Algebraic Attacks on Combiners with Memory and Several Outputs*

Nicolas T. Courtois

Axalto Cryptographic Research & Advanced Security, 36-38 rue de la Princesse
BP 45, F-78430 Louveciennes Cedex, France, courtois@minrank.org

Abstract. Algebraic attacks on stream ciphers [14] recover the key by solving an overdefined system of multivariate equations. Such attacks can break several interesting cases of LFSR-based stream ciphers, when the output is obtained by a Boolean function, see [14–16]. Recently this approach has been successfully extended also to combiners with memory, provided the number of memory bits is small, see [1, 16, 2]. In [2] it is shown that, for ciphers built with LFSRs and an arbitrary combiner using a subset of k LFSR state bits, and with l inner state/memory bits, a polynomial attack always do exist when k and l are fixed. Yet this attack becomes very quickly impractical: already when k and l exceed about 4.

In this paper we give a simpler proof of this result from [2], and prove a more general theorem. We show that much faster algebraic attacks exist for any cipher that (in order to be fast) outputs several bits at a time. In practice our result substantially reduces the complexity of the best attack known on four well known constructions of stream ciphers when the number of outputs is increased. We present attacks on modified versions of Snow, E0, LILI-128, Turing, and some other ciphers.

Note: This is an extended version of the paper published in ICISC 2004.

Key Words: algebraic attacks on stream ciphers, pseudo-random generators, multivariate equations, linearization, XL algorithm, Gröbner bases, nonlinear filtering, Boolean functions, combiners with memory, LILI-128, Turing cipher, Snow, Nessie, E0, Bluetooth.

1 Introduction

In this paper we study LFSR-based stream ciphers. In such ciphers there is an inner state updated by an iterated linear function, and a stateful or stateless nonlinear combiner that produces the output, given the inner state of the first (linear) part. Our goal is to extend the recent very powerful and very general algebraic attacks on stream ciphers to the case of combiners with several outputs. Such constructions appear naturally if we want to construct ciphers being fast in practice.

Up till recently, for stateless combiners - using a Boolean function - most general attacks known were so called correlation attacks, see for example [30, 23, 11, 8]. In order to resist such attacks, many authors focused

* Work supported by the French Ministry of Research RNRT Project “X-CRYPT”.

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 [11]. Unfortunately there is a tradeoff between these two properties. One of the proposed remedies is to use a stateful combiner. This idea is used in the Bluetooth wireless protocol cipher E0 [6]. Yet the simplicity of E0 made it vulnerable to advanced correlation attacks [25] and other attacks [2, 1, 16].

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 [14]. The paper [14], proposes an algebraic approach to the cryptanalysis of stream ciphers. It will reduce the problem of key recovery, to solving an overdefined system of algebraic equations. Following [14] and [15], all LFSR-based stream ciphers are (potentially) vulnerable to algebraic attacks. The argument says that, if by some method, we are able to deduce from the output bit(s), only one multivariate equation of low degree in the LFSR state bits, then the same can (probably) be done for many other states. Each equation remains also linear with respect to any other LFSR state, and given many keystream bits, we inevitably obtain a very overdefined system of equations (i.e. many equations). Then we may apply the XL algorithm from Eurocrypt 2000 [36], adapted for this purpose in [14], or the simple linearization as in [15, 5], to efficiently solve the system.

In the paper [15], 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. This gives attacks that are not correlation attacks anymore, and are purely algebraic attacks on stream ciphers. The key ingredient is a simple but very powerful method to reduce the degree of the equations: instead of considering outputs as functions of inputs, one should rather study **algebraic relations** between the input and output bits. They turn out to have a substantially lower degree. The general idea of using multivariate algebraic relations in cryptanalysis of various public and secret key cryptosystems is not new and have been proposed (for very different purposes) by Patarin'95 [34], Jakobsen'98 [26], Courtois [13, 17, 18], and recently by Courtois-Pieprzyk in attempt to break AES [19].

In stream ciphers, this type of attacks have been proposed first in [15] and turn out to be quite powerful. In most cases, as already explained, due to the recursive structure of the cipher, finding just one such multivariate relation will give a polynomial attack on a stream cipher. Very surprisingly, this "multivariate relation" attack [15], extends also to combiners with memory, in particular when the number of possible inner states is small. This can be seen as an algebraic counterpart of previous results by Meier, Staffelbach and Golic on correlation attacks on combiners with one or a few memory bits [31, 24]. For algebraic attacks, the possibility of eliminating memory bits has been first suggested by Courtois and Meier in [15]. The heuristics of [15] only says that such attacks may exist, and exhibits also a counter-example for which the current method will fail to find a useful multivariate relation that would lead to an attack (cf. Section 7 of [15]). Yet, considering relations that imply potentially many output bits, seems very promising, except that finding useful relations becomes a hard problem (how to know which outputs will be used in the relation ?). The first attack of this type for a realistic stream cipher E0, has been found by careful elimination by hand, done by Armknecht [1]. A substantial speed-up for this attack is called "Fast Algebraic Attack": [16, 3, 28].

Even more surprisingly, Krause and Armknecht have recently proven a Theorem, to the effect that for **any** combiner with k inputs and l bits of memory, an algebraic attack of this type will always exist [2]. More precisely, they show that required multivariate relations do always exist with degree at most $\lceil k(l+1)/2 \rceil$. It generalises an earlier theorem due to Courtois and Meier, giving degree $\lceil k/2 \rceil$ for $l = 0$, published in [15].

With this bound on the degree from [2], starting from about $l = 4$ memory bits algebraic attacks will quickly become quite impractical. In this paper we will give a new, much simpler proof of this theorem, and we will present a **much more general** theorem, for combiners that use several outputs instead of one. For correlation attacks, this issue has been studied in [39, 8]. For algebraic attacks, we will show that having several outputs allows to substantially lower the degree of the relations, which in turn will dramatically decrease the complexity of an algebraic attack on most LFSR-based stream ciphers. Our new theorem will also give new and valuable results for combiners without memory (i.e. using just Boolean functions).

2 Notation

We consider stream ciphers in which there is a state with a linear feedback function (for example composed of one or several LFSRs). Let $K = (K_0, \ldots, K_{n-1})$ be an n-bit secret key. Let $s = K$ be the initial state of the LFSR or the linear part of the cipher. At each clock $t = 0, 1, 2, \ldots$, the new state of the linear part is computed as $s \leftarrow L(s)$, with L being some multivariate linear transformation, for example corresponding to the connection polynomial of an LFSR, a combination of several parallel LFSRs, or a linear cellular automaton. We assume that L is public.

Only k out of n bits of the linear part of the cipher are used in the next part of the cipher called the combiner. The combiner has k inputs, m outputs and l internal memory bits. At each clock $t = 0, 1, 2, \ldots$, the combiner outputs m bits $y_0^{(t)}, \ldots, y_{m-1}^{(t)}$, for $t = 0, 1, 2, \ldots$. These output bits depend deterministically on the k input bits $x_0^{(t)}, \ldots, x_{k-1}^{(t)}$ and on internal memory bits that before and at the time t are $a_0^{(t-1)}, \ldots, a_{l-1}^{(t-1)}$. In all generality, the second component is described as a pair of functions $F = (F_1, F_2) : GF(2)^{n+l} \rightarrow GF(2)^{m+l}$, that given the current state and the input, compute the next state and the output:

$$F : \begin{cases} (y_0^{(t)}, \ldots, y_{m-1}^{(t)}) = F_1(x_0^{(t)}, \ldots, x_{k-1}^{(t)}, a_0^{(t-1)}, \ldots, a_{l-1}^{(t-1)}) \\ (a_0^{(t)}, \ldots, a_{l-1}^{(t)}) = F_2(x_0^{(t)}, \ldots, x_{k-1}^{(t)}, a_0^{(t-1)}, \ldots, a_{l-1}^{(t-1)}) \end{cases}$$

The initial inner state is $a^{(-1)}$, exists before $t = 0$, and can be anything (it is and remains unknown, the goal of the attacks being to eliminate all the monomials in the a_i).

3 Algebraic Attacks on Stream Ciphers

This Section summarizes the general idea of algebraic attacks on stream ciphers from [14], greatly extended and developed in [15].

We recall that the linear part of our cipher (a combination of one or several binary LFSRs) is composed of n bits $s_0, \ldots, s_{n-1}$. At the beginning $s = K$ (the initial LFSR state) and at each clock of the cipher, it is updated as $s \leftarrow L(s)$, with L being some known multivariate linear transformation. The general algebraic attack on such stream ciphers, following closely [15] or [16], works as follows:

- Find (by some method that is very different for each cipher) one (at least, but one is enough) multivariate relation Q of low degree d between the LFSR state bits and some M following outputs, for example:
$$Q(s_0, s_1, \ldots, s_{n-1},\ y^{(0)}, \ldots, y^{(M-1)}) = 0$$
- The same equation will apply to all consecutive windows of M states
$$Q([L^t(K)]_0, [L^t(K)]_1, \ldots, [L^t(K)]_{n-1},\ y^{(t)}, \ldots, y^{(t+M-1)}) = 0$$
- The $y^{(t)}, \ldots, y^{(t+M-1)}$ are replaced by their values known from the observed output of the cipher.
- For each keystream bit, we get a multivariate equation of degree k in the x_i.

- Due to the linearity of L, for any t, the degree of these equations is still d.
- Given many keystream bits, we inevitably obtain a very overdefined system of equations (i.e. great many multivariate equations of degree d in the K_i).
- Then we may apply the XL algorithm from Eurocrypt 2000 [36], adapted to equations of degree higher than 2 in [14]. Better results should be obtained with modern Gröbner bases techniques, such as the F5 algorithm [22].
- If we dispose of a sufficient amount of keystream, (which is frequently not very big, see [15]), the XL algorithm is not necessary and may be replaced by the so called linearization method that is particularly simple. 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 about $\binom{n}{d} + M$ keystream bits, and therefore $R = \binom{n}{d}$ equations on successive windows of M bits, 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.
- In theory, the Gaussian elimination takes time T^{ω} with $\omega \leq 2.376$ [12]. However the fastest practical algorithm we are aware of, is Strassen's algorithm [38] 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. Thus, in all numerical complexity results given in this paper we will give $7/64 \cdot T^{log_2 7}$ as an estimation of the number of CPU clocks necessary in the attack.

4 The Proof Method

Our general Theorem 5.1, given later, considers arbitrary combiners with k input bits, l memory bits, and m output bits and shows the existence of equations of some degree that lead to an algebraic attack. It generalises the main result of [2] for arbitrary combiners with one output, i.e. with $m = 1$, which in turn generalises a result obtained in [15] for memoryless combiners with single output, i.e. for $m = 1$ and $l = 0$. Our proof technique is very different than in [2] and is very similar to one used in [15].

In this section, in order to illustrate the simplicity of our proof technique, we will first prove the following theorem for combiners with $m = 1$ and $l = 1$, that is in fact a special case of both our general Theorem 5.1 given later, and of the main theorem of [2].

Theorem 4.1 (Special Case of Krause-Armknecht Theorem). Let F be an arbitrary fixed circuit/component with k binary inputs x_i, one bit of memory a, and one output y. (The output and the next state of the memory bit a, depend in an arbitrary way (but deterministically) on the k inputs and the previous memory bit.)

Then, given $M = 2$ consecutive states $(t, t+1)$, there is a multivariate equation R of degree k in the $x_j^{(i)}$, that relates only the input and the output bits, **without** any of the inner state/memory bits $a^{(t-1)}, a^{(t)}$, as follows:

$$R\left(x_0^{(t)}, \ldots, x_{k-1}^{(t)};\quad x_0^{(t+1)}, \ldots, x_{k-1}^{(t+1)};\quad y^{(t)}, y^{(t+1)}\right) = 0.$$

Remark: In this and later theorems, we will only limit the degree of the equations in the $x_j^{(i)}$. The degree in the $y_j^{(i)}$ is not important, as in an attack these values will be fixed.

Proof: We consider $2k$ variables as follows: $x_0^{(t)}, \ldots, x_{k-1}^{(t)}, x_0^{(t+1)}, \ldots, x_{k-1}^{(t+1)}$. We know that the following two memory bits $a^{(t)}$ and $a^{(t+1)}$ and the two outputs $y^{(t)}, y^{(t+1)}$, do depend only on these $2k$ variables, plus additionally on the bit $a^{(t-1)}$ present in memory at the beginning. Thus, the four values $a^{(t)}$, $a^{(t+1)}$, $y^{(t)}$ and $y^{(t+1)}$, do depend deterministically only on the $2k+1$ variables $x_0^{(t)}, \ldots, x_{k-1}^{(t+1)}$ and $a^{(t-1)}$. This is summarised on the following picture:

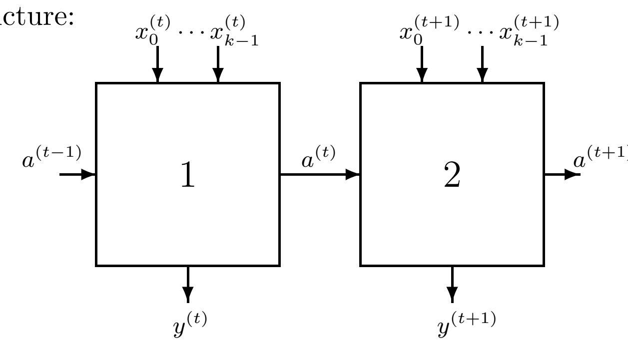

Fig. 1. Two successive applications of a combiner with k inputs, 1 output and 1 memory bit

We define the following set of monomials A: we consider all the monomials of degree up to k in the following $2k$ variables: the $x_i^{(t)}$ together with the $x_j^{(t+1)}$. The size of A is exactly $\sum_{i=0}^{k} \binom{2k}{i} = 2^{2k-1} + \frac{1}{2}\binom{2k}{k}$, which is strictly greater than 2^{2k-1}.

Now we will create the following matrix:

– Lines are all the possible values for $x_0^{(t)}, \ldots, x_{k-1}^{(t)}, x_0^{(t+1)}, \ldots, x_{k-1}^{(t+1)}$ and for the memory bit $a^{(t-1)}$. There are 2^{2k+1} lines.

- The columns correspond to products of successive monomials of A, multiplied by any out of the 4 possible monomials in the two variables $y^{(t)}, y^{(t+1)}$. There are $4 \cdot |A| = 2^{2k+1} + 2\binom{2k}{k} > 2^{2k+1}$ columns.
- Each entry in the matrix is the value $\in \{0,1\}$ of the column monomial in the case corresponding to the current line.

The number of columns is strictly greater than the number of lines. Therefore one column must be a linear combination of other columns. Since columns are products of monomials, and all the cases are treated, this gives a multivariate equation, true with probability 1, for all possible entries and whatever is the initial value of $a^{(t-1)}$. By construction, it does not involve memory bits $a^{(i)}$. This ends the proof of Theorem 4.1. $\square$

Remark 1: It can be seen that there are at least $2\binom{2k}{k} - 1$ such equations, which could greatly reduce the keystream requirements of some attacks. For simplicity we do not exploit this in the present paper.

Remark 2: In Appendix A, we give another proof of this Theorem, in which the result will be a bit stronger: in the above theorem, there are arbitrary products of degree k of the $x_i^{(t)}$ and the $x_j^{(t+1)}$, that are multiplied by one of the 4 possible monomials $1, y^{(t)}, y^{(t+1)}, y^{(t)}y^{(t+1)}$. Surprisingly, it is sufficient to consider products that do not mix the input/output variables for the first step t, with any of the variables for the second step $t+1$. This results in much less monomials being present.

5 New General Result on Combiners with Memory

In a very similar way, we will prove our main result, that extends the main theorem of [2].

Theorem 5.1 (Our Key Theorem).

Let F be an arbitrary fixed circuit/component with k binary inputs x_i, l bits of memory a_i, and m outputs y_i. Let d and M be two integers such that:

$$2^{Mm} \cdot \sum_{i=0}^{d} \binom{Mk}{i} > 2^{Mk+l} \qquad (\mathbf{KE})$$

Then, considering M consecutive steps/states $(t, \ldots t+M-1)$, there is a multivariate equation (and relation) R of degree d in the $x_j^{(i)}$, relating [1] the input and the output bits for these states

$$R\left(x_0^{(t)}, \ldots, x_{k-1}^{(t)}, \ldots, x_0^{(t+M-1)}, \ldots, x_{k-1}^{(t+M-1)};\right.$$
$$\left. y_0^{(t)}, \ldots, y_{m-1}^{(t)}, \ldots y_0^{(t+M-1)}, \ldots, y_{m-1}^{(t+M-1)}\right) = 0.$$

[1] Again, **without** any of the inner state/memory bits $a_j^{(i)}$.

Proof: .

Our proof is very similar as in the special case above (Theorem 4.1), and also gives a new, much simpler proof of the original (but less general) result of [2].

We start with the following (cf. Fig. 2):

- We have $M \cdot m$ output bits: $y_0^{(t)}, \ldots, y_{m-1}^{(t)}; \ldots; y_0^{(t+M-1)}, \ldots, y_{m-1}^{(t+M-1)}$
- The total of $M{\cdot}k$ input bits, $x_0^{(t)}, \ldots, x_{k-1}^{(t)}; \ldots; x_0^{(t+M-1)}, \ldots, x_{k-1}^{(t+M-1)}$.
- We have l initial memory bits, $a_0^{(t-1)}, \ldots, a_{l-1}^{(t-1)}$.
- In all we have $l + Mk$ input variables. The memory bits for second and following inner states, $a_j^{(t+i)}$, $0 < i < M$ do depend only on these $l + Mk$ variables.
- Thus, for our M consecutive steps/states $t, \ldots, t + M - 1$, all the outputs $y_j^{(t+i)}$, $i < M$ do depend deterministically only on the $l + Mk$ variables listed above.

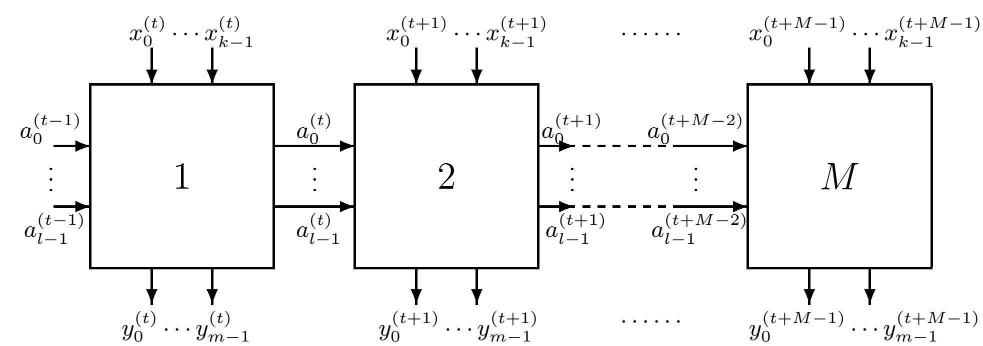

Fig. 2. M successive applications of a combiner with k inputs, m outputs and l bits of memory

We define the following set of monomials A: we consider all the monomials of degree up to d in all the Mk variables $x_i^{(t+i)}$. The size of A is exactly $\sum_{i=0}^{d} \binom{Mk}{i}$. Now we will create the following matrix:

- Lines are all the possibilities for the $l + Mk$ input variables. There are 2^{Mk+l} lines.
- The columns are all products of monomials of A, multiplied by any of the possible monomials in the $y_j^{(t+i)}$. There are $2^{Mm} \cdot |A| = 2^{Mm} \cdot \sum_{i=0}^{d} \binom{Mk}{i}$ columns.
- Each entry in the matrix is the value $\in \{0, 1\}$ of the column monomial in the case corresponding to the current line.

The key argument is the same as before. The number of columns in our matrix should be strictly greater than the number of lines, and the requirement to achieve this, is precisely our previous assumption:

$$2^{Mm} \cdot |A| = 2^{Mm} \cdot \sum_{i=0}^{d} \binom{Mk}{i} > 2^{Mk+l}$$

Therefore we get at least one non-trivial linear combination of columns (i.e. monomials) that is zero, for all possible entries and all possible initial (memory) states. This multivariate equation is (with the monomials we have chosen) exactly of the form required by our Theorem 5.1, and this ends the proof. □

Remark: It is easy to see that, if in the cipher we use LFSRs, there should be common input bits between the M consecutive states. Then Theorem 5.1 will be improved, replacing Mk by a corresponding lower value, in the right side of the above inequality. For simplicity we do not exploit this improvement in the present paper.

6 Application of Theorem 5.1 to Stream Cipher Cryptanalysis

Theorem 5.1 and other results of this paper, allow to find equations and execute the algebraic attack described in Section 3. In some cases this Theorem would work even when $d = 0$, when other variables are such that (KE) holds, but the equations of degree 0 in the $x_j^{(i)}$ will only contain the $y_j^{(i)}$, and cannot be used to recover the secret key of a cipher (though can probably be exploited to predict the future keystream). For simplicity, in this paper we will always apply Theorem 5.1 for $d \geq 1$.

6.1 The Complexity of the Attacks based on Theorem 5.1

Our algebraic attack on stream ciphers has two main steps:

Step 1. Find the equations by Gaussian reduction on the matrix given in the proof of the Theorem. This step requires about $2^{\omega(Mk+l)}$ computations.

Step 2. From the Step 1. for each keystream bit, we get one equation of degree d in the $x_j^{(i)}$ (with $d \geq 1$). The $x_j^{(i)}$ are known linear combinations of the key bits K_i and these equations are also of degree d in the key bits. When the $x_j^{(i)}$ are replaced by their actual values obtained from the keystream, we get multivariate equations that only contain monomials of degree d in the key bits K_i. Then, given about $T = \binom{n}{d}$ keystream bits, we solve these equations by linearization in about $T^\omega \approx 2^{\omega d \log n}$ computations.

In some cases (when M is small), the complexity of the first step may be negligible compared to the second step (cf. Section 7.5 and examples in Table 2). In some cases the complexity of the first step may always be very large (examples in Table 3). In other cases there will be a tradeoff between the complexity of the two steps, see Section 7.6.

Remark: The complexity of replacing the $x_j^{(i)}$ in the equations of Step 1., by the relevant (known) linear combinations of the key bits K_i (cf. Section 3) has been neglected for simplicity (it can be seen to be smaller than the maximum of complexities given above).

6.2 Important Remark

It is important to understand that, in general, this Theorem 5.1 does not show that the algebraic attack will always work. There are some (very special) cases in which it will not work as well as expected from our Theorem 5.1. We will see this on an example.

Assume that we have a component that has $m = 10$ outputs, and we artificially add 10 more outputs computed as some 10 Boolean functions of the "real" outputs:

$$(y_{10}, \ldots, y_{19}) = (F_{10}(y_0, \ldots, y_9), \ldots, F_{19}(y_0, \ldots, y_9)) .$$

Now we have (in theory) $l = 20$, and from the formula (KE) we see easily that in most cases our Theorem 5.1 will give for $m = 20$ equations of substantially lower degree than for $m = 10$. These equations are real (their existence is proven). Yet these equations will not be useful in an attack. For example there will be equations such as $y_{10} = F_{10}(y_0, \ldots, y_9)$, and a great many of derived equations: different linear combinations of these equations multiplied by many different monomials. All these equations are in a sense "artificial" and unfortunately they will all reduce to 0 later in the attack, after when the $y_0, \ldots, y_{19}$ are replaced by their values obtained from the output of the cipher.

This example shows that in some very special cases, the algebraic attack will probably not work for the degree given by our Theorem 5.1. Yet, it will probably work perfectly well for the degree corresponding to the "real" value of $l = 10$. It is conjectured that when the output bits are fully independent and not related by some algebraic relation, and if the output takes all the possible 2^m values, the attack should always work, for **every** equation obtained from the above Theorem 5.1. Moreover, in practice, the difference between the number of lines and the number of columns, in the matrix (the one we generated to prove the theorem) will be big, and there will be not only one but, (for example) thousands of equations obtained. The chances that the attack would not work for all of them, are negligible.

7 How to Choose Parameters in Theorem 5.1

In the previous Section 6 we showed that it is straightforward to use Theorem 5.1 to design an algebraic attack on stream ciphers following Section 3. Another question is to choose parameters in such a way that the complexity of the attack will be optimal. For this we need to study the behaviour of the key inequality (KE): $2^{Mm} \cdot \sum_{i=0}^{d} \binom{Mk}{i} > 2^{Mk+l}$.

In order to minimise the complexity of Step 2. of the attack (cf. Section 6.1) we simply need to choose M that gives the smallest possible d. Yet, as we will see later (in particular when $m \geq k$, cf. Section 7.6) things are not always as simple to optimise the Step 1.

7.1 Asymptotic Behaviour of $(\boldsymbol{KE})$ and Theorem 5.1

In order choose (M, d) that satisfy (KE): $2^{Mm} \cdot \sum_{i=0}^{d} \binom{Mk}{i} > 2^{Mk+l}$ we have two cases:

A. If $m < k$, when $M \to \infty$ we have no hope to satisfy the key inequality (KE). In this case we conjecture that the best attack (and the smallest degree d) will be achieved taking M as small as possible (or close to it). This case is studied in Section 7.5.

B. If $m \geq k$ then when $M \to \infty$ we can always satisfy the key inequality (KE). In this case we should take M as big as possible, but not too big because the complexity to find the equations required by the attack (Step 1. cf. Section 6.1) could become bigger than the complexity of the attack itself (Step 2.). This case is studied in Section 7.6.

Remark: For the (less general) theorem from [2], there is only the case A., because $m = 1$.

7.2 Necessary Condition for $(\boldsymbol{KE})$ and Theorem 5.1

We want to solve (KE) given the values m and l. Since one always has $\sum_{i=0}^{d} \binom{Mk}{i} \leq 2^{Mk}$, we cannot have $Mm \leq l$, and this gives **a necessary condition** $Mm > l$, hence $Mm \geq l + 1$ which gives

$$M \geq \lceil (l+1)/m \rceil. \qquad (C)$$

7.3 Sufficient Conditions for $(\boldsymbol{KE})$ and Theorem 5.1

Conversely, it is easy to see that, each time $M \geq \lceil (l+1)/m \rceil$, we have $Mm \geq l + 1$, and the formula (KE) will be satisfied for some $d \leq Mk$.

Sufficient Condition 1: For any given values m and l, and for any $M \geq \lceil (l+1)/m \rceil$, the formula (KE) will be satisfied by some d being at most $d \leq Mk$.

When the minimum $M = \lceil (l+1)/m \rceil$ is chosen, we can use $d = k \cdot \lceil (l+1)/m \rceil$, but in fact one can do better. A smaller d can be achieved for this same (minimal) M. Indeed, since M is an integer, the minimal value of M does not imply that we need to take a maximal value for d. From (KE) we get the following condition:

$$\sum_{i=0}^{d} \binom{Mk}{i} > 2^{Mk} \cdot 2^{l-m\cdot\lceil(l+1)/m\rceil}$$

It can be seen that $d = \lceil kM/2 \rceil = \lceil k\lceil(l+1)/m\rceil/2\rceil$ is always sufficient. Indeed we always have:

$$\sum_{i=0}^{d} \binom{Mk}{i} > 2^{Mk}/2.$$

And we also always have:

$$\frac{1}{2} \geq 2^{l-m\cdot\lceil(l+1)/m\rceil}.$$

Sufficient Condition 2: From the above, we get immediately the following Theorem:

Theorem 7.4 (Generalised Krause-Armknecht Theorem). Let F be an arbitrary fixed circuit/component with k binary inputs, l bits of memory, and m outputs. Then, considering $M = \lceil(l+1)/m\rceil$ consecutive steps/states $(t, \ldots t+M-1)$, there is a multivariate relation, involving only the input bits (the $x_j^{(i)}$) and the output bits (the $y_j^{(i)}$) for these states, and with degree $\lceil kM/2 \rceil = \lceil k\lceil(l+1)/m\rceil/2\rceil$ in the $x_j^{(i)}$.

Remark: If we put $m = 1$ in this Theorem 7.4 (1 output bit), we obtain exactly the main result of [2]. This in turn generalises the theorem given in [15], which is exactly the above result with $m = 1$ and $l = 0$, i.e. the case of Boolean functions that are memoryless combiners with 1 output bit.

7.5 How to Use Theorem 5.1 when $m < k$

All the remarks above are true both for $m < k$ and for $m \geq k$, however we expect that (cf. Section 7.1) choosing the smallest possible M should be optimal (or close to optimal) only when $m < k$. In some cases, the choice of Theorem 7.4 above: $M = \lceil(l+1)/m\rceil$ and $d = \lceil kM/2\rceil$ will be optimal for Theorem 5.1. However in most cases, there will be a non-zero difference between $M = \lceil(l+1)/m\rceil = 1$ and $(l+1)/m$ that will imply that $\frac{1}{2} \gg 2^{l-m\cdot\lceil(l+1)/m\rceil}$ in the derivation of Theorem 7.4 above. In such cases, it seems that the best method [2] is to take still $M = \lceil(l+1)/m\rceil$ (or very close to this) and try to the lowest d that satisfies the key requirement of Theorem 5.1 which is $2^{Mm} \sum_{i=0}^{d} \binom{Mk}{i} > 2^{Mk+l}$.

[2] Again when $m < k$, if in similar case $m \geq k$, it could be even better to increase M, cf. Section 7.6.

The Complexity of the Attacks based on Theorem 7.4

Let $d = \lceil k \lceil (l+1)/m \rceil /2 \rceil$ be the degree obtained in Theorem 7.4. Following Section 6.1, the complexity of the first step of the attack (to find the equations) will be about $2^{\omega(Mk+l)} = 2^{\omega(k\lceil (l+1)/m \rceil + l)}$ and this is roughly $\left(2^{\omega(d/2+l)}\right)$. For the second step the complexity will be about $\binom{n}{d}^{\omega} \approx n^{d\omega}$ (see Section 3). Though this d is not always the best degree we will get and use in an attack, we expect that when $m < k$ the complexity of the first step of the attack will frequently be substantially smaller than for the second step (cf. examples in Table 2).

7.6 How to Use Theorem 5.1 when $m \geq k$

If $m \geq k$, then when $M \to \infty$ we can always satisfy the key inequality (KE).

$$2^{Mm} \cdot \sum_{i=0}^{d} \binom{Mk}{i} > 2^{Mk+l} \qquad (KE)$$

This fact is obvious when $m > k$ and still true when $m = k$, because then it is sufficient to take $M = \lceil (l+1)/k \rceil$ and $d = Mk$. (Remark: here M cannot be smaller than $\lceil (l+1)/k \rceil$ because following Section 7.2, $M \geq \lceil (l+1)/m \rceil$ and here it is equal to $\lceil (l+1)/k \rceil$.)

It can be seen that in all cases when $m \geq k$, when $M \to \infty$, then d may be an arbitrarily small integer > 0 (i.e. we will even get $d = 1$ when M is large enough).

In practice, we should take M as big as possible, but not too big because the complexity to find the equations (Step 1 of the attack) will become too big: it is following Section 6.1 about $2^{\omega(Mk+l)}$ computations. (While Step 2. requires about $\binom{n}{d}^{\omega} \approx 2^{\omega \log_2 nd}/d!$)

In order to get the best attack, we need to minimise $2^{\omega(Mk+l)} + 2^{\omega \log_2 nd}/d!$ under the condition $\binom{Mk}{d} > 2^{M(k-m)+l}$. The behaviour of these complexities is not simple, because $M \geq \lceil (l+1)/m \rceil$ and must be an integer. Our experience shows that sometimes $M = \lceil (l+1)/m \rceil$ is optimal, sometimes it isn't. Sometimes the best attack will be when both complexities are about equal, sometimes the first step will always take much more time than the second step (even for the minimal $M = \lceil (l+1)/m \rceil$). Some relevant examples are given in Table 3 and Table 2.

7.7 Summary or How To Design the Best Algebraic Attack

In order to find the fastest attack with Theorem 5.1, we recommend to proceed as follows:

- First we try to apply Theorem 7.4, and get a (working) solution (M, d).

- Then with same M, take the lowest d such that the key condition (KE) still holds.
- In addition, when $m \geq k$, as long as the complexity of the first step of the attack is less than the complexity of the second step, we may try to increase M, compute the lower possible d, and see if we get a better result (Cf. Section 7.6).

7.8 Addendum, Some Interesting Cases, Theorem 5.1 vs. Theorem 7.4

Case when $\mathbf{m = l}$: In this particular case, our Theorem 7.4 gives $M = 2$ and $d = k$. In Appendix we prove a slightly stronger Theorem A.2, with the same $M = 2$ and $d = k$, but in which the variables from each of the $M = 2$ consecutive steps do not mix, and therefore much less monomials are used in the resulting equations (it is almost as if $M = 1$, yet two states are used).

Case when $\mathbf{m = l + 1}$: For $m = l + 1$, the Theorem 7.4 gives $M = 1$ and $d = \lceil k/2 \rceil$. The value of M has decreased from $M = 2$ to $M = 1$, and the value of d has decreased from $d = k$ to $d \approx k/2$. Since the complexity of the whole attack is about $\binom{n}{d}^{\omega}$, with 1 less memory bit we gain about the square root in the attack complexity. Surprisingly, the complexity of algebraic attacks **decreases a lot**, when the number of outputs m increases from l to $l + 1$, see some examples in Table 3. A similar phenomenon is observed for XL algorithm over finite fields of large characteristic, see [36].

Remark: When $m = l + 1$ and $m \geq k$, then in some cases, (following Section 7.6) it may be more interesting to increase M here to get a lower d. But it may also be bad idea, as we will show on an example taken from Table 3. When $k = 64$, $l = 64$, $k = l + 1 = 65$, from Theorem we get $M = 1$ and $d = 32$. With $M = 1$ we verify that the value of d cannot be lowered. The complexity of first/second step of the attack is then respectively about 2^{356} / 2^{471}. Yet, for $M = 2$ we may still have (KE) when $d = 15$. However with this (lower) degree we will not get a faster attack, because now the first/second step takes respectively 2^{536} / 2^{262}.

We see that though (since $m \geq k$) with bigger M we get a lower d, the complexity of the first step of the attack will become too big. This also means that in some cases, much faster attacks may exist than showed in the present paper, if there is a shortcut (due to the specific structure of the cipher) that allows to find the initial equation faster (replacing the Step 1 of the attack).

Cases when $\mathbf{m > l + 1}$. In this case we can still have $M = 1$ and d will decrease, and the attack will become better, with growing m. In particular, from Theorem 5.1, we see that when $M = 1$ and $m = k + l$ the inequality (KE) $2^{Mm} \sum_{i=0}^{d} \binom{Mk}{i} > 2^{Mk+l}$ will already be satisfied for $d = 1$, and the Step 2. of the attack will be very fast. Moreover, from Section 7.6 we see that $m = k$ is already sufficient to achieve $d = 1$ for bigger values of M. In both these cases however, the complexity of the first step of the attack may become prohibitive (unless there is a special method to find equations that comes from the structure of the cipher).

8 Application to Some Known Stream Cipher Constructions

8.1 Application to modified LILI-128

Our attack can be applied to the second component of LILI-128 cipher [37]: we have an LFSR with $n = 89$ bits, and a Boolean function with $k = 10$ inputs. There is no memory bits ($m = 0$). In [15], a generic attack on LILI-128 is given, that requires $n^{5\omega}$ computations, (whatever is the Boolean function used). From our Theorem 5.1 we see that if in LILI we use simultaneously several Boolean functions, the complexity of the generic attack will substantially decrease. It will be $\binom{n}{d}^{\omega}$ with d given by Theorem 5.1. The resulting degree d quickly decreases with m:

m	1	2	3	5	7
d	5	4	3	2	1

Following closely [15], each of these attacks on the second component of LILI-128 can be transformed into an attack on the whole LILI-128 cipher in two possible ways. Either (A:) the complexity is multiplied by 2^{39} (one needs to guess the 39-bit state of LFSR in the clocking component), or (B:) the keystream requirements are multiplied by about 2^{39} (at each step the first component is clocked $2^{39}-1$ times). See [15] for more details. This gives the following generic attack on modified LILI-128 with several outputs:

Table 1. Generic attacks on modified LILI-128 with m outputs

m	1		2		3		5		7	
M	1		1		1		1		1	
d	5		4		3		2		1	
keystream	2^{25}	2^{64}	2^{21}	2^{60}	2^{17}	2^{56}	2^{12}	2^{51}	$\mathbf{2^{6}}$	$\mathbf{2^{45}}$
time(Step 1.)	2^{25}	2^{25}	2^{25}	2^{25}	2^{25}	2^{25}	2^{25}	2^{25}	2^{25}	2^{25}
time(Step 2.)	2^{107}	2^{68}	2^{95}	2^{56}	2^{83}	2^{44}	2^{69}	2^{30}	$\mathbf{2^{54}}$	$\mathbf{2^{15}}$

We see that for ciphers that combine LFSR and Boolean functions, such as LILI-128, if we replace a Boolean function by a component that outputs a few bits at a time, the security will be dramatically reduced, and this for any component (worst case).

Note: There are attacks on LILI-128 itself, that are faster than the generic attack given here for $m = 1$, see [15, 16]. However for some of the modified versions of LILI-128 with many outputs, our attack will probably be the fastest general attack known on such ciphers.

8.2 Stream Ciphers Proposed by Gregory L. Mayhew

In [29], the author proposes a LFSR-based stream cipher that, in order to be fast, uses simultaneously several Boolean functions. These ciphers are known to be very weak.

For example, the following parameters are proposed: $n = 61$, $k = 6$, $l = 0$, $m = 16$ or $n = 61$, $k = 8$, $l = 0$, $m = 8$. We will apply our Theorem 5.1 with $M = 1$. We need to have: $2^m \cdot \sum_{i=0}^{d} \binom{k}{i} > 2^{k+l}$. On both examples we get that even $d = 0$ works. However we will put $d = 1$, because equations that do not include key bits cannot be used to recover the key (they should be useful for distinguishing attacks). We see that these ciphers are indeed very weak: when $d = 1$ they are broken by a simple algebraic attack in about $\binom{n}{1}^\omega \approx 2^{13}$ CPU clocks.

Moreover, with the results of our paper we can break much stronger versions of these ciphers. For example, it is easy to see that, if we add up to $l = 24$ memory bits to the cipher, for $M = 1$ from Theorem 5.1 we still get $d = 1$ and the cipher is still easily broken within about $\binom{n}{1}^\omega \approx 2^{13}$ CPU clocks.

We also observe that:

1. If we have $l = 15$ memory bits in the combiner, for $M = 1$ as recommended by Theorem 7.4, we get $d = 3$. Yet, for $M = 2$ we get $d = 1$. Though M is smaller, the attack is faster (we know that the Theorem 7.4 is not always optimal).
2. For $l = 16$ the Theorem 7.4 gives $d = 1$ which is optimal and we could not do better.
3. When $l = 23$, we get $M = 2$ and $d = 1$.
4. When $l = 24$ and $M = 1$ as recommended by Theorem 7.4, we get $d = 2$. For $M = 3$ we get $d = 1$ which again is faster.
5. When $d = 2$ the attack is still quite fast, which means that we can still break these ciphers quite efficiently, for even higher values of l.

8.3 Application to modified E0

For the basic component of the stream cipher E0, we have $n = 128$, $k = 4$, $l = 4$, $m = 1$. The Krause-Armknecht theorem gives $d = 10$, see [2]. With our Theorem 5.1 we get the following results:

Table 2. Generic attacks on modified E0 with m outputs

m	1	2	3	4	5	6
M	5	3	2	3	1	1
d	10	5	3	2	2	1
keystream	2^{48}	2^{28}	2^{18}	2^{13}	2^{13}	$\mathbf{2^{7}}$
time(Step 1.)	2^{64}	2^{42}	2^{30}	2^{30}	2^{19}	$\mathbf{2^{19}}$
time(Step 2.)	2^{131}	2^{76}	2^{49}	2^{33}	2^{33}	$\mathbf{2^{16}}$

We see that for ciphers that combine LFSRs and a combiner with 4 inputs, and 4 memory bits, such as E0, if one outputs several bits at a time (computed in an arbitrary way), the complexity of the attack and the keystream amount required dramatically decreases.

Note: Here we treat the worst case by a generic method, for E0 itself there are attacks faster than what we get for $m = 1$, see [2, 1, 16]. However most of modified versions of E0 with many outputs, our attack is probably the fastest attack known.

8.4 Application to Snow and Modified Versions of Snow

We consider both Snow and Snow 2.0. that have an LFSR with $n = 512$ bits that is connected to a stateful combiner that outputs $m = 32$ bits at a time. We obtain:

1. In Snow 1.0. we have $k = 64$, $l = 64$ and $m = 32$. With Theorem 7.4 we get $M = \lceil (l+1)/m \rceil = 3$ and $d = \lceil kM/2 \rceil = 96$ that can be lowered to $d = 54$ and still satisfies the requirements of the Theorem 5.1 (for reasons explained in Section 7.5).
2. Similarly, in Snow 2.0. we have $k = 96$, $l = 64$ and $m = 32$. With Theorem 7.4 we get $M = \lceil (l+1)/m \rceil = 3$ and $d = \lceil kM/2 \rceil = 144$ that can be lowered to $d = 92$.

These degrees are by far too large to give any hope for practical attacks on Snow.

Algebraic Attacks on Modified Snow

We will look how the complexity of the attack on Snow 1.0. and 2.0. when the number of output bits increases. This could arise if, in order to build a faster cipher, we add to Snow some **arbitrary** S-boxes or Boolean functions that derive some additional output bits, from the k inputs and the l memory bits of Snow combiner.

Since the size of LFSR is 512 bits, an attack will be considered significant if it takes less than 2^{512}. (We study academic attacks on modified Snow, and do not claim to break the actual Snow in which the key is expanded from a shorter key of 128 or 256 bits.)

Table 3. Generic attacks on modified Snow ciphers with m outputs

Snow 1.0. $n = 512, l = 64, k = 64$						
m	32	64	65	80	100	120
M	3	2	1	1	1	1
d	54	16	32	16	7	2
keystream	2^{245}	2^{99}	2^{169}	2^{99}	2^{51}	2^{17}
time(Step 1.)	2^{715}	2^{536}	2^{356}	2^{356}	2^{356}	2^{356}
time(Step 2.)	2^{684}	2^{276}	2^{471}	2^{276}	2^{139}	$\mathbf{2^{45}}$

Snow 2.0. $n = 512, l = 64, k = 96$					
m	32	64	65	120	150
M	3	2	1	1	1
d	92	35	48	9	2
keystream	2^{344}	2^{352}	2^{226}	2^{62}	2^{17}
time(Step 1.)	2^{985}	2^{715}	2^{446}	2^{446}	2^{446}
time(Step 2.)	2^{962}	2^{503}	2^{631}	2^{172}	$\mathbf{2^{45}}$

We see that when the number of outputs increases, the security of the cipher collapses. The complexity of the first step of the attack may be $< 2^{512}$ but remains very high. However, one should not think that Snow with added outputs will be very secure: we only gave here the complexity of the generic method to find a useful equation. For a specific cipher, in many cases, there could be a much faster method that exploits the description of the cipher, and will give one multivariate equation, exploited by the main attack (Step 2.). The second step is already very fast.

Note: Our attacks are very general. For the original cipher Snow 1.0. itself, much faster attacks are known, see [10, 7, 27].

8.5 Application to a Modified Turing Cipher

Turing is a stream cipher proposed in 2003 by Rose and Hawkes [35]. It is a new kind of stream cipher which outputs many bits at a time, and in which the combiner is key-dependent. We have $n = 17 * 32 = 544$, $m = 5 * 32 = 160$, $l = 0$ (no memory), $k = 9 * 32 = 288$. These values are very large and any attack faster than the exhaustive search of all possible states $2^n = 2^{544}$ should be considered as interesting.

We will study a modified version of Turing, in which the combiner is NOT key-dependent. Then with Theorem 7.4 we get $M = \lceil (l+1)/m \rceil = 1$ and $d = \lceil kM/2 \rceil = 160$ that can be lowered to $d = 37$ and still satisfies the requirements of the Theorem 5.1 (for reasons explained in Section 7.5). This degree $d = 37$ is still by far too large to give any hope for practical attacks on Turing. We get an attack on modified Turing with time(Step 1.) $= 2^{805}$ and time(Step 2.) $= 2^{534}$. The second step is faster then the exhaustive search which would be in 2^{544}. The first step can also probably be improved to be faster than 2^{544}.

9 Extension to Ciphers With Unknown or Key Dependent Combiners

The results of this paper can be also applied to ciphers in which the combiner is only partially known (or key dependent). For example, at some place inside the combiner we XOR the data with the secret key. Or, we use Boolean functions with some coefficients being unknown or key-dependent. Let l' be the total number of unknown bits in the combiner with parameters (k, l, m). Then we may just consider this cipher as a cipher in which the combiner is known with parameters $(k, l + l', m)$: we just have l' additional memory bits that are not updated, they remain the same all the time. All the attacks described in the present paper will apply and when l' is not too big, or when $m \geq k$, they should be (relatively) efficient general attacks on stream ciphers.

10 Conclusion

In this paper we studied generic algebraic attacks on stream ciphers built with an LFSR and a combiner having a small number of memory bits. Our main result is that the complexity of algebraic attacks on stream ciphers will substantially decrease if the cipher outputs more bits at a time. We substantially extended and gave a much simpler proof of the important Theorem of [2]. Our new Theorem can be applied to substantially decrease the complexity of the best worse-case (generic) algebraic attack (whatever is the internal structure of the combiner component) for modified versions of four well known stream ciphers E0, LILI-128, Snow and Turing.

We demonstrated the existence of (yet another) very general tradeoff between speed and security of stream ciphers with (and without) memory.

Acknowledgements

The authors wish to thank Willi Meier and the anonymous reviewers of Crypto 2004, SAC 2004 and ICISC 2004, for careful reading and valuable 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,* Crypto 2003, LNCS 2729, pp. 162-176, Springer.
3. Frederik Armknecht: *Improving Fast Algebraic Attacks,* to appear in FSE 2004, LNCS, Springer.
4. Ross Anderson: *Searching for the Optimum Correlation Attack,* FSE'94, LNCS 1008, Springer, pp 137-143.
5. Elad Barkan, Eli Biham, and Nathan Keller: *Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication,* In Crypto 2003, LNCS 2729, pp: 600-616, Springer, 2003.
6. Bluetooth CIG, Specification of the Bluetooth system, Version 1.1, February 22 2001, available from www.bluetooth.com.
7. Christophe De Canniere, *Guess and Determine Attack on SNOW,* Nessie public report, 12/11/2001, NES/DOC/KUL/WP5/011/a, available from www.cryptonessie.org.
8. Claude Carlet, Emmanuel Prouff: *On a new notion of Nonlinearity relevant to multi-output pseudo-random generators,* SAC 2003, LNCS 3006, pp. 291-305, Springer 2004.
9. Will Meier, Enes Pasalic and Claude Carlet: *Algebraic Attacks and Decomposition of Boolean Functions,* In Eurocrypt 2004, pp. 474-491, LNCS 3027, Springer, 2004.
10. Don Coppersmith, Shai Halevi and Charanjit Jutla, Cryptanalysis of stream ciphers with linear masking, Crypto 2002, LNCS 2442, Springer, 2002. Available at http://eprint.iacr.org/2002/020/
11. Paul Camion, Claude Carlet, Pascale Charpin and Nicolas Sendrier, *On Correlation-immune Functions,* Crypto'91, LNCS 576, Springer, pp. 86-100.
12. Don Coppersmith, Shmuel Winograd: *Matrix multiplication via arithmetic progressions,* J. Symbolic Computation (1990), 9, pp. 251-280.
13. Nicolas Courtois: *The security of Hidden Field Equations (HFE)*, Cryptographers' Track Rsa Conference 2001, LNCS 2020, Springer, pp. 266-281.
14. Nicolas Courtois: *Higher Order Correlation Attacks, XL algorithm and Cryptanalysis of Toyocrypt,* ICISC 2002, LNCS 2587, pp. 182-199, Springer.
15. 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
16. Nicolas Courtois: *Fast Algebraic Attacks on Stream Ciphers with Linear Feedback,* Crypto 2003, LNCS 2729, pp: 177-194, Springer.
17. Nicolas Courtois: *The Inverse S-box, Non-linear Polynomial Relations and Cryptanalysis of Block Ciphers,* in AES 4 Conference, Bonn May 10-12 2004, LNCS, Springer.
18. Nicolas Courtois: *General Principles of Algebraic Attacks and New Design Criteria for Components of Symmetric Ciphers,* in AES 4 Conference, Bonn May 10-12 2004, LNCS, Springer.
19. 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/.

20. Patrik Ekdahl, Thomas Johansson, *SNOW - a new stream cipher,* Proceedings of First NESSIE Workshop, Heverlee, Belgium, 2000.
21. Patrik Ekdahl, Thomas Johansson, *A new version of the stream cipher SNOW,* in SAC 2002, LNCS 2595, Springer, pp. 47-61. Available from `http://www.it.lth.se/cryptology/snow/`
22. Jean-Charles Faugère: "A new efficient algorithm for computing Gröbner bases without reduction to zero (F5)" Workshop on Applications of Commutative Algebra, Catania, Italy, ACM Press, 2002.
23. Jovan Dj. Golic: *On the Security of Nonlinear Filter Generators,* FSE'96, LNCS 1039, Springer, pp. 173-188.
24. Jovan Dj. Golic: *Correlation Properties of a General Binary Combiner with Memory.* Journal of Cryptology vol. 9(2), pp. 111-126 (1996).
25. Jovan Dj. Golic, Vittorio Bagini, Guglielmo Morgari: *Linear Cryptanalysis of Bluetooth Stream Cipher,* Eurocrypt 2002, LNCS 2332, Springer, pp. 238-255.
26. Thomas Jakobsen: *Cryptanalysis of Block Ciphers with Probabilistic Non-Linear Relations of Low Degree,* Crypto 98, LNCS 1462, Springer, pp. 212-222, 1998.
27. Philip Hawkes, Gregory Rose: *Guess-and-determine attacks on SNOW,* in SAC 2002, LNCS 2595, Springer, pp. 37-46.
28. Philip Hawkes, Gregory Rose: *Rewriting Variables: the Complexity of Fast Algebraic Attacks on Stream Ciphers,* in Crypto 2004, LNCS 3152, pp. 390-406, Springer, 2004. Available from `eprint.iacr.org/2004/081/`.
29. Gregory L. Mayhew: *A Low cost high speed encryption system and method,* Proc. of 1994 IEEE Computer Society Press, pp. 147-154, 1994
30. Willi Meier and Othmar Staffelbach: *Fast correlation attacks on certain stream ciphers,* Journal of Cryptology, 1(3):159-176, 1989.
31. Willi Meier and Othmar Staffelbach: *Correlation Properties of Combiners with Memory in Stream Ciphers,* Journal of Cryptology 5(1): pp. 67-86 (1992).
32. Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone: *Handbook of Applied Cryptography*, Chapter 6, CRC Press.
33. Nessie Security Report v2.0. or Nessie deliverable D20, available from `www.cryptonessie.org`.
34. Jacques Patarin: *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88,* Crypto'95, Springer, LNCS 963, pp. 248-261, 1995.
35. Gregory G. Rose and Philip Hawkes: Turing: a Fast Stream Cipher, FSE 2003, LNCS, Springer.
36. Adi Shamir, Jacques Patarin, Nicolas Courtois, Alexander Klimov, *Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations*, Eurocrypt'2000, LNCS 1807, Springer, pp. 392-407.
37. L. Simpson, E. Dawson, J. Golic and W. Millan: *LILI Keystream Generator,* SAC'2000, LNCS 2012, Springer, pp. 248-261,
38. Volker Strassen: *Gaussian Elimination is Not Optimal,* Numerische Mathematik, vol 13, pp 354-356, 1969.
39. Muxiang Zhang, Agnes Chan: *Maximum Correlation Analysis of Nonlinear S-boxes in Stream Ciphers.* In Crypto 2000, LNCS 1880, pp. 501-514, Springer 2000.

A Another Proof of Theorem 4.1 in a Stronger Version

Theorem A.1 (Strong Version of the Special Case of Krause-Armknecht Thm.). Let F be an arbitrary fixed circuit with k binary inputs x_i, one bit of memory a, and one output y. Then, given $M = 2$ consecutive states $(t, t+1)$, there is a multivariate relation of degree k in the $x_j^{(i)}$, that relates only the input and the output bits, **without** any of the inner state/memory bits $a^{(t-1)}, a^{(t)}$:

$$R\left(x_0^{(t)}, \ldots, x_{k-1}^{(t)}\right) + y^{(t)} \cdot S\left(x_0^{(t)}, \ldots, x_{k-1}^{(t)}\right) +$$
$$+T\left(x_0^{(t+1)}, \ldots, x_{k-1}^{(t+1)}\right) + y^{(t+1)} \cdot U\left(x_0^{(t+1)}, \ldots, x_{k-1}^{(t+1)}\right) = 0.$$

Proof: We have $m = l = 1$. To prove this result, we will prove that basically the same type of result is true also for any m, provided that we have $m = l$ (or more).

This new theorem, will give the same M and the same d than our most general Theorem 7.4. Yet it exhibits equations that use much less monomials (and thus are easier to find).

Theorem A.2 (Strong Version of Theorem 5.1 when $m = l$). Let F be an arbitrary component with k binary inputs x_i, l bits of memory a, and m outputs y_i with $m = l$. Then, given $M = 2$ consecutive applications of the component $(t, t+1)$, there is a multivariate relation (being of degree k in $x_j^{(i)}$) of the form:

$$R\left(x_0^{(t)}, \ldots, x_{k-1}^{(t)}, y_0^{(t)}, \ldots, y_{k-1}^{(t)}\right) = S\left(x_0^{(t+1)}, \ldots, x_{k-1}^{(t+1)}, y_0^{(t+1)}, \ldots, y_{k-1}^{(t+1)}\right).$$

Proof: First we consider only one state, at time t. The output bits $y_i^{(t)}$ the next memory bits $a_i^{(t)}$ do depend only on the previously existing memory bits $a_i^{(t-1)}$, and the k inputs $\left(x_0^{(t)}, \ldots, x_{k-1}^{(t)}\right)$. We have 2^{k+l} possibilities. We consider the following matrix: Line are all the 2^{k+l} possibilities, and columns are all the 2^{k+l} monomials of type $\prod x_i^{(t)} \cdot \prod y_j^{(t)}$. We have now 2^{k+l} lines and also $2^{k+m} = 2^{k+l}$ columns.

Let $Q(x_0^{(t)}, \ldots, x_{k-1}^{(t)}, a_0^{(t-1)}, \ldots, a_{l-1}^{(t-1)})$ be an arbitrary function. We add one more column to the matrix, in which we put the values of $Q()$. We get the matrix with 2^{k+l} lines and $2^{k+l} + 1$ columns, and we know that a linear dependency must exist between the columns, whatever is the function $Q()$. In this linear dependency the last column $Q()$ may or may not appear, which we denote by $[Q(x_0^{(t)}, \ldots, x_{k-1}^{(t)}, a_0^{(t-1)}, \ldots, a_{l-1}^{(t-1)})]$. By definition, all the other columns are monomials, and their linear combination is a polynomial. Therefore, for every Boolean function Q, our dependency means that there exist a multivariate polynomial Rel_Q such that:

$$Rel_Q\left(x_0^{(t)},\ldots,x_{k-1}^{(t)},y_0^{(t)},\ldots,y_{k-1}^{(t)}\right)=\left[Q\left(x_0^{(t)},\ldots,x_{k-1}^{(t)},a_0^{(t-1)},\ldots,a_{l-1}^{(t-1)}\right)\right].$$

(As explained above, $[Q(\ldots)]$ means $\alpha Q(\ldots)$ with $\alpha=0$ or 1).

For any Q, we may by Gaussian reduction explicitly compute this polynomial Rel_Q for step t. We may put for example, for any i, $Q\left(x_0^{(t)},\ldots,x_{k-1}^{(t)},a_0^{(t-1)},\ldots,a_{l-1}^{(t-1)}\right)=a_i^{(t-1)}$ but we are also free to have $Q(\ldots)=a_j^{(t)}$ for any j. Indeed both the previous and the current memory bits are a deterministic function of these $k+l$ variables. (Hence, for example we may not put $Q(\ldots)=a_0^{(t+1)}$, as it depends on other variables.) Now, in our proof, for step t, we will use the first bit of the first inner state: $Q=a_0^{(t)}$. Then for $t+1$, we are allowed to use the same bit, since $a_0^{((t+1)-1)}=a_0^{(t)}$ (but for the following step we cannot put $Q=a_0^{(t)}$ anymore). Thus we will eliminate $a_0^{(t)}$.

More precisely, we do the following. Let $Q(\ldots)=a_0^{(t)}$, there exists $R=Rel_Q$ such that

$$R\left(x_0^{(t)},\ldots,x_{k-1}^{(t)},y_0^{(t)},\ldots,y_{k-1}^{(t)}\right)=\left[a_0^{(t)}\right]\qquad(R)$$

Now, for the next step $t+1$, let $Q'\left(x_0^{(t+1)},\ldots,x_{k-1}^{(t+1)},a_0^{(t)},\ldots,a_{l-1}^{(t)}\right)=a_0^{(t)}$, and we know that there is another polynomial $S=Rel_{Q'}$ such that:

$$S\left(x_0^{(t+1)},\ldots,x_{k-1}^{(t+1)},y_0^{(t+1)},\ldots,y_{k-1}^{(t+1)}\right)=\left[a_0^{(t)}\right]\qquad(S)$$

With these two equations (R) and (S) we may always eliminate $a_0^{(t)}$ and we get the claimed result. $\square$