with about 2^{37} bits of memory.

Note: In the real-life implementation of Bluetooth cipher, at most about $2745 \approx 2^{11}$ bits can be obtained, see [17, 6]. However this attack shows that the design of E0 is not (cryptographically) very good. It is possible that even a real-life application of E0 will be broken by our attack, if some other equations with a smaller d are found. This possibility cannot at all be excluded, the computer simulations only allow to explore equations that combine a few (e.g. up to 10) output bits. Better equations may exist when more bits are combined, and may be found by a clever elimination as in [1, 2].

6.4 Summary of the Results

The fast algebraic attack gives the best attack known on three well known stream ciphers:

7 Fast General Attacks on Stream Ciphers Using Boolean Functions

In [11] it is shown that for any cipher with linear feedback and a non-linear stateless filtering function that uses only a small subset of state bits, for example k bits out of n state bits, the key can be recovered in essentially $\binom{n}{k/2}^{\omega}$ operations, which at most about $n^{k\frac{\omega}{2}} \leq n^{1.19k}$ using the fairly theoretical result from [8]. In practice, if we consider the Strassen's algorithm, we get rather $n^{1.4k}$. This attack works given any subset of $\binom{n}{k/2}$ keystream bits.

In this paper we show that, given less than $\binom{n}{k}$ keystream bits, that (now) have to be consecutive, we may recover the key using only essentially n^k computations instead of $n^{1.4k}$. In the two following subsections we will show two separate attacks of this type that both will be better than $n^{k\omega/2}$.

The first attack assumes that the Boolean function of the cipher is constructed in such a way that it can be computed in constant time (for example using a table, the usual case, otherwise the cipher is not practical !). Given $\binom{n}{d}$ consecutive keystream bits, the key is recovered in $n^{d+\mathcal{O}(1)}$ computations, with $d \leq k$ being the degree of the Boolean function used. This attack can be sometimes much faster than n^k, as frequently we have $d < k$.

We also present a fast version of the general attack in $n^{k\frac{\omega}{2}}$ from [11], that for ciphers using only a small subset of output bits k, (k is a small constant, e.g. $k = 10$), will require (substantially) less than $\binom{n}{k}$ consecutive keystream bits, and about $n^{k+\mathcal{O}(1)}$ computations.

7.1 Fast General Attack on Stream Ciphers Using Boolean Functions of Small Degree

Theorem 7.1.1 (New General Attack on Stream Ciphers with Linear Feedback). If the degree of the Boolean function f is d, and its output can be computed in time $\mathcal{O}(1)$, then given $\binom{n}{k}$ consecutive keystream bits, one can recover the key using only $\mathcal{O}(n^{d+2})$ operations.

All this is achieved immediately by a straightforward application of our fast algebraic attack. The cipher is first described by the following set of equations:

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

We will split this equation into two $0 = f(s) + b_t = \mathrm{Left}(s) + \mathrm{Right}(s, b_t)$ with:

- $\mathrm{Right}(s, b_t)$ containing b_t and the linear part of f.
- $\mathrm{Left}(s)$ being exactly the part of degrees $2 \ldots d$ of f.

It is easy to see that, from the assumption that the computation of f is done in constant time, the time to compute $Left()$ will be in $\mathcal{O}(n)$, knowing that it differs from f only by the linear part. Then we get the following set of equations for some m consecutive bits $t = 0 \ldots m-1$:

$$\mathrm{Left}\left(L^t(k_0, \ldots, k_{n-1})\right) = \mathrm{Right}\left(L^t(k_0, \ldots, k_{n-1}),\ b_t\right), \qquad t = 0, 1, 2, \ldots$$

with the left sides being of type K^d and right sizes of type $K \cup b$, and using only one b_i each. We have "double-decker equations" with degrees $(d, e, f) = (d, 1, 1)$.

Table 3. Fast algebraic attack when f has degree d

Consecutive Data	$\binom{n}{d}$
Memory	$\mathcal{O}(n^{d+1})$
Pre-computation	$\mathcal{O}(n^{d+2})$
Complexity	$\mathcal{O}(n^{d+1})$

The LFSR Synthesis. Let $S = \binom{n}{d}$. Let k' be a random key. From Section 5.1, we recall that the time to compute all the $L^i(k'), i = 0 \ldots 2S-1$ is at most $\mathcal{O}(Sn^2)$, and in many practical cases even $\mathcal{O}(Sn)$.
Since the time to compute $Left(s)$ is $\mathcal{O}(n)$ computing $2S$ outputs $Left(L^t(k'))$ for $t = 0 \ldots 2S-1$ will take time $\mathcal{O}(Sn)$. Then as in Section 5 we compute the LFSR connection polynomial α which takes about $\mathcal{O}(S \log S)$ operations.
Using the Pre-computed Information α. Again, as in Section 5, with a pre-computation in time at most $\mathcal{O}(Sn^2 + S \log S)$ we get an equation in which a linear combination of $S = \binom{n}{d}$ successive keystream bits b_i is equal to a linear combination of the k_i:

$$\forall t \sum_{i=0}^{S-1} \alpha_{t+i} b_{t+i} = \text{ResultingLinearCombination}(k_0, \ldots, k_{n-1}).$$

By using this equation n times for n successive windows of S keystream bits, we get a linear system that will give the key k. The complexity of the whole fast pre-computation attack is at most about $\mathcal{O}(Sn^2)$. In practice, it will be frequently at most $\mathcal{O}(Sn)$ and at any rate this time can be thought as essentially linear in S, since $n \ll S$. We get the following attack:

Later, in Table 5, we will compare this attack to the attack from the next section, and other previously known general attacks.

Important Remark. This is a very simple linear attack on a stream cipher. Such equations of size $S = \binom{n}{d}$ do always exist. Let LC of the linear complexity of a cipher. We always have $LC \leq \binom{n}{d}$ and when $S < LC$, the existence of such an equation of size S can be excluded. Conversely, if the linear complexity LC is less than expected ($LC < \binom{n}{d}$), the attack described here will still work with a complexity essentially linear in LC. This attack shows that, ciphers in which the Boolean functions f is of low degree and can be computed in constant time, can be broken in time essentially linear in the linear complexity LC of the cipher. This is, to the best of our knowledge, has never happened before.

7.2 Fast General Attack on Stream Ciphers Using a Subset of LFSR bits

This section improves the general attack from [11]. We consider a stream cipher with n state bits, in which the keystream bit is derived by a Boolean function f using only a small subset of k state bits: $\{x_1, x_2, \ldots, x_k\} \subset \{s_0, s_1, \ldots, s_{n-1}\}$. We assume that k is a small constant and n is the security parameter. For example in LILI-128 $k = 10, n = 89$.
As in [11] we are looking for polynomials $g \neq 0$, such that fg is of low degree, and for this we check for linear dependencies in the set of polynomials $C = A \cup B$ defined as follows. A contains all 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 e (this degree corresponds to the degree of g):

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

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 set, linear dependencies will exist. Such linear combinations allow to find a function g of degree $\leq e$ such that $f \cdot g$ is of degree $\leq d$. More precisely, we will prove the following theorem that generalizes the Theorem given in [11]:

Theorem 7.2.1 (Tradeoff Between the Degrees of fg and g). Let f be any Boolean function $f : GF(2)^k \to GF(2)$. For any pair of integers (d, e) such that $d + e \geq k$ there is a Boolean function $g \neq 0$ of degree at most e such that: $f(x) \cdot g(x)$ is of degree at most d.

Proof: We have $$|A| = \sum_{i=0}^{d} \binom{k}{i} \quad \text{and} \quad |B| = \sum_{i=0}^{e} \binom{k}{i}.$$

$$|C| = \sum_{i=0}^{d} \binom{k}{i} + \sum_{i=0}^{e} \binom{k}{i} = \sum_{i=0}^{d} \binom{k}{i} + \sum_{i=0}^{e} \binom{k}{k-i} \; > \; \sum_{i=0}^{k} \binom{k}{i} = 2^k.$$

Then, since the rank of $C = A \cup B$ cannot exceed 2^k, and $|C| > 2^k$, some linear dependencies must exist. Moreover, $g \neq 0$ because there are no linear dependencies in A, and therefore linear dependencies must combine either only the elements of B, or both A and B. This ends the proof. □

From this we see that for any pair of integers (d, e) such that $d + e \geq k$ and for any stream cipher with linear feedback, for which the non-linear filter uses k variables, it is possible to generate "double-decker equations" with degrees $(d, e, 1)$ in the n keystream bits.

With this assumption, computing Left(), and Right(), that by construction have at most 2^{k-1} monomials, will be very fast (using a table) and given 2^k bits of memory. this will be small, because we assumed that k is a small constant, for example for LILI-128 $k = 10$, $2^k = 1024$ is very small compared to the memory requirements of other parts of the attack.

Then the whole LFSR synthesis will take a time of at most $\mathcal{O}(Sn^2 + S \log S)$. By inspection we verify that our fast pre-computation attack gives roughly about:

Table 4. Fast algebraic attack when f has k inputs

$\forall(d,e)$ s.t. $d+e \geq k$	
Consecutive Data	$\binom{n}{d} + \binom{n}{e}$
Memory	$\mathcal{O}(2^k + \binom{n}{d}\binom{n}{e})$
Pre-computation	$\binom{n}{d}^{1+o(1)}$
Complexity	$\mathcal{O}(\binom{n}{d}\binom{n}{e}) + \binom{n}{e}^{\omega}$

Table 5. General attacks on LFSR-based stream ciphers using a Boolean function f

State bits	n	n	n	n
# inputs of f	k	k	k	k
d	k	$\lceil k/2 \rceil$	$d = deg(f) \leq k$	$k(1-1/\omega)$
e		$\lfloor k/2 \rfloor$	1	$k(1/\omega)$
Attack	[4, 16]	[11]	Section 7.1	Section 7.2
Data	$\binom{n}{k}$	$\binom{n}{\lceil k/2 \rceil}$	$\binom{n}{d}$	$\binom{n}{k(1-1/\omega)}$
Consecutive	*no*	*no*	*yes*	*yes*
Memory	$\binom{n}{k}^2$	$\binom{n}{\lceil k/2 \rceil}^2$	$\mathcal{O}(n^{d+1})$	$\mathcal{O}(n^{k+1})$
Pre-computation			$\mathcal{O}(n^{d+2})$	$\mathcal{O}(n^{k+2})$
Attack Cxty	$\mathcal{O}(n^{k\omega})$	$\mathcal{O}(n^{k\omega/2})$	$\mathcal{O}(n^{d+1})$	$\mathcal{O}(n^{k+1})$

Under the condition $d + e \geq k$, we may assume $d + e = k$, and we see that very roughly: $\mathcal{O}(\binom{n}{d}\binom{n}{e}) \approx n^d/d! \cdot n^e/e! \approx n^k \cdot \binom{k}{d}$. The complexity of this attack will never be essentially lower than n^k. However, to choose a bigger e will substantially decrease the amount of keystream needed for the attack. Many tradeoffs are possible, but two of them seem particularly interesting:

- It is easy to see that the maximum e that can be used in this attack without increasing the complexity, is such that $\binom{n}{e}^{\omega} \approx \mathcal{O}(\binom{n}{d}\binom{n}{e}) \approx n^k$. Thus we have $e \approx k(1/\omega)$ and then d is about $k(1-1/\omega)$. The amount of consecutive keystream required is then about $\binom{n}{d} = \binom{n}{k(1-1/\omega)}$.
- We may also try to achieve the smallest possible amount of keystream, which is achieved when $e = d \approx k/2$, with a slower attack. It can be seen that in this attack the fast pre-computation will not help because the final step will be slower than the pre-computation step. Moreover, sine $d = e$ the pre-computation step is not necessary at all to obtain equations of degree e, and this attack finally boils down to the general attack given in [11].

7.3 Summary - Fast General Attacks on Stream Ciphers Using Boolean Functions

In the following table we give a simplified analysis of the complexity of the attacks from Section 7.1 and 7.2 compared to previously known attacks.

8 Conclusion

In this paper we have studied algebraic attacks on stream ciphers with linear feedback, using overdefined systems of algebraic multivariate equations over $GF(2)$. This gives many interesting attacks on both stateless combiners based on Boolean functions and also on combiners with memory. Using equations of a very special form, and by a novel application of known asymptotically fast results on LFSR reconstruction, we are able to propose the best attack known so far on three important stream ciphers. All these ciphers were believed quite secure up till very recently, now they can be broken quickly on a PC.

We also present two general algebraic attacks, for all regularly clocked ciphers with linear feedback (e.g. using LFSRs) filtered by a Boolean function. Both may be faster than all previously known attacks. In particular we show a surprising result to the effect that some ciphers may be broken in time essentially linear in their linear complexity.

There are good arguments to say that these attacks will work exactly as predicted, because one may generate as many equations as one wants, and if some are linearly dependent, one may generate more new equations. The computer simulations done in the extended version of [11] suggest that the number of equations that are linearly dependent in our attacks will be negligible.

Our attacks lead to, more or less the same design criterion for stream ciphers as proposed in [11]: the non-existence of multivariate relations of reasonable size and low degree linking the key bits and the output bits. This criterion turns out to be almost identical to the security criterion defined in Section 2 of [12] for multivariate trapdoor functions, and also to the requirements advocated in [13] for S-boxes of block ciphers. However, in this paper we show that some equations may be found and used in a time linear (or even sub-linear) in the size of the equations. Therefore such equations should be taken very seriously in the design of stream ciphers, and this even for systems of equations of sizes bigger than e.g. 2^{80}. Then, the existence of such big systems of equations cannot be a priori excluded by computer simulations, that would be too slow.

Therefore we do not recommend using stream ciphers with linear feedback with fairly simple clocking control. However, all algebraic attacks we are aware of, will not work, when the connections polynomials of the LFSRs are secret, or when the clocking is very complex and uses the full entropy of the key, and/or when the output sequence is decimated in a very complex way (as in shrinking generators).

Acknowledgments: The pre-computation attack on stream ciphers has been designed following an initial idea suggested by Philip Hawkes. I also thank Frederik Armknecht, Philip Hawkes, Willi Meier, Greg Rose, David Wagner, and above all, the anonymous referees of Crypto, for their very helpful comments.

References

1. Frederik Armknecht: *A Linearization Attack on the Bluetooth Key Stream Generator,* Available on http://eprint.iacr.org/2002/191/. 13 December 2002
2. Frederik Armknecht, Matthias Krause: *Algebraic Atacks on Combiners with Memory,* in these proceedings, Crypto 2003, LNCS 2729, Springer, 2003.
3. Ross Anderson: *Searching for the Optimum Correlation Attack,* FSE'94, LNCS 1008, pp 137-143, Springer. 1994.
4. Steve Babbage: *Cryptanalysis of LILI-128,* Nessie project internal report, https://www.cosic.esat.kuleuven.ac.be/nessie/reports/, 22 January 2001.
5. R. E. Blahut: *Theory and Practice of Error Control Codes,* Addison-Wesley, 1983.
6. Bluetooth CIG, Specification of the Bluetooth system, Version 1.1, February 22 2001, available from www.bluetooth.com.
7. R. P. Brent, F. G. Gustavson, D. Y. Y. Yun: *Fast solution of Toeplitz systems of equations and computation of Padé approximants.* J. Algorithms, 1:259-295, 1980.
8. Don Coppersmith, Shmuel Winograd: *Matrix multiplication via arithmetic progressions,* J. Symbolic Computation (1990), 9, pp. 251-280.
9. Paul Camion, Claude Carlet, Pascale Charpin and Nicolas Sendrier, *On Correlation-immune Functions,* Crypto'91, LNCS 576, pp. 86-100, Springer, 1991.
10. Nicolas Courtois: *Higher Order Correlation Attacks, XL algorithm and Cryptanalysis of Toyocrypt,* ICISC 2002, LNCS 2587, Springer. An updated version (2002) is available at http://eprint.iacr.org/2002/087/.
11. Nicolas Courtois and Willi Meier: *Algebraic Attacks on Stream Ciphers with Linear Feedback,* Eurocrypt 2003, Warsaw, Poland, LNCS 2656, pp. 345-359, Springer. An extended version is available at http://www.minrank.org/toyolili.pdf
12. Nicolas Courtois: *The security of Hidden Field Equations (HFE)*, Cryptographers' Track Rsa Conference 2001, San Francisco 8-12 April 2001, LNCS 2020, Springer, pp. 266-281, April 2001.
13. Nicolas Courtois and Josef Pieprzyk, *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations,* Asiacrypt 2002, LNCS 2501, Springer, a preprint with a different version of the attack is available at http://eprint.iacr.org/2002/044/.
14. Jean-Louis Dornstetter: *On the Equivalence Between Berlekamp's and Euclid's Algorithms.* IEEE Trans. on Information Theory. IT-33(3): 428-431. May 1987.
15. Eric Filiol: *Decimation Attack of Stream Ciphers,* Indocrypt 2000, LNCS 1977, pp. 31-42, 2000. Available on eprint.iacr.org/2000/040.
16. Jovan Dj. Golic: *On the Security of Nonlinear Filter Generators,* FSE'96, LNCS 1039, pp. 173-188, Springer, 1996.
17. Jovan Dj. Golic, Vittorio Bagini, Guglielmo Morgari: *Linear Cryptanalysis of Bluetooth Stream Cipher,* Eurocrypt 2002, LNCS 2332, pp. 238-255, Springer, 2002.
18. Bernhard Löhlein: *Attacks based on Conditional Correlations against the Nonlinear Filter Generator,* Available at http://eprint.iacr.org/2003/020/,
19. J. L. Massey: *Shift-register synthesis and BCH decoding,* IEEE Trans. Information Theory, IT-15 (1969), 122-127.
20. Willi Meier and Othmar Staffelbach: *Fast correlation attacks on certain stream ciphers*, Journal of Cryptology, 1(3):159-176, 1989.
21. Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone: *Handbook of Applied Cryptography*, Chapter 6, CRC Press.
22. 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.
23. Jacques Patarin: *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88,* Crypto'95, Springer, LNCS 963, pp. 248-261, 1995.
24. Rainer A. Rueppel: *Analysis and Design of Stream Ciphers,* Springer, 1986.
25. Adi Shamir, Jacques Patarin, Nicolas Courtois, Alexander Klimov, *Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations*, Eurocrypt'2000, LNCS 1807, pp. 392-407, Springer, 2000.
26. L. Simpson, E. Dawson, J. Golic and W. Millan: *LILI Keystream Generator,* SAC'2000, LNCS 2012, pp. 248-261, Springer, 2000. See www.isrc.qut.edu.au/lili/.
27. Markku-Juhani Olavi Saarinen: *A Time-Memory Tradeoff Attack Against LILI-128,* FSE 2002, LNCS 2365, pp. 231-236, Springer, 2002, available at http://eprint.iacr.org/2001/077/.