Improving Fast Algebraic Attacks

Frederik Armknecht*

Theoretische Informatik
Universität Mannheim
68131 Mannheim, Germany
Armknecht@th.informatik.uni-mannheim.de

Abstract. An algebraic attack is a method for cryptanalysis which is based on finding and solving a system of nonlinear equations. Recently, algebraic attacks where found helpful in cryptanalysing LFSR-based stream ciphers. The efficiency of these attacks greatly depends on the degree of the nonlinear equations. At Crypto 2003, Courtois [8] proposed Fast Algebraic Attacks. His main idea is to decrease the degree of the equations using a precomputation algorithm. Unfortunately, the correctness of the precomputation step was neither proven, nor was it obvious.

The three main results of this paper are the following: First, we prove that Courtois' precomputation step is applicable for cryptographically reasonable LFSR-based stream ciphers. Second, we present an improved precomputation algorithm. Our new precomputation algorithm is parallelisable, in contrast to Courtois' algorithm, and it is more efficient even when running sequentially. Third, we demonstrate the improved efficiency of our new algorithm by applying it to the key stream generator E_0 from the Bluetooth standard. In this case, we get a theoretical speed-up by a factor of about 8, even without any parallelism. This improves the fastest attack known. Practical tests confirm the advantage of our new precomputation algorithm for the test cases considered.

Keywords: algebraic attacks, stream ciphers, linear feedback shift registers, Bluetooth

1 Introduction

Stream ciphers are designed for online encryption of secret plaintext bit streams $M = (m_1, m_2, \cdots)$, $m_i \in \mathrm{F}_2$, which have to pass an insecure channel. Depending on a secret key $\mathcal{K}$, the stream cipher produces a regularly clocked key stream $\mathcal{Z} = (z_1, z_2, \ldots)$, $z_i \in \mathrm{F}_2$, and encrypts M by adding both streams termwise over F_2. The legal receiver who uses the same stream cipher and the same key $\mathcal{K}$, decrypts the received message by applying the same procedure.

* This work has been supported by grant 620307 of the DFG (German Research Foundation)

Many popular stream ciphers are LFSR-based. They consist of some linear feedback shift registers (LFSRs) and an additional device, called the nonlinear combiner. An LFSR produces a sequence over F_2 depending on its initial state. They can be constructed very efficiently in hardware and can be chosen such that the produced sequence has a high period and good statistical properties. For these reasons, LFSR-based stream ciphers are widely used in cryptography.

A lot of different nontrivial approaches to the cryptanalysis of LFSR-based stream ciphers were discussed in the literature, e.g. fast correlation attacks (Meier, Staffelbach [18], Chepyzhov, Smeets [5], Johansson, Joensson [12, 13]), backtracking attacks (Golic [10], Zenner, Krause, Lucks [24], Fluhrer, Lucks [9], Zenner [23]), time-space tradeoffs (Biryukov, Shamir [4]), BDD-based attacks (Krause [15]) etc. For such stream ciphers many corresponding design criteria (correlation immunity, large period and linear complexity, good local statistics etc.) were developed, e.g. Rueppel [20]. Recently, a new kind of attack was proposed: algebraic attack. For some ciphers, algebraic attacks outmatched all previously known attacks (e.g. Courtois, Meier [7], Armknecht, Krause [1], Courtois [6]).

An algebraic attack consists of two steps: First find a system of equations in the bits of the secret key $\mathcal{K}$ and the output bits z_t. If the combiner is memoryless, the methods of [7] can be used. A general method, which applies to combiners with memory too, was presented in [1]. If enough low degree equations and known key stream bits are given, the secret key $\mathcal{K}$ can be recovered by solving this system of equations in a second step. For this purpose, several methods (Linearization, XL, XSL, Groebner bases, ...) exist. Most of them run faster if the degree of the equations is low. Hence, the search for systems of low degree equations is a desirable goal in algebraic attacks.

Having this in mind, fast algebraic attacks were introduced by Courtois at Crypto 2003 [8], using a very clever idea. Before solving the system of equations, the degree of the system of equations is decreased in a precomputation step. The trick is to eliminate all high degree monomials independent of the key stream bits which has to be done only once. For this purpose, an algorithm A was proposed, using the well known Berlekamp-Massey algorithm [17, 19]. Unfortunately, the correctness of A was neither proven, nor is it obvious. Therefore, the applicability of fast algebraic attacks on LFSR-based stream ciphers in general was still an open question.

In this paper, we finally give a positive answer to this question. We present a non-trivial proof that fast algebraic attacks work under rather weak assumptions which are satisfied by all LFSR-based stream ciphers we know. In particular, it is not necessary that the periods are pairwise co-prime which is not true in general (e.g., see the cipher used in the Bluetooth standard). To get the result, we prove some statements about minimal polynomials of linear recurring sequences. To the best of our knowledge, these statements have not been published elsewhere in open literature. Thus, our results may be of independent interest.

In general, an attacker has knowledge of the whole cipher except the secret key. Hence, it seems to be logical to exploit these information to perform the precomputation step more directly, reducing the number of operations. This was the motivation for developing a new algorithm B. Contrary to A, which runs strictly sequential, B can be performed partly in parallel. Thus, our new method improves the efficiency of fast algebraic attacks significantly.

We demonstrate this by applying B to the E_0 key stream generator used in the Bluetooth standard for wireless communication. A theoretical examination shows that our algorithm has a speed-up factor of about 8, even without any parallelism. This improves the fastest known attack against the E_0 cipher. Practical tests on reduced versions of E_0 confirm the advantage of B for the test cases considered.

The paper is organized as follows: in Section 2 we describe fast algebraic attacks. In Section 3 we give the missing correctness proof for fast algebraic attacks. In Section 4 we improve fast algebraic attacks by introducing a new precomputation step. In Section 5 we demonstrate the higher efficiency of our new method in theory and practice. Section 6 concludes the paper.

2 Fast Algebraic attacks

Let $\mathcal{Z} := (z_t) := (z_t)_{t=0}^{\infty}$ be the key stream produced by an LFSR-based key stream generator, using a secret key $\mathcal{K} \in \{0,1\}^n$. $\mathcal{K}$ is defined as the initial states of the used LFSRs. At each clock t, the internal state of the cipher consists of $L^t(\mathcal{K})$ and some additional memory bits[1], where $L : \{0,1\}^n \to \{0,1\}^n$ is a linear boolean function known to the attacker. An algebraic attack works as follows:

The first step is to find a boolean function $\hat{F} \neq 0$ such that for an integer $\delta \geq 0$ the equation

$$\hat{F}(L^t(\mathcal{K}), \ldots, L^{t+\delta}(\mathcal{K}), z_t, \ldots, z_{t+\delta}) = 0 \tag{1}$$

is true for all clocks t. If the cipher is memoryless, the methods described in [7] can be used. Unfortunately, these methods neither work with ciphers using memory, nor are they guaranteed to find equations with the lowest degree. Therefore, in general the method proposed in [1] is the better choice. It can be applied to every cipher and finds for certain the equations with the lowest degree.

The second step is to use (1) to get a system of equations describing $\mathcal{K}$ in dependence of the observed key stream $\mathcal{Z}$. For each row $z_t, \ldots, z_{t+\delta}$ of output bits known to the attacker, replace these values in (1). The result is a valid equation in the bits of the secret key $\mathcal{K}$.

The third and final step is to recover the secret key $\mathcal{K}$ by solving this system of equations. One possibility to do so is the linearization method. Due to the linearity of L, all equations (1) have a degree $\leq \deg \hat{F}$. Therefore, the number m of different monomials occurring is limited. If the attacker has enough known key stream bits at his disposal the number of linearly independent equations equals m. By substituting each monomial by a new variable, the attacker gets

[1] Where zero memory bits are possible.

a linear system of equations in m unknowns which can be solved by Gaussian elimination or more refined methods like the one by Strassen [22].

In general, m will be about $\binom{n}{d}$. Hence, the lower the degree d the more efficient the attack. Therefore, an attacker using an algebraic attack will always try to find a system of low degree equations. A very clever approach, called "fast algebraic attack", to decrease the degree of a given system of equations was presented in [8]. We will discuss this method now. Suppose that (1) can be rewritten as

$$\begin{aligned} 0 &= \hat{F}(L^t(\mathcal{K}), \ldots, L^{t+\delta}(\mathcal{K}), z_t, \ldots, z_{t+\delta}) \\ &= F(L^t(\mathcal{K}), \ldots, L^{t+\delta}(\mathcal{K})) + G(L^t(\mathcal{K}), \ldots, L^{t+\delta}(\mathcal{K}), z_t, \ldots, z_{t+\delta}) \\ &=: F_t(\mathcal{K}) + G_t(\mathcal{K}, \mathcal{Z}) \end{aligned} \tag{2}$$

where the degree e of G in $\mathcal{K}$ is lower than the degree d of $\hat{F}$.[2] Furtheron, we assume that the attacker knows coefficients $c_0, \ldots, c_{T-1} \in \{0,1\}$ such that

$$\sum_{i=0}^{T-1} c_i \cdot F_{t+i}(\mathcal{K}) = 0 \quad \forall t, \mathcal{K}. \tag{3}$$

Using (2) and (3), we get by $\sum_{i=0}^{T-1} c_i \cdot G_{t+i}(\mathcal{K}, \mathcal{Z}) = 0$ an equation in $\mathcal{K}$ and $\mathcal{Z}$ with a lower degree $e < d$. Therefore, the attacker can reduce the system of equations given by (1) with degree d into a new system of equations of degree $e < d$ where all equations are of the type $\sum c_i G_{t+i}$. Note, that this improves the third step enormously, but requires more known key stream bits z_t to perform step two.

Of course, it is vital for the whole approach that such coefficients $c_0, \ldots, c_{T-1}$ can be found efficiently. In [8], the following algorithm A was proposed:

1. Chose a reasonable[3] key $\hat{\mathcal{K}}$ and compute $\hat{z}_t := F_t(\hat{\mathcal{K}})$ for $t = 1, \ldots, 2T$.
2. Apply the Berlekamp-Massey algorithm to find $c_0, \ldots, c_{T-1}$ with

$$\sum_{i=0}^{T-1} c_i \cdot F_{t+i}(\hat{\mathcal{K}}) = 0 \quad \forall t. \tag{4}$$

It is known that the Berlekamp-Massey algorithm finds coefficients with the smallest value of T fulfilling (4). This needs about $\mathcal{O}(T^2)$ basic operations. Together with the first step, algorithm A has to perform about $\mathcal{O}(T^2 + 2T|\mathcal{K}|)$ steps. In general, the exact value of T is unknown but an upper bound is the maximum number of different monomials occurring.

The result of algorithm A is correct if (4) implies (3) which has not been proven in [8]. The only correctness argument indicated there was based on the

---

[2] For example, this assumption is true for the three ciphers E_0, Toyocrypt and LILI-128.

[3] In [8], $\hat{\mathcal{K}}$ can be any value in $\{0,1\}^n$. But if one of the LFSRs is initialised with the all-zero state, the algorithm returns a wrong result in most cases. Therefore, $\hat{\mathcal{K}}$ has to be chosen such that the initial states are all non-zero.

assumption that the sequences produced by the LFSRs have pairwise co-prime periods. But this is not true in general. A counter-example is the key stream generator E_0 used in the Bluetooth standard for wireless communication: the two periods $2^{33}-1$ and $2^{39}-1$ share the common factor 7.

On the other hand algorithm A does not work correctly in general without any preconditions. An example is given in appendix D. This raises the question which preconditions are necessary for the correctness of algorithm A and if they are fulfilled by E_0.

One of the major achievements from this paper is to show the correctness of algorithm A under weaker assumptions. As we are not aware of any LFSR-based stream cipher for which these conditions are not true (including E_0), we assume that fast algebraic attacks as described in [8] can be mounted against most LFSR-based stream ciphers discussed in public.

3 Proof of Correctness

In this section, we prove the correctness of algorithm A under cryptographically reasonable assumptions. First, we repeat some known facts about linear recurring sequences.

Theorem 1. *(Lidl, Niederreiter [16]) A sequence $\mathcal{Z} = (z_t)$ over F_2 is called a linear recurring sequence if coefficients $c_0, \ldots, c_{T-1} \in \{0,1\}$ (not all zero) exist such that $\sum c_i z_{t+i} = 0$ is true for all values $t \geq 1$. In this case, $\sum c_i x^i \in \mathrm{F}_2[x]$ is called a characteristic polynomial of the sequence $\mathcal{Z}$. Amongst all characteristic polynomials of $\mathcal{Z}$ exists one unique polynomial $min(\mathcal{Z})$ which has the lowest degree. We will call it the minimal polynomial of $\mathcal{Z}$. A polynomial $f(x) \in \mathrm{F}_2[x]$ is a characteristic polynomial of $\mathcal{Z}$ if and only if $min(\mathcal{Z})$ divides $f(x)$.*

From now on, a sequence $\mathcal{Z}$ will always stand for a linear recurring sequence. Furtheron, we will denote by $\overline{\mathrm{F}_2}$ the algebraic closure of the field F_2.[4] By the roots of $f(x)\mathrm{F}_2[x]$, we will always mean the roots in $\overline{\mathrm{F}_2}$.

Definition 1. *Let $R_1, \ldots, R_\kappa \subseteq \overline{\mathrm{F}_2}$ be pairwise disjunct, $R := R_1 \,\dot\cup\, \ldots \,\dot\cup\, R_\kappa$. We say that a pair of vectors $(\alpha_1, \ldots, \alpha_n) \in R^n$, $(\beta_1, \ldots, \beta_m) \in R^m$ factorizes uniquely over $R_1, \ldots, R_\kappa$ if the following holds*

$$\alpha_1 \cdot \ldots \cdot \alpha_n = \beta_1 \cdot \ldots \cdot \beta_m \Rightarrow \prod_{\alpha_i \in R_l} \alpha_i \cdot \prod_{\beta_j \in R_l} (\beta_j)^{-1} = 1, \quad 1 \leq l \leq \kappa$$

For a monomial $\mu = \prod_{j=1}^{k} x_{i_j} \in \mathrm{F}_2[x_1, \ldots, x_n]$ with $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ and $\alpha = (\alpha_1, \ldots, \alpha_n) \in R^n$, we define the vector $\overrightarrow{\mu(\alpha)} := (\alpha_{i_1}, \ldots, \alpha_{i_k}) \in R^k$.

Example Set $R_1 := \{\alpha, \alpha\beta\}$ and $R_2 := \{\beta\}$ with $\beta \neq 1$. The pair of vectors (α, β) and $(\alpha\beta)$ does not factorize uniquely over R_1, R_2 because of $\alpha \cdot \beta = \alpha\beta$

[4] I.e., $\overline{\mathrm{F}_2}$ is the smallest field such that $\mathrm{F}_2 \subset \overline{\mathrm{F}_2}$ and each polynomial $f(x) \in \mathrm{F}_2[x]$ has at least one root in $\overline{\mathrm{F}_2}$.

but $\alpha \cdot (\alpha\beta)^{-1} = \beta^{-1} \neq 1$.

The motivation for this definition is that we need in our main theorem that certain products of roots of minimal polynomials are unique in the sense above (see appendix A. The main theorem of our paper is:

Theorem 2. *Let $\mathcal{Z}_1 = (z_t^{(1)}), \ldots, \mathcal{Z}_\kappa = (z_t^{(\kappa)})$ be sequences with pairwise co-prime minimal polynomials which have only non-zero roots. Let R_i denote the set of roots of $min(\mathcal{Z}_i)$ in $\overline{\mathrm{F}_2}$, $F : \mathrm{F}_2^n \to \mathrm{F}_2$ be an arbitrary boolean function and $I := (i_1, \ldots, i_n) \in \{1, \ldots, \kappa\}^n$ and $\delta := (\delta_1, \ldots, \delta_\kappa) \in \mathrm{N}^\kappa$ be two vectors.*

We set $R := R_{i_1} \times \ldots \times R_{i_n}$ and divide $F = \sum \mu_i$ into a sum of monomials. Furtheron, for arbitrary $d := (d_1, \ldots, d_\kappa) \in \mathrm{N}^\kappa$ the sequences $\mathcal{Z} := (z_t)$ and $\mathcal{Z}^{(d)} := (z_t^{(d)})$ are defined by

$$z_t := F(z_{t+\delta_1}^{(i_1)}, \ldots, z_{t+\delta_n}^{(i_n)}),\ z_t^{(d)} := F(z_{t+\delta_1+d_{i_1}}^{(i_1)}, \ldots, z_{t+\delta_n+d_{i_n}}^{(i_n)})$$

If all pairs of vectors $\overrightarrow{\mu_i(\alpha)}, \overrightarrow{\mu_j(\alpha')}$ with $\alpha, \alpha' \in R$ factorize uniquely over $R_1, \ldots, R_\kappa$, then $min(\mathcal{Z}) = min(\mathcal{Z}^{(d)})$.

What is the connection to algorithm A? From the theory of LFSRs, it can be easily argued that the sequence $\hat{\mathcal{Z}} = (\hat{z}_t)$ from algorithm A is a linear recurring sequence and that $\hat{c}_0, \ldots, \hat{c}_{T-1}$ correspond to its minimal polynomial m. $\hat{\mathcal{Z}}$ is produced in the way sequence $\mathcal{Z}$ is described in theorem 2 which assures that the minimal polynomial m remains unchanged if we shift each of the sequences produced by the LFSRs individually. As in the general the produced sequences have maximal period, the minimal polynomial found by algorithm A is the same for each possible key $\mathcal{K}$. In appendix B we show that the conditions of theorem 2 are satisfied for a large class of LFSR-based ciphers automatically, independent of F,I and δ. Before we can prove theorem 2, we need some statements about minimal polynomials.

Theorem 3. *([16], Theorem 6.21) Let $\mathcal{Z} = (z_t)$ be a sequence with characteristic polynomial $f(x) = \prod_{i=1}^n (x - \alpha_i)$ where the roots lie in $\overline{\mathrm{F}_2}$. If the roots $\alpha_1, \ldots, \alpha_n$ are all distinct, i.e. each root has multiplicity one, then for each t, z_t can be expressed in the following way:*

$$z_t = \sum_{i=1}^{n} A_i \alpha_i^t$$

where $A_1, \ldots, A_t \in \overline{\mathrm{F}_2}$ are uniquely determined by the initial values of the sequence $\mathcal{Z}$.

Theorem 4. *Let $\mathcal{Z} = (z_t)$ be a sequence with $z_t = \sum_{i=1}^n A_i \alpha_i^t$ with pairwise distinct elements $\alpha_i \in \overline{\mathrm{F}_2}$ and non-zero coefficients A_i. Let $m(x) \in \mathrm{F}_2[x]$ be the polynomial with the lowest degree such that $m(\alpha_i) = 0$ for $1 \leq i \leq n$. Then $m(x)$ is the minimal polynomial $min(\mathcal{Z})$. In particular, each root of $min(\mathcal{Z})$ has multiplicity one.*

Proof. We show that $f(x) \in \mathrm{F}_2[x]$ is a characteristic polynomial of $\mathcal{Z}$ if and only if $f(\alpha_i) = 0$ for all i. Thus, $m(x)$ is the characteristic polynomial with the lowest degree what is the definition of $min(\mathcal{Z})$. Let $f(x) = \sum_{k=0}^{r} c_k x^k$. Then for each t, we have

$$\sum_{k=0}^{r} c_k z_{t+k} = \sum_{k=0}^{r} c_k \left(\sum_{i=1}^{n} A_i \alpha_i^{t+k} \right) = \sum_{i=1}^{n} \left(A_i \sum_{k=0}^{r} c_k \alpha_i^k \right) \alpha_i^t = \sum_{i=1}^{n} \left(A_i f(\alpha_i) \right) \alpha_i^t$$

For $1 \le i \le n$ and $0 \le t \le n-1$, let $M := (\alpha_i^t)$ be a Vandermonde-matrix of size $n \times n$. As the elements α_i are pairwise distinct, M is regular. Thus, the expression above equals to zero for each t if and only if $(A_1 f(\alpha_1), \ldots, A_n f(\alpha_n)) \in \{0,1\}^n$ is an element of the kernel of M, i.e. $A_i f(\alpha_i) = 0$. As the coefficients A_i were assumed to be non-zero, this is equivalent to $f(\alpha_i) = 0$ for all i. □

Proof of theorem 2. By theorem 4 all roots in R_i have multiplicity one. Therefore, by theorem 3, each sequence $\mathcal{Z}_i$ can be expressed by $z_t^{(i)} = \sum_{\alpha \in R_i} A_\alpha \alpha^t$ with <u>unique</u> coefficients A_α. For each i it holds

$$z_{t+\delta_i}^{(i)} = \sum_{\alpha \in R_i} A_\alpha \alpha^{t+\delta_i} = \sum_{\alpha \in R_i} \left(A_\alpha \alpha^{\delta_i} \right) \alpha^t$$

and therefore

$$z_t = F(\sum_{\alpha \in R_{i_1}} (A_\alpha \alpha^{\delta_{i_1}}) \alpha^t, \ldots, \sum_{\alpha \in R_{i_n}} (A_\alpha \alpha^{\delta_{i_n}}) \alpha^t)$$

We set $\mathcal{P} := \{\mu_i(\alpha) | \alpha \in R, 1 \le i \le l\}$. The sequences $\mathcal{Z}$ and $\mathcal{Z}^{(d)}$ can be expressed by

$$z_t = \sum_{\pi \in \mathcal{P}} A_\pi \pi^t, \quad z_t^{(d)} = \sum_{\pi \in \mathcal{P}} A_\pi^{(d)} \pi^t$$

with <u>unique</u> coefficients A_π and $A_\pi^{(d)}$. We show that A_π is non-zero if and only if $A_\pi^{(d)}$ is non-zero. Then the equality of $min(\mathcal{Z})$ and $min(\mathcal{Z}^{(d)})$ follows by theorem 4.

We express the coefficients A_π and $A_\pi^{(d)}$ in dependence of the coefficients A_α. For $\pi = \alpha_1 \cdot \ldots \cdot \alpha_m \in \mathcal{P}$, $\alpha_i \in \dot{\bigcup} R_i$, we define by π^d the product $\left(\prod_{\alpha_i \in R_1} \alpha_i^{d_1} \right) \cdot \ldots \cdot \left(\prod_{\alpha_i \in R_\kappa} \alpha_i^{d_\kappa} \right)$. As all pairs of vectors $\overrightarrow{\pi_1}, \overrightarrow{\pi_2}$ factorize uniquely over $R_1, \ldots, R_\kappa$, this expression is independent of the factorization $\alpha_1 \cdot \ldots \cdot \alpha_m$. Analogously, for $\alpha = (\alpha_1, \ldots, \alpha_n) \in R$ we set $\alpha^d := (\alpha_1^{d_{i_1}}, \ldots, \alpha_n^{d_{i_n}})$. With these definitions, we get

$$z_t = \sum_{\pi \in \mathcal{P}} \underbrace{\sum_{\mu_i} \sum_{\substack{\alpha \in \mathcal{R} \\ \mu_i(\alpha) = \pi}} \mu_i(A_\alpha)}_{= A_\pi} \pi^t$$

where $A_\alpha = (A_{\alpha_{i_1}}, \ldots, A_{\alpha_{i_n}})$. Therefore, the coefficients $A_\pi^{(d)}$ can be expressed by

$$A_\pi^{(d)} = \sum_{\mu_i} \sum_{\substack{\alpha \in \mathcal{R} \\ \mu_i(\alpha)=\pi}} \mu_i(A_\alpha \alpha^d) = \sum_{\mu_i} \sum_{\substack{\alpha \in \mathcal{R} \\ \mu_i(\alpha)=\pi}} \mu_i(A_\alpha)\mu_i(\alpha^d) = \sum_{\mu_i} \sum_{\substack{\alpha \in \mathcal{R} \\ \mu_i(\alpha)=\pi}} \mu_i(A_\alpha)\pi^d$$
$$= A_\pi \cdot \pi^d$$

As the roots of $m_i(x)$ are all non-zero by assumption, it is $\pi^d \neq 0$. Therefore, $A_\pi \neq 0$ iff $A_\pi^{(d)} \neq 0$. □

The proof shows why a precondition is necessary for the correctness of the pre-computation step. Otherwise, it could happen that for some π it is $A_\pi \neq 0$ but $A_\pi^{(d)} = 0$ (or vice versa). In this cases, the corresponding minimal polynomials could be different (see appendix D).

4 Improvements

In this section, we show how algorithm A can be improved. Let $min(F)$ denote the (unique) minimal polynomial found by algorithm A. Until now, we made no use of the knowledge of the minimal polynomials $m_1(x), \ldots, m_\kappa(x)$. The idea is to compute $min(F)$ and/or the parameter T more or less directly from the known minimal polynomials and F. For this purpose, we cite some statements about minimal polynomials.

Definition 2. *Consider two co-prime polynomials $f(x) = \prod_{i=1}^n (x - \alpha_i)$ and $g(x) = \prod_{j=1}^m (x - \beta_j)$ with no multiple roots. Then we define*

$$f(x) \otimes g(x) := \prod_{i,j} (x - \alpha_i \beta_j), \quad f(x) \otimes f(x) := \prod_{1 \leq i < j \leq n} (x - \alpha_i \alpha_j) \cdot f(x).$$

Theorem 5. *([16], Th. 6.57 + 6.67) Let $\mathcal{Z}_1 = (z_t^{(1)}), \ldots, \mathcal{Z}_\kappa = (z_t^{(\kappa)})$ be sequences with pairwise co-prime $min(\mathcal{Z}_i)$. Then*

$$min(\mathcal{Z}_1 + \ldots + \mathcal{Z}_\kappa) = min(\mathcal{Z}_1) \cdot \ldots \cdot min(\mathcal{Z}_\kappa)$$
$$min(\mathcal{Z}_i \cdot \mathcal{Z}_j) = min(\mathcal{Z}_i) \otimes min(\mathcal{Z}_j), \quad \forall i \neq j$$

where $\mathcal{Z}_1 + \ldots + \mathcal{Z}_\kappa := (z_t^{(1)} + \ldots + z_t^{(\kappa)})$ and $\mathcal{Z}_i \cdot Z_j := (z_t^{(i)} \cdot z_t^{(j)})$.

Theorem 6. *(Key [14], Th. 1) Let $\mathcal{Z} = (z_t)$ be a sequence and $l := \deg(min(\mathcal{Z}))$. If d is an integer with $1 \leq d < l$, then the sequence $(z_t \cdot z_{t+d})$ has the minimal polynomial $min(\mathcal{Z}) \otimes min(\mathcal{Z})$ of degree $\frac{l(l+1)}{2}$.*

Before we proceed further, we will take a closer look on the complexities of the operators $''\cdot''$ and $''\otimes''$.

Theorem 7. *(Schoenhage [21]) Let two polynomials $f(x)$ resp. $g(x)$ of $\mathrm{F}_2[x]$ be given of degrees $\leq m$. Then, the product $f(x) \cdot g(x)$ can be computed with an effort of $\mathcal{O}(m \log m \log\log m)$.*

Theorem 8. *(Bostan, Flajolet, Salvy, Schost [3], Theorem 1) Let $f(x)$ resp. $g(x)$ be two co-prime polynomials of degree n resp. m with no multiple roots. Then the polynomial $f(x) \otimes g(x)$ can be computed directly within*

$$\mathcal{O}(\underbrace{nm \log^2(nm/2) \log\log(nm/2) + nm \log(nm) \log\log(nm)}_{=:\mathcal{T}(nm)})$$

operations in F_2 *without knowing the roots of $f(x)$ or $g(x)$.*

This implies a kind of divide-and-conquer approach for computing the minimal polynomial from F. The trick is to split the function F into two or more functions $F_1, \ldots, F_l$ such that the corresponding minimal polynomials $min(F_i)$ are pairwise co-prime. Then by theorem 5 it holds $min(F) = min(F_1) \cdot \ldots \cdot min(F_l)$. In some cases, such a partition can be hard to find or may not even exist. If the minimal polynomials $min(F_i)$ are not pairwise co-prime the product $p(x) := min(F_1) \cdot \ldots \cdot min(F_l)$ is a characteristic polynomial of each possible sequence $\hat{\mathcal{Z}}$, i.e. the coefficients of $p(x)$ fulfill equation (3) also. Therefore, using $p(x)$ makes a fast algebraic attack possible though it may require more known key stream bits than really necessary.

We compare the effort of this approach to that of algorithm A. For simplicity we assume $l = 2$, i.e. $min(F) = min(F_1) \cdot min(F_2)$. Let $T_1 := \deg(min(F_1)) \leq \deg(min(F_2)) =: T_2$. Then $\deg(min(F)) = T_1 + T_2$. As said before, algorithm A needs about

$$\mathcal{O}(T_1^2 + T_2^2 + 2(T_1 + T_2)|\mathcal{K}| + 2T_1T_2))$$

basic operations. Instead of using algorithm A, we can do the following: First compute $min(F_1)$ and $min(F_2)$. In general, this can be done with algorithm A or in some cases by using the $\otimes$-product (see details later). If we use algorithm A, the complexity of these operations are $\mathcal{O}(T_1^2 + 2T_1)$ resp. $\mathcal{O}(T_2^2 + 2T_2)$. Notice that both operations can be performed in parallel. Having computed $min(F_1)$ and $min(F_2)$, the second and final step consists of computing the product $min(F_1) \cdot min(F_2) = min(F)$. By theorem 7, this has an effort of $\mathcal{O}(T_2 \log T_2 \log\log T_2)$ which implies an overall effort of

$$\mathcal{O}(T_1^2 + T_2^2 + 2(T_1 + T_2)|\mathcal{K}| + T_2 \log T_2 \log\log T_2)$$

In general, it is $\log T_2 \log\log T_2 \ll 2T_1$. Thus, our new approach has a lower runtime than algorithm A. The advantage increases if F can be divided in more than two parts.

In some cases, the precomputation step can be improved even a bit further. Assume that at least one of the F_i mentioned above can be written as a product $F_i = G_1 \cdot G_2$ such that $min(G_1)$ and $min(G_2)$ are co-prime. This is for example almost always the case when F_i is a monomial. Then, by theorem 5 it holds $min(F_i) = min(G_1) \otimes min(G_2)$ which implies a similar strategy. Let again be $T_1 := \deg(min(G_1)) \leq \deg(min(G_2)) =: T_2$. Using algorithm A would need about

$$\mathcal{O}(T_1^2 T_2^2 + 2T_1T_2|\mathcal{K}|)$$

operations. Instead, we can we compute $min(G_1)$ and $min(G_2)$ with algorithm A in a first step. This takes $\mathcal{O}(T_1^2 + T_2^2 + 2(T_1 + T_2)|\mathcal{K}|)$ operations. If $min(G_1)$ and/or $min(G_2)$ are already known, than this step can be omitted. This is for example the case if F_i is the product of the output of two or several distinct LFSRs (see the E_0 example in appendix C). In the second step, we use the algorithm described in [3] to compute $min(G_1) \otimes min(G_2)$. The effort is $\mathcal{O}(\mathcal{T}(T_1 T_2))$ which is in $\mathcal{O}(T_1 T_2 \log^2(T_1 T_2) \log\log(T_1 T_2))$. Altogether, this approach needs

$$\mathcal{O}(T_1 T_2 \log^2(T_1 T_2) \log\log(T_1 T_2) + T_1^2 + T_2^2 + 2(T_1 + T_2)|\mathcal{K}|)$$

operations. This shows the improvement. If we perform the operations of the first step in parallel, the time needed to get the result can be decreased further. In the following, we summarize our approaches by proposing the following new algorithm B:

Algorithm B

Given: Pairwise co-prime primitive polynomials $m_1(x), \ldots, m_k(x)$, an arbitrary boolean function F as described in section 2 and a partition $F = F_1 + \ldots + F_l$ such that the minimal polynomials $min(F_i)$ are pairwise co-prime

Task: Find $min(F)$

Algorithm:

- Compute the minimal polynomials $min(F_i)$ by using algorithm A. This can be done in parallel.
  If $F_i = G_1 \cdot G_2$ with co-prime $min(G_1)$ and $min(G_2)$ (e.g., F_i is a monomial), then the $\otimes$-product algorithm described in [3] can be used.
- Compute $min(F) = min(F_1) \cdot \ldots \cdot min(F_l)$ using the algorithm in [21].

5 Application to the E_0 key stream generator

In this section, we demonstrate the efficiency of algorithm B on the E_0 key stream generator which is part of the Bluetooth standard for wireless communication [2]. It uses four different LFSRs with pairwise co-prime primitive minimal polynomials m_1, m_2, m_3, m_4 of degrees $T_1 = 25$, $T_2 = 31$, $T_3 = 33$ and $T_4 = 39$. The sequences produced by the LFSRs have the periods $2^{T_1} - 1, \ldots, 2^{T_4} - 1$. As the values $2^{T_3} - 1$ and $2^{T_4} - 1$ share the common factor 7, the assumption made in [8] is not satisfied.

In [1], a boolean function $\hat{F}$ was developed fulfilling equation (1). $\hat{F}$ can be divided as shown in (2) into $F + G$ with $\deg(F) = 4 > 3 = \deg(G)$. The function F is

$$F = \sum_{\substack{1 \le i < j \le 4 \\ 1 \le k < l \le 4}} z_t^{(i)} z_t^{(j)} z_{t+1}^{(k)} z_{t+1}^{(l)} + z_t^{(1)} z_t^{(2)} z_t^{(3)} z_t^{(4)}$$

where $\mathcal{Z}_i = (z_t^{(i)})$ is the sequence produced by LFSR i. Let R_i be the set of roots of m_i. Inspired by the definition in theorem 2, we define

$$\mathcal{P} := \{\alpha_i\alpha_j\alpha_k\alpha_l |\ \alpha_s \in R_s, 1 \leq i < j \leq 4, 1 \leq k < l \leq 4\} \cup \{\alpha_1\alpha_2\alpha_3\alpha_4 \mid \alpha_i \in R_i\}.$$

We have checked with Maple that for all pairs $\pi_1, \pi_2 \in \mathcal{P}$ the pair of vectors $\vec{\pi_1}$ and $\vec{\pi_2}$ factorize uniquely over the union $R_1 \dot{\cup} R_2 \dot{\cup} R_3 \dot{\cup} R_4$. Hence, the weaker assumptions from theorem 2 are fulfilled here and a unique minimal polynomial $min(F)$ exists. Furtheron, it can be showed that F can be written as $F = F_1 + \ldots + F_{11}$ such that the minimal polynomials $min(F_i)$ are pairwise co-prime (see appendix C). By theorem 5, $min(F)$ can be expressed by

$$min(F) = \prod_{i=1}^{11} min(F_i) \tag{5}$$

The degree of $min(F)$ and thus the parameter T is upper bounded by

$$\sum_{1 \leq i < j \leq 4} \frac{T_i(T_i+1)T_j(T_j+1)}{4} + \sum_{1 \leq i < j < k \leq 4} T_iT_jT_k \frac{T_i + T_j + T_k - 1}{2} + T_1T_2T_3T_4$$

In [2], the degrees T_1, T_2, T_3, T_4 are defined as 25, 31, 33, 39 respectively. Thus, the degree of $min(F)$ is $\leq 8.822.188 \approx 2^{23.07}$. The computation of $min(F)$ using algorithm A would need most about $2^{46.15}$ basic operations.

Equation (5) implies the usage of algorithm B. In the first step we apply algorithm A to compute the minimal polynomials $min(F_i)$, $i = 1, \ldots, 11$. This takes an overall number of basic operations of $\approx 2^{43.37}$. Exploiting parallelism, we have only to <u>wait</u> the time needed to perform $\approx 2^{41.91}$ basic operations.[5]

The second step is the computation of the product of these minimal polynomials. Here, this takes altogether about $\approx 2^{28.25}$ basic operations. Performed in sequential, algorithm B needs about $2^{43.37}$ basic operations which is almost 8 times faster than algorithm A. If we exploit the parallelism mentioned above, the number of basic operations we have to wait is about $2^{41.91}$ which is more than 16 times faster than in algorithm A. This improves the fastest known attack against the E_0 cipher. Table 1 sums up the fastest previous attacks known:

Table 1. Fastest previous attacks against E_0

Attack	Data	Memory	Pre-computation	Attack Complexity
[1]	2^{24}	2^{48}		2^{68}
[8]	2^{24}	2^{37}	2^{46}	2^{49}
new (sequential)	2^{24}	2^{37}	2^{43}	2^{49}
new (parallel)	2^{24}	2^{37}	2^{42}	2^{49}

[5] Of course, the number of operations remains unchanged.

An ad-hoc implementation in Maple (without using parallelism and the algorithm of [3]), applied to reduced versions of E_0 with shorter LFSRs, confirmed the improved efficiency of our new algorithm. The results can be found in table 2. In the first four columns, the coefficients of the four minimal polynomials are given. The next two columns show the time consumptions of algorithm A and B respectively which are compared in the last column. In all cases, our new

Table 2. Comparison of algorithm A and B on reduced versions of E_0

C_1	C_2	C_3	C_4	Algorithm A	Algorithm B	A/B
110	1100	10100	1000100	10 h 41 m 43 s	12 m 3 s	53.32
101	1001	10010	1000001	11 h 2 m 49 s	12 m 7 s	54.75
				10 h 50 m 0 s	11 m 59 s	54.30
				10 h 52 m 59 s	11 m 55 s	54.86
				10 h 53 m 31 s	11 m 58 s	54.65
110	1100	10100	10100010100	78 h 30 m 16 s	1 h 43 m 25 s	45.55
101	10100	1100000	11100010000	18 d 18 h 26 m 0 s	13 h 50 m 7 s	32.56

algorithm B was significantly faster than algorithm A, even without using parallelism. The speed-up factor was much higher than predicted theoretically and depended on the chosen minimal polynomials and the initial states.

In all cases, the degree of $min(F)$ was equal to the upper bound estimated on page 11. Hence, we expect that the upper bound is tight for the real E_0 key stream generator also.

6 Conclusion

In this paper, we discussed the fast algebraic attacks introduced by Courtois at Crypto 2003. Using a very clever idea, "traditional" algebraic attacks can be improved significantly in many cases by performing an efficient precomputation step. For this purpose, an algorithm A was proposed. Unfortunately, neither a correctness proof was given, nor was the correctness obvious.

In this paper, we gave the missing proof based on a cryptographically reasonable assumptions. To do so, it was necessary to prove some non-trivial statements about minimal polynomials of linear recurring sequences. To the best of our knowledge, these have not been published elsewhere in open literature.

In addition, we showed that the knowledge of the minimal polynomials of the LFSRs used in the cipher can help to improve fast algebraic attacks. For this reason, we developed an algorithm B which is based on a deeper understanding of minimal polynomials of linear recurring sequences.

Finally, we demonstrated the higher efficiency of algorithm B by applying it to the E_0 key stream generator used in the Bluetooth standard for wireless communication. In this case, theoretical analysis yielded that algorithm B runs almost 8 times faster than algorithm A. This improves the fastest known attack against E_0. An ad-hoc implementation applied to reduced versions of E_0 confirmed the advantage of our new algorithm for the test cases considered, even without using parallelism.

Acknowledgment

The author would like to thank Erik Zenner, Stefan Lucks, Matthias Krause and some unknown referees for helpful comments and discussions.

References

1. Frederik Armknecht, Matthias Krause: *Algebraic attacks on Combiners with Memory*, Proceedings of Crypto 2003, LNCS 2729, pp. 162-176, Springer, 2003.
2. Bluetooth SIG, *Specification of the Bluetooth system*, Version 1.1, February 22, 2001. Available at http://www.bluetooth.com/.
3. Alin Bostan, Philippe Flajolet, Bruno Salvy, Eric Schost: *Fast Computation With Two Algebraic Numbers*, submitted, 2003.
4. Alex Biryukov, Adi Shamir: *Cryptanalytic Time/Memory/Data tradeoffs for Stream Ciphers*, Proceedings of Asiacrypt 2000, LNCS 1976, pp. 1-13, Springer, 2000.
5. Vladimor V. Chepyzhov, Ben Smeets: *On A Fast Correlation Attack on Certain Stream Ciphers*, Proceedings of Eurocrypt 1991, LNCS 547 pp. 176-185, Springer, 1991.
6. Nicolas Courtois: *Higher Order Correlation Attacks, XL Algorithm and Cryptanalysis of Toyocrypt*, ICISC 2002, LNCS 2587. An updated version (2002) is available at http://eprint.iacr.org/2002/087/.
7. Nicolas Courtois, Willi Meier: *Algebraic attacks on Stream Ciphers with Linear Feedback*, Eurocrypt 2003, Warsaw, Poland, LNCS 2656, pp. 345-359, Springer, 2003. An extended version is available at http://www.minrnak.org/toyolili.pdf
8. Nicolas Courtois: *Fast Algebraic Attacks on Stream Ciphers with Linear Feedback*, Proceedings of Crypto '03, LNCS 2729, pp. 177-194, Springer, 2003.
9. Scott R. Fluhrer, Stefan Lucks: *Analysis of the E_0 Encryption System*, Proceedings of Selected Areas of Cryptography '01, LNCS 2259, pp. 38-48, Springer, 2001.
10. Jovan Dj. Golic: *Cryptanalysis of Alleged A5 Stream Cipher*, Proceedings of Eurocrypt 1997, LNCS 1233, pp. 239-255, Springer, 1997.
11. Rainer Goettfert, Harald Niederreiter: *On the Linear Complexity of Products of Shift-Register Sequences*, Proceedings of Eurocrypt '93, pp. 151-158, LNCS 765, Springer, 1994.
12. Thomas Johansson, Fredrik Joensson: *Fast Correlation Attacks Based on Turbo Code Techniques*, Proceedings of Crypto 1999, LNCS 1666, pp. 181-197, Springer, 1999.
13. Thomas Johansson, Fredrik Joensson: *Improved Fast Correlation Attacks on Stream Ciphers via Convolutional Codes*, Proceedings of Eurocrypt 1999, pp. 347-362, Springer, 1999.
14. Edwin L. Key: *An Analysis of the Structure and Complexity of Nonlinear Binary Sequence Generators*, IEEE Transactions on Information Theory, Vol. IT-22, No. 6, November 1976.

15. Matthias Krause: *BDD-Based Cryptanalysis of Key stream Generators*; Proceedings of Eurocrypt '02, pp. 222-237, LNCS 2332, Springer, 2002.
16. Rudolf Lidl, Harald Niederreiter: *Introduction to finite fields an their applications*, Cambridge University Press, 1994.
17. J. L. Massey: *Shift-register synthesis and BCH decoding*, IEEE Trans. Information Theory, IT-15 (1969), pp. 122-127, 1969.
18. Willi Meier, Othmar Staffelbach: *Fast Correlation Attacks on certain Stream Ciphers*, Journal of Cryptology, pp. 159-176, 1989.
19. Alfred J. Menezes, Paul C. Oorschot, Scott A. Vanstone: *Handbook of Applied Cryptography*, Chapter 6, CRC Press.
20. Rainer A. Rueppel: *Stream Ciphers*; Contemporary Cryptology: The Science of Information Integrity. G. Simmons ed., IEEE Press New York, 1991.
21. A. Schoenhage: *Schnelle Multiplikation von Polynomen ueber Koerpern der Charakteristik 2*, Acta Informatica 7 (1977), pp. 395-398, 1977.
22. Volker Strassen: *Gaussian Elimination is Not Optimal*; Numerische Mathematik, vol 13, pp 354-356, 1969.
23. Erik Zenner: *On the Efficiency of the Clock Control Guessing Attack*, Proceedings of ICISC 2002, LNCS 2587, Springer, 2002.
24. Erik Zenner, Matthias Krause, Stefan Lucks: *Improved Cryptanalysis of the Self-Shrinking Generator ACISP 2001*, LNCS 2119, Springer, 2001.

A Motivation

In this section, we give a motivation for the somewhat technical definition 1. For this purpose we discuss some kind of "abstract example". Let $A = (a_t)$ resp. $B = (b_t)$ be two sequences produced by the co-prime minimal polynomials $m_a(x) = \prod(x - \alpha_i)$ and $m_b(x) = \prod(x - \beta_i)$. Then by theorem 3, a_t resp. b_t can be expressed by

$$a_t = \sum_i c_{\alpha_i} \alpha_i^t, \quad b_t = \sum_i c_{\beta_i} \beta_i^t$$

Consequently, the shifted sequences can be expressed by

$$\begin{aligned} a_{t+d_a} &= \sum_i c_{\alpha_i} \alpha_i^{t+d_a} = \sum_i c_{\alpha_i} \alpha_i^{d_a} \alpha_i^t =: \sum_i \tilde{c}_{\alpha_i} \alpha_i^t \\ b_{t+d_b} &= \sum_i c_{\beta_i} \beta_i^{t+d_b} = \sum_i c_{\beta_i} \beta_i^{d_b} \beta_i^t =: \sum_i \tilde{c}_{\beta_i} \beta_i^t \end{aligned}$$

Furtheron, we define the sequences

$$\begin{aligned} z_t &= a_t \cdot a_t + b_t && = \textstyle\sum_{i,j} c_{\alpha_i} c_{\alpha_j} \alpha_i^t \alpha_j^t + \sum_i c_{\beta_i} \beta_i^t = \sum_\pi c_\pi \pi^t \\ \tilde{z}_t &= a_{t+d_a} \cdot a_{t+d_a} + b_{t+d_b} && = \textstyle\sum_\pi \tilde{c}_\pi \pi^t \end{aligned}$$

where $\pi \in P := \{\alpha_i \cdot \alpha_j\} \cup \{\beta_i\}$. If we assume that all roots α_i and β_j are non-zero[6] this holds for the elements in P too. The goal is to find a necessary precondition which guarantees that $\min(z_t) = \min(\tilde{z}_t)$ (regardless of the values of d_a and d_b). By theorem 4 we know that such a precondition is

$$c_\pi \neq 0 \Leftrightarrow \tilde{c}_\pi \neq 0 \tag{6}$$

How can we be sure that this is true in our case? We show what can go wrong on an example. Let $\pi \in P$ fixed. We distinguish now between two different cases:

1. π has the following two different representations in P: $\pi = \alpha_1 \cdot \alpha_2 = \alpha_3$
2. π has the following two different representations in P: $\pi = \alpha_1 \cdot \alpha_2 = \alpha_3 \cdot \beta_1$ with $\beta_1 \neq 1$

In the first case, we can express c_π and $\tilde{c}_\pi$ by

$$\begin{aligned} c_\pi &= (c_{\alpha_1} \cdot c_{\alpha_2} + c_{\alpha_3}) \\ \tilde{c}_\pi &= (\tilde{c}_{\alpha_1} \cdot \tilde{c}_{\alpha_2} + \tilde{c}_{\alpha_3}) \\ &= (c_{\alpha_1} \cdot \alpha_1^{d_a} \cdot c_{\alpha_2} \cdot \alpha_2^{d_a} + c_{\alpha_3} \cdot \alpha_3^{d_a}) \\ &= (c_{\alpha_1} \cdot c_{\alpha_2} \cdot (\underbrace{\alpha_1 \cdot \alpha_2}_{=\pi})^{d_a} + c_{\alpha_3} \cdot (\underbrace{\alpha_3}_{=\pi})^{d_a}) \\ &= (c_{\alpha_1} \cdot c_{\alpha_2} + c_{\alpha_3}) \cdot \pi^{d_a} \\ &= c_\pi \pi^{d_a} \end{aligned}$$

[6] This is for example true if $m_a(x)$ and $m_b(x)$ are irreducible and have a degree > 1.

As we said before $\pi \in P$ is non-zero. Hence, it is $c_\pi \neq 0 \Leftrightarrow \tilde{c}_\pi \neq 0$ and condition 6 is fulfilled. Therefore, we can be sure that $min(\tilde{Z})$ is the same for all choices of d_a and d_b.

Now let us have a look at the second case. W.l.o.g., we assume $d_a < d_b$. Then

$$\begin{aligned} c_\pi &= (c_{\alpha_1} \cdot c_{\alpha_2} + c_{\alpha_3} \cdot c_{\beta_1}) \\ \tilde{c}_\pi &= (\tilde{c}_{\alpha_1} \cdot \tilde{c}_{\alpha_2} + \tilde{c}_{\alpha_3} \cdot \tilde{c}_{\beta_1}) \\ &= (c_{\alpha_1} \cdot c_{\alpha_2} \cdot \pi^{d_a} + c_{\alpha_3} \cdot c_{\beta_1} \cdot \pi^{d_a} \cdot \beta_1^{d_b - d_a}) \\ &= (c_{\alpha_1} \cdot c_{\alpha_2} + c_{\alpha_3} \cdot c_{\beta_1} \cdot \beta_1^{d_b - d_a}) \cdot \pi^{d_a} \end{aligned}$$

In this case (depending on c_{α_1}, c_{α_2}, c_{α_3}, c_{β_1}, β_1, d_a and d_b) it could happen that $c_\pi \neq 0$ but $\tilde{c}_\pi = 0$ (or vice versa).

The motivation for definition 1 was to avoid cases like case 2. To match the case considered here on definition 1, we set $R_1 := \{\alpha_i | i = \ldots\}$ (the roots of $m_a(x)$), $R_2 := \{\beta_j | j = \ldots\}$ and $R = R_1 \dot\cup R_2$. Let $V = (v_1, \ldots, v_n) \in R^n$ and $W := (w_1, \ldots, w_m) \in R^m$. Definition 1 was that V and W factorize uniquely over R_1, R_2 if

$$v_1 \cdot \ldots \cdot v_n = w_1 \cdot \ldots \cdot w_m \Rightarrow \prod_{v_i \in R_1} v_i \cdot \prod_{w_j \in R_1} w_j^{-1} = 1 \text{ and } \prod_{v_i \in R_2} v_i \cdot \prod_{w_j \in R_2} w_j^{-1} = 1$$

Now we check if the definitions are fulfilled for case 1 and 2 for the representations of π:

Case 1: It is $V = (v_1, v_2) = (\alpha_1, \alpha_2) \in R_1 \times R_1$ and $W = (w_1) = (\alpha_3) \in R_1$. As all of them are elements of R_1 we only have to check

$$v_1 \cdot v_2 \cdot w_1^{-1} = \pi \cdot \pi^{-1} = 1$$

Case 2: It is $V = (v_1, v_2) = (\alpha_1, \alpha_2) \in R_1 \times R_1$ and $W = (w_1, w_2) = (\alpha_3, \beta_1) \in R_1 \times R_2$. We have to check if $v_1 \cdot v_2 \cdot w_1^{-1} = 1$ and $w_2^{-1} = 1$. But this is not true as by assumption $w_2 = \beta_1 \neq 1$.

In the proof of theorem 2 it is shown that cases like the second one can be avoided if we require that the vectors $\overrightarrow{\pi}$ with $\pi \in P$ do all factorize uniquely over R_1, R_2.

B On the practical Relevance of Theorem 2

In this section we show that theorem 2 applies to a large class of LFSR-based ciphers automatically. Let $m_1(x), \ldots, m_\kappa(x) \in \mathrm{F}_2[x]$ be the primitive minimal polynomials of the used LFSRs such that the roots are all pairwise distinct and non-zero. For the following classes of ciphers, the assumptions of theorem 2 are always satisfied:

1. The cipher is a filter generator, i.e. $\kappa = 1$.
2. The degrees of the minimal polynomials are pairwise co-prime.

Set $R := R_1 \dot\cup \ldots \dot\cup R_\kappa$. We define by $\mathcal{P} := \{\alpha_1 \cdot \ldots \cdot \alpha_n | \alpha_i \in R, n \in \mathrm{N}\}$ the set of all possible multiple products of elements in R. We show now that in both cases, all pairs of vectors $\overrightarrow{\alpha}, \overrightarrow{\beta}$ with $\alpha, \beta \in \mathcal{P}$ factorize uniquely over $R_1, \ldots, R_\kappa$. As $\mathcal{P}$ is a superset for all possible sets $\{\mu(\alpha)| \ldots\}$ this proves that the conditions of theorem 2 are satisfied. Let from now on $\overrightarrow{\alpha} = (\alpha_1, \ldots, \alpha_n)$ and $\overrightarrow{\beta} = (\beta_1, \ldots, \beta_m)$ denote two vectors such that $\alpha_1 \cdot \ldots \cdot \alpha_n$ and $\beta_1 \cdot \ldots \cdot \beta_m$ are elements in $\mathcal{P}$.

Let us start with the first case. Here, we have

$$\alpha_1 \cdot \ldots \cdot \alpha_n = \beta_1 \cdot \ldots \cdot \beta_m \Leftrightarrow \prod \alpha_i \cdot \prod \beta_j^{-1} = 1 \Leftrightarrow \prod_{\alpha_i \in R_1} \alpha_i \cdot \prod_{\beta_j \in R_1} \beta_j^{-1} = 1$$

This concludes the first case.

For the second case we remember the fact that $\mathrm{F}_{2^n} \subseteq \mathrm{F}_{2^m}$ iff n divides m. In particular, $\mathrm{F}_{2^n} \cap \mathrm{F}_{2^m} = \mathrm{F}_{2^c}$ with $c := gcd(n, m)$. We denote by T_i the degree of the minimal polynomial $m_i(x)$. Then the elements $\alpha, \alpha^{-1}, \alpha \in R_i$, and all multiple products are elements of $\mathrm{F}_{2^{T_i}}$. Let l be arbitrary with $1 \le l \le \kappa$ and set $S_l := T_1 \cdot \ldots \cdot T_{l-1} \cdot T_{l+1} \cdot \ldots \cdot T_\kappa$. Then $\alpha_1 \cdot \ldots \cdot \alpha_n = \beta_1 \cdot \ldots \cdot \beta_m$ implies

$$\gamma_l := \underbrace{\prod_{\alpha_i \in R_l} \alpha_i \prod_{\beta_j \in R_l} \beta_j^{-1}}_{\in \mathrm{F}_{2^{T_l}}} = \underbrace{\prod_{\alpha_i \notin R_l} \alpha_i \prod_{\beta_j \notin R_l} \beta_j^{-1}}_{\in \mathrm{F}_{2^{S_l}}}$$

Therefore, $\gamma_l \in \mathrm{F}_{2^{T_l}} \cap \mathrm{F}_{2^{S_l}}$. By assumption the values T_i are pairwise co-prime. Hence, it is $gcd(T_l, S_l) = 1$ and $\gamma_l \in \mathrm{F}_{2^1} = \mathrm{F}_2$. As the roots are all non-zero, γ_l equals to 1 for each choice of l. This concludes the second case.

C The E_0 key stream generator

In this section, we show that theorem 2 and algorithm B are applicable to the E_0 cipher together with the following boolean function:

$$F = \sum_{\substack{1 \le i < j \le 4 \\ 1 \le k < l \le 4}} z_t^{(i)} z_t^{(j)} z_{t+1}^{(k)} z_{t+1}^{(l)} + z_t^{(1)} z_t^{(2)} z_t^{(3)} z_t^{(4)} \tag{7}$$

Let from now on denote i, j, k, l integers from the set $\{1, 2, 3, 4\}$. We define the following three sets of indices

$$I_2 := \{(i, j) \mid i < j\}$$
$$I_3 := \{(i, j, k) \mid i < j < k\}$$
$$I_4 := \{(i, j; k, l) \mid i < j, k < l, \{i, j\} \cup \{k, l\} = \{1, 2, 3, 4\}\}$$

Then, F can be rewritten as

$$F = \sum_{(i,j) \in I_2} \underbrace{z_t^{(i)} z_{t+1}^{(i)} z_t^{(j)} z_{t+1}^{(j)}}_{=: F_{(i,j)}} \tag{8}$$

$$+ \sum_{(i,j,k)\in I_3} \underbrace{\left(f_{ij} \cdot z_t^{(k)} z_{t+1}^{(k)} + f_{ik} \cdot z_t^{(j)} z_{t+1}^{(j)} + f_{jk} \cdot z_t^{(i)} z_{t+1}^{(i)}\right)}_{=:F_{(i,j,k)}} \tag{9}$$

$$\underbrace{+ \sum_{(i,j;k,l)\in I_4} z_t^{(i)} z_{t+1}^{(j)} z_t^{(k)} z_{t+1}^{(l)} + z_t^{(1)} z_t^{(2)} z_t^{(3)} z_t^{(4)}}_{\tilde{F}} \tag{10}$$

where $f_{ij} = z_t^{(i)} z_{t+1}^{(j)} + z_t^{(j)} z_{t+1}^{(i)}$. Let R_i be the set of roots of m_i, the minimal polynomial of the ith LFSR, and $R_i^2 := \{\alpha\alpha' | \alpha, \alpha' \in R_i, \alpha \neq \alpha'\}$. We define the sets :

$$\begin{aligned} R_{(i,j)} &:= \{\alpha_i\alpha_j \mid (i,j) \in I_2, \alpha_s \in R_s \cup R_s^2\} \\ R_{(i,j,k)} &:= \{\alpha_i\alpha_j\alpha_k \mid (i,j,k) \in I_3, \alpha_s \in R_s \cup R_s^2\} \\ \tilde{R} &:= \{\alpha_i\alpha_j\alpha_k\alpha_l \mid (i,j;k,l) \in I_4, \alpha_s \in R_s\} \end{aligned}$$

The first thing we observe is that the sets defined above are all pairwise disjunct. Furthermore, it is easy to see that the roots of $min(F_{(i,j)})$ are a subset of $R_{(i,j)}$ and so on. Thus, the minimal polynomials are pairwise co-prime and we can write by theorem 5 $min(F)$ as the product of 11 different minimal polynomials:

$$min(F) = \prod_{(i,j)\in I_2} min(F_{(i,j)}) \cdot \prod_{(i,j,k)\in I_3} min(F_{(i,j,k)}) \cdot min(\tilde{F})$$

Actually, even more can be said. Using theorem 5 and the fact that the polynomials m_i are pairwise co-prime, we have $min(F_{(i,j)}) = (m_i \otimes m_i) \otimes (m_j \otimes m_j)$ of degree $T_i(T_i+1)/2 \cdot T_j(T_j+1)/2$. Before we can estimate $min(F_{(i,j,k)})$ and $min(\tilde{F})$, we need the following theorem:

Theorem 9. *([16], Th. 6.55) Given sequences $\mathcal{Z}_1, \ldots, \mathcal{Z}_\kappa$, the minimal polynomial $min(\mathcal{Z}_1 + \ldots + \mathcal{Z}_\kappa)$ divides $lcm(min(\mathcal{Z}_1), \ldots, min(\mathcal{Z}_\kappa))$.*

First, we consider $min(F_{(i,j,k)})$. By theorem 5, it can be easy seen that

$$min(z_t^{(i)} z_{t+1}^{(j)}) = min(z_t^{(j)} z_{t+1}^{(i)}) = m_i \otimes m_j,$$

and by theorem 6, that $min(z_t^{(k)} z_{t+1}^{(k)}) = m_k \otimes m_k$. By theorem 9, the minimal polynomial $min(F_{(i,j,k)})$ divides the least common multiple of the polynomials $(m_i \otimes m_i) \otimes (m_j \otimes m_k)$, $(m_j \otimes m_j) \otimes (m_i \otimes m_k)$ and $(m_k \otimes m_k) \otimes (m_i \otimes m_j)$. Notice that all three polynomials share the common factor $m_i \otimes m_j \otimes m_k$.[7] Thus, the degree of $min(F_{(i,j,k)})$ is upper bounded by

$$\begin{aligned} & T_iT_jT_k + \frac{T_i(T_i-1)}{2}T_jT_k + \frac{T_j(T_j-1)}{2}T_iT_k + \frac{T_k(T_k-1)}{2}T_iT_j \\ &= T_iT_jT_K\frac{T_i+T_j+T_k-1}{2} \end{aligned}$$

[7] The reason is that f divides $f \otimes f$ and that $(f \cdot g) \otimes h = (f \otimes h) \cdot (g \otimes h)$.

The minimal polynomial $min(\tilde{F})$ can be found exactly. We observe that

$$min(z_t^{(i)} z_{t+1}^{(j)} z_t^{(k)} z_{t+1}^{(l)}) = min(z_t^{(1)} z_t^{(2)} z_t^{(3)} z_t^{(4)}) = m_1 \otimes m_2 \otimes m_3 \otimes m_4 =: m$$

Thus, $min(\tilde{F})$ divides m. As $m_1, \ldots, m_4$ are irreducible, this holds for m too. Therefore, $min(\tilde{F})$ is equal to 1 or equal to m. But the first case would imply that the expression in (10) is the all-zero sequence which is obviously wrong. Ergo, $min(\tilde{F})$ is equal to m of degree $T_1T_2T_3T_4$. Summing up, for the degree T of $min(F)$ it holds

$$T \leq \sum_{(i,j)\in I_2} \frac{T_i(T_i+1)T_j(T_j+1)}{4} + \sum_{(i,j,k)\in I_3} T_iT_jT_k \frac{T_i+T_j+T_k-1}{2} + T_1T_2T_3T_4$$

D Why preconditions are necessary

In this section, we show that algorithm A (and B) do not work properly without some preconditions. To do so we give an example. We consider the case of two LFSRs $\mathcal{L}_a$ and $\mathcal{L}_b$ with the primitive minimal polynomials $m_a(x) = 1 + x + x^2$ and $m_b(x) = 1 + x + x^4$. Notice that the polynomials are co-prime but the corresponding periods are not. Let a_t resp. b_t denote the outputs of LFSR $\mathcal{A}$ and $\mathcal{B}$ and define the sequence $\mathcal{Z} := (z_t)$ by

$$z_t := a_tb_t + a_t + b_t + a_ta_{t+1} + b_tb_{t+1}.$$

Let $\mathcal{K}_a = (a_1, a_2)$ be the initial state of LFSR $\mathcal{L}_a$ and $\mathcal{K}_b = (b_1, b_2, b_3, b_4)$ be the initial state of LFSR $\mathcal{L}_b$. For the correctness of the pre-computation step, $min(\mathcal{Z})$ should be the same for all non-zero choices of $\mathcal{K}_a$ and $\mathcal{K}_b$. This is not the case. For $\mathcal{K}_a = (1, 0)$ and $\mathcal{K}_b = (1, 1, 1, 0)$ it is $min(\mathcal{Z}) = 1+x^2+x^3+x^6+x^7+x^9$. But for $\mathcal{K}_a = (0, 1)$ and $\mathcal{K}_b = (1, 1, 1, 1)$ it is $min(\mathcal{Z}) = 1 + x^{15}$.