Multivariate Correlation Attacks and the Cryptanalysis of LFSR-based Stream Ciphers

Isaac A. Canales-Martínez [1] and Igor Semaev [2]

[1]Cryptography Research Centre, Technology Innovation Institute, Abu Dhabi, UAE
isaac.canales@tii.ae
[2]Department of Informatics, University of Bergen, Bergen, Norway
igor.semaev@uib.no

Abstract

Cryptanalysis of modern symmetric ciphers may be done by using linear equation systems with multiple right hand sides, which describe the encryption process. The tool was introduced by Raddum and Semaev in [35] where several solving methods were developed. In this work, the probabilities are ascribed to the right hand sides and a statistical attack is then applied. The new approach is a multivariate generalisation of the correlation attack by Siegenthaler [37]. A fast version of the attack is provided too. It may be viewed as an extension of the fast correlation attack in [30] by Meier and Staffelbach, based on exploiting so called parity-checks for linear recurrences. Parity-checks are a particular case of the relations that we introduce in the present work. The notion of a relation is irrelevant to linear recurrences. We show how to apply the method to some LFSR-based stream ciphers including those from the Grain family. The new method generally requires a lower number of the keystream bits to recover the initial states than other techniques reported in the literature.

Keywords — Cryptanalysis, Multivariate correlation attacks, Test-and-extend algorithm, Stream ciphers, LFSRs, Grain

1 Introduction

The goal of a key recovery attack against a stream cipher is to get the secret key given a sequence of the generated keystream bits. The key is used for initialising various components of the cipher. On devices employing linear feedback shift registers (LFSR), the key is used to set their initial states. We focus on attacks against LFSR-based stream ciphers whose goal is to recover the cipher's initial state that produced the given keystream.

A non-linear filter generator is a keystream generator used for constructing stream ciphers. It consists of a binary LFSR of length n and a Boolean function f in ℓ variables, as depicted in Figure 1. The LFSR's feedback taps are defined by its degree-n primitive polynomial[1] $g = x^n - c_{n-1}x^{n-1} - \ldots - c_1 x - 1 \in \mathbb{F}_2[x]$. The LFSR sequence $s_1, s_2, \ldots$ satisfies the linear recurrence relation

$$s_{i+n} = c_{n-1}s_{i+n-1} + \ldots + c_1 s_{i+1} + s_i, \tag{1}$$

where the arithmetic is in $\mathbb{F}_2$. Let S_i be the LFSR state at time i, then $S_i = M^{i-1}S_1$, where M is the transpose of the companion matrix of g, i.e.,

$$S_i = \begin{pmatrix} s_i \\ s_{i+1} \\ \vdots \\ s_{i+n-1} \end{pmatrix} \quad \text{and} \quad M = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 1 & c_1 & \cdots & c_{n-1} \end{pmatrix}.$$

The keystream bit at time i is $z_i = f(s_{i+k_1}, \ldots, s_{i+k_\ell})$, where $f : \mathbb{F}_2^\ell \to \mathbb{F}_2$ and $0 \le k_1 < \cdots < k_\ell \le n-1$. The polynomial g, the filtering function f and the indices $k_1, \ldots, k_\ell$ are considered to be public.

[1]It is not necessary for the polynomial to be primitive, however, when it is, the LFSR sequence has maximum period $2^n - 1$ on a non-zero initial state.

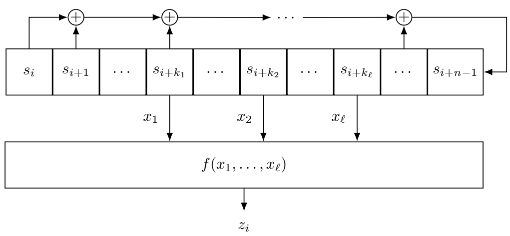

Figure 1: Model of a filter generator.

Let Λ be an $\ell \times n$ matrix which "selects" the inputs to f from S_i, i.e.,

$$\begin{pmatrix} s_{i+k_1} \\ \vdots \\ s_{i+k_\ell} \end{pmatrix} = \Lambda S_i \quad \text{and} \quad \Lambda = \begin{pmatrix} e_{k_1+1} \\ \vdots \\ e_{k_\ell+1} \end{pmatrix},$$

where $e_i = (0,\dots,0,1,0,\dots,0)$, $i = 1,\dots,n$, and the only 1 is in position i from the left. Define $A_i = \Lambda M^{i-1}$. Then $z_i = f(\Lambda S_i) = f(\Lambda M^{i-1} S_1) = f(A_i X)$, where $X = S_1$. The pre-image of z_i is the set of all possible values of $A_i X$ in $\mathbb{F}_2^\ell$ such that $z_i = f(A_i X)$. We assign a uniform probability distribution on the pre-image of z_i and all other values in $\mathbb{F}_2^\ell$ get probability 0. That defines a probability distribution for a random variable X_i on the values of $A_i X$. We assume that X is uniformly distributed on $\mathbb{F}_2^n$ and that X_i are independent. Let N keystream bits be available. The attack on the filter generator is then to find the value $X = x$ with maximum likelihood under the condition that

$$A_i X = X_i,\ i = 1,\dots,N. \tag{2}$$

We present a statistical approach to solve (2) and employ it to find the LFSR's initial state. This new method is general and does not use the properties of the recurrence (1), including *parity-checks* as in Fast Correlation Attacks (see Section 1.1 below).

1.1 Published Attacks

There is a substantial literature on key recovery attacks against LFSR-based stream ciphers, where Fast Correlation Attacks (FCA) are among the most important. The first FCA was discovered by Meier and Staffelbach [30] as an improvement to the correlation attack by Siegenthaler [37]. Algebraic attacks [11, 10] form another important class. The attacks by Anderson [3], Golić et. al [17] and Leveiller et. al [27] are examples of the so-called deterministic attacks, which are efficient for very specific devices. There is also a general class of time/memory/data trade-offs [5]. FCAs are the most relevant technique in our context and we focus exclusively on them.

Assume we have N keystream bits generated by an LFSR-based stream cipher and let g be the LFSR's feedback polynomial. A *parity-check* is an equation

$$1 + x^{i_1} + \cdots + x^{i_{d-1}} \equiv 0 \bmod g, \tag{3}$$

for some indices $0 < i_1 < \cdots < i_{d-1} < N$, and d is called its *weight*. The LFSR sequence satisfies these equations: $s_j + s_{i_1+j} + \cdots + s_{i_{d-1}+j} = 0$ for $1 \le j \le N - i_{d-1}$. Parity-checks only depend on g and N. A more general definition of a parity-check is given in [9].

In FCAs, recovering the LFSR's initial state is represented as a decoding problem. The sequence $\{s_i\}_{i=1}^N$ generated by the LFSR is the information transmitted and the keystream $\{z_i\}_{i=1}^N$, is the information received at the other end of a binary symmetric channel with crossover probability $1-p$, where s_i and z_i are correlated as $p = \Pr(s_i = z_i) \neq 1/2$. FCAs are usually comprised of a precomputation phase and a decoding phase. The objective in precomputation is to obtain many low-weight parity-checks. The initial state of the LFSR is recovered during the decoding phase. In brief, the decoding algorithm removes the noise using the information on how the keystream satisfies the parity-checks. Algorithms for the decoding phase are either one-pass or iterative.

The efficiency of FCAs is affected by some characteristics of the device, notably the weight (i.e., the number of non-zero terms) of g. Also, for the filter generator, the nonlinearity of f (see [31]). Golić presents in [16] a thorough list of design criteria to make the filter generator resistant to various attacks. Table 1 summarises some results of published FCAs for a variety of parameters, where d is the weight of

Attack	deg(g)	weight(g)	d	$1-p$	N
Johansson, Jönsson [24]	40	17	2	0.260	$4 \cdot 10^4$
	40	17	2	0.400	$4 \cdot 10^5$
Johansson, Jönsson [23]	40	17	2	0.300	$4 \cdot 10^4$
	40	17	2	0.410	$4 \cdot 10^5$
Chepyzhov et al. [8]	60	?	3	0.300	$6.3 \cdot 10^4$
	60	?	3	0.400	$6 \cdot 10^5$
	70	?	3	0.350	$1.12 \cdot 10^6$
Canteaut,Trabbia [6]	40	17	4	0.440	$4 \cdot 10^5$
	40	17	5	0.482	$3.6 \cdot 10^5$
Johansson, Jönsson [25]	40	17	2	0.450	$4 \cdot 10^5$
	60	13	3	0.320	$1.5 \cdot 10^5$
	60	13	2	0.430	$4 \cdot 10^7$
Mihaljević et al. [32]	40	17	3	0.469	$4 \cdot 10^5$
	40	17	3	0.490	$3.6 \cdot 10^5$
	*89	?	3	0.469	$\approx 2.5 \cdot 10^{11}$
	*89	?	3	0.478	$\approx 10^{12}$
	*89	?	3	0.480	$\approx 4 \cdot 10^{12}$
Chose et al. [9]	40	17	4	0.469	$8 \cdot 10^4$
	*40	17	4	0.490	$8 \cdot 10^4$
	*89	?	4	0.469	2^{28}
Molland et al. [33]	60	?	4	0.430	$1.5 \cdot 10^7$
	60	?	4	0.470	$1 \cdot 10^8$
Leveiller et al. [28]	40	17	5	0.375	$1.7 \cdot 10^4$
	100	3	3	0.4375	$3 \cdot 10^4$
Didier [12]	53	?	5	0.4375	$\approx 4 \cdot 10^5$
	59	?	5	0.4531	$\approx 1.45 \cdot 10^6$
	61	?	5	0.4531	$\approx 2.1 \cdot 10^6$

Table 1: Some published results of FCAs. *: theoretical results only. ?: neither g is explicitly presented nor its weight is reported.

the parity-checks, $p = \Pr(s_i = z_i)$ and the length of the available keystream is N. For details on FCAs, we refer to the original FCA [30], the sources in Table 1 and the surveys [29, 2].

More recently, some attacks to the Grain family [21] of ciphers have been published. Todo et al. [38] presented a new FCA against the ciphers Grain-v1 [22], Grain-128 [20] and Grain-128a [1]. The attack exploits a commutative property obtained by considering parity-checks as elements of a finite field. The authors also propose a new one-pass algorithm to recover the initial state and show new linear approximations of the ciphers with high correlation. The number of required keystream bits and number of operations is, respectively, $2^{75.1}$ and $2^{76.7}$ for Grain-v1, $2^{112.8}$ and $2^{114.4}$ for Grain-128, and $2^{113.8}$ and $2^{115.4}$ for Grain-128a. Zhou et al. [39] later presented an attack using vectorial parity-checks and a new vectorial iterative decoding algorithm which generalises the original iterative method in [30]. For Grain-128a, this attack is reported to require around 2^{91} keystream bits and 2^{92} operations. Vectorial versions of the FCA were introduced in [18, 19].

1.2 Overview of the New Method and Applications

We first formulate (2) in more generality and describe a multivariate generalisation of the correlation attack by Siegenthaler [37] in Sections 2.1 and 2.2, respectively. The correct solution is recovered by deciding whether a candidate solution follows a uniform distribution or the distribution induced by (2). For this, we use a maximum likelihood type statistic. In Section 2.4, similarly to [9], we employ the Fast Fourier Transform (FFT) to compute the values of the statistic. This largely reduces the time complexity of the method while increasing its space complexity. However, a straightforward application of this basic multivariate correlation attack still has time complexity at least 2^n, though requiring a lower number of equations compared to the original univariate attack. The goal is to solve (2) in a more efficient way.

The new method is constructed upon relations with a more general nature than parity-checks used in FCAs. Hence, such relations may be regarded as a generalisation of parity-checks and vectorial parity-checks used in [19, 39]. Let $B = B_r$ denote a matrix of size $r \times n$ and of rank $1 \le r \le n$. A subset $I \subseteq \{1, 2, \ldots, N\}$ is called a *relation modulo* B if the space spanned by the rows of A_i, $i \in I$, and the space spanned by the rows of B have an intersection of dimension $r_I \ge 1$. Let T_I be a matrix of size $r_I \times r$ such that the rows of the matrix $T_I B$ form a basis of this intersection. Also, let $Y = BX$. One may compute a probability distribution for a random variable Y_I on the values of $T_I Y = T_I BX$. The distribution of Y_I only depends on the distribution of X_i, $i \in I$. A set $\mathcal{I}_r$ of relations I, where the distribution of Y_I is non-uniform, is collected. Then, the system of equations

$$T_I Y = Y_I, \quad I \in \mathcal{I}_r.$$

is similar to (2), but of a smaller dimension. The values of $Y = BX$ are tested with the multivariate correlation method. Even though the distributions of Y_I are closer to uniform compared to the distributions of X_i, and the random variables Y_I may be dependent, the number of values to test is reduced. The trade-off may be positive and that is proved by our experiments with the filter generator, where the time complexity is lower than 2^n.

To compute the solution $X = x$ to (2) from the system above, a test-and-extend algorithm is used. The algorithm is similar to a tree search method in a variation of linear cryptanalysis for block ciphers (see [15]). This new method has two variations and is presented in Section 3 in detail. Since recovering the initial state is not modelled as a decoding problem, the algorithm is not a decoding procedure as in FCAs. Two techniques for obtaining a set $\mathcal{I}_r$ of relations are in Section 4. Several methods to compute the probability distribution of Y_I, $I \in \mathcal{I}_r$, are introduced in Section 5. The analysis of the new technique and implementation details are given in Sections 6 and 7, respectively.

The new method is applied to "hard" instances of the filter generator; the results are reported in Section 8 and summarised in Table 2. Experimentally, our method requires less keystream bits to recover the LFSR's initial state compared to existing FCAs for similar parametrs, and it is still significantly faster than brute force (see Section 8.2.2). When the LFSR is an internal component of a stream cipher, we may have (2) with the distributions of X_i defined differently. For instance, we showcase an application to Grain-v1 in Section 9. Compared to the $2^{75.1}$ keystream bits required in [38], our new technique requires $2^{53.5}$ bits with a trade-off on time complexity ($2^{90.5}$ vs $2^{76.7}$ in [38]). Additionally, we found linear approximations to Grain-v1 with higher correlation than those reported in [38].

$\deg(g)$	weight(g)	d	$1-p$	N	time complex.
40	17	3	0.375	$5 \cdot 10^3$	2^{32}
40	17	3	0.375	$5 \cdot 10^3$	2^{35}
64	17	3	0.375	$1 \cdot 10^4$	2^{57}
80	7	3	0.375	$1 \cdot 10^4$	2^{71}

Table 2: Result of experimental attacks against the filter generator.

The idea of the method and theoretical results in Sections 2-6 are due to Semaev. All computer calculations in this work and application of the method to the filter generator and the Grain family of stream ciphers in Sections 7-9 are due to Canales-Martínez.

2 Multivariate Correlation Attacks

In this section, the general problem we plan to solve is formulated. Also, multivariate extensions of known correlation attacks are introduced. That is a basis for the new method in Section 3.

2.1 General Problem

Let A_i, $i = 1, \ldots, N$, be $\ell_i \times n$ matrices of rank ℓ_i over a finite field $\mathbb{F}_q$, where ℓ_i are small compared to n. Let X be a vectorial random variable uniformly distributed on $\mathbb{F}_q^n$ and X_i, $i = 1, \ldots, N$, be vectorial random variables on $\mathbb{F}_q^{\ell_i}$, where $\Pr(X_i = a) = P_i(a)$ for some probability distribution P_i on $\mathbb{F}_q^{\ell_i}$. We consider the system of equations

$$A_i X = X_i, \quad i = 1, \ldots, N. \tag{4}$$

The task is to find the value x that maximises

$$\Pr\left(X = x \mid A_i X = X_i,\, i = 1, \ldots, N\right).$$

It is equivalent to maximising the likelihood $\Pr(X_1 = A_1x, \dots, X_N = A_Nx)$. If X_i are independent, we may maximise $\sum_{i=1}^{N} \ln \Pr(X_i = A_ix) = \sum_{i=1}^{N} \ln P_i(A_ix)$, for $P_i(A_ix) \neq 0$. We will assume the variables $X, X_1, \dots, X_N$ to be independent.

With the description as in Section 1, the attack against the filter generator is a particular case of this problem. Multiple right hand side equation systems introduced in [35] are also a particular case of the problem.

2.2 Basic Multivariate Correlation Attack

For $x \in \mathbb{F}_q^n$, we decide whether $x_i = A_ix$, $i = 1, \dots, N$, were independently taken from the distributions P_i on $\mathbb{F}_q^{\ell_i}$ (Hypothesis 1) or independently taken from uniform distributions on $\mathbb{F}_q^{\ell_i}$ (Hypothesis 0). Given a threshold $c \in \mathbb{R}$, we say x *survives* if

$$P_i(x_i) \neq 0, \quad i = 1, \dots, N, \tag{5}$$

$$S(x) = \sum_{i=1}^{N} \ln P_i(x_i) \geq c. \tag{6}$$

Let β be a prescribed success probability. We will show how to compute c such that $\Pr(S(x) \geq c) = \beta$ under Hypothesis 1. The number of incorrect survivors is on average αq^n, where α is the probability of an incorrect x to pass the test, that is under Hypothesis 0. We define asymptotic distributions of the statistic $S(x)$ in these two cases:

- Hypothesis 1. Let

$$\mu_{1i} = \sum_{y \in \mathbb{F}_q^{\ell_i}} P_i(y) \ln P_i(y) \quad \text{and} \quad \sigma_{1i}^2 = \sum_{y \in \mathbb{F}_q^{\ell_i}} P_i(y) \ln^2 P_i(y) - \mu_{1i}^2$$

  be the expectation and the variance of $\ln P_i(x_i)$, respectively. Then $\mu_1 = \sum_{i=1}^{N} \mu_{1i}$ and $\sigma_1^2 = \sum_{i=1}^{N} \sigma_{1i}^2$ are the expectation and the variance of $S(x)$, respectively. Let P_i be close to the uniform distributions on their supports. Then, the Lyapunov condition [4] is satisfied for $S(x)$. Thus, for large enough N, its distribution approximately follows the normal distribution $\mathbf{N}(\mu_1, \sigma_1^2)$ by the Lyapunov Central Limit Theorem. The threshold c is then computed from $\beta = \Pr(\mathbf{N}(\mu_1, \sigma_1^2) \geq c)$.

- Hypothesis 0. Let K_i denote the size of the support of P_i, and

$$\mu_{0i} = \sum_{y \in \mathbb{F}_q^{\ell_i} : P_i(y) \neq 0} \frac{\ln P_i(y)}{K_i} \quad \text{and} \quad \sigma_{0i}^2 = \sum_{y \in \mathbb{F}_q^{\ell_i} : P_i(y) \neq 0} \frac{\ln^2 P_i(y)}{K_i} - \mu_{0i}^2$$

  be the expectation and the variance of $\ln P_i(x_i)$, respectively. Then $\mu_0 = \sum_{i=1}^{N} \mu_{0i}$ and $\sigma_0^2 = \sum_{i=1}^{N} \sigma_{0i}^2$ are the expectation and the variance of $S(x)$, respectively. Under the condition that $P_i(x_i) \neq 0$, $i = 1, \dots, N$, the distribution of $S(x)$ approximately follows $\mathbf{N}(\mu_0, \sigma_0^2)$ by the Lyapunov Central Limit Theorem. Thus $\alpha = (\prod_{i=1}^{N} \frac{K_i}{q^{\ell_i}}) \Pr(\mathbf{N}(\mu_0, \sigma_0^2) \geq c)$.

We may get multiple candidate solutions. In practice, however, the solution is unique for large enough N. The complexity of this straightforward attack is $O(Nq^n)$ operations. If the distributions P_i are uniform on their supports (as in equations (2) for the filter generator) the statistic $S(x)$ is a constant. Then, only (5) works to test the candidate solutions and the method is reduced to brute force on the LFSR initial state. Another example is the correlation attack in [37]. This attack is a particular case for $q = 2$, $\ell_i = 1$ and there are only two different distributions among P_i. In that case, only (6) works to test the candidate solutions.

2.3 Number of Equations

Let the distributions $P_1, \dots, P_N$ be permutations of the same distribution. Given a desired success probability β and the number of survivors αq^n, we estimate the number of necessary equations N (e.g., the amount of required keysteam bits in the cryptanalysis of a filter generator) and define the threshold c. Since

$$\mu_0 = N\mu_{01},\, \sigma_0^2 = N\sigma_{01}^2 \quad \text{and} \quad \mu_1 = N\mu_{11},\, \sigma_1^2 = N\sigma_{11}^2,$$

we find c and N from the equations

$$\alpha \prod_{i=1}^{N} \frac{q^{\ell_i}}{K_i} = \Pr(\mathbf{N}(N\mu_{01}, N\sigma_{01}^2) \geq c) \quad \text{and} \quad \beta = \Pr(\mathbf{N}(N\mu_{11}, N\sigma_{11}^2) \geq c).$$

2.4 Improved Time Complexity with FFT

Let every P_i be close to the uniform distribution on $\mathbb{F}_q^{\ell_i}$ such that $q^{\ell_i}P_i(y) = 1 + o(1)$. We show that with the Fast Fourier Transform (FFT) the time and space complexity of the multivariate correlation attack is $O(\sum_{i=1}^{N} \ell_i q^{\ell_i} + nq^n)$ and q^n, respectively. This is a multivariate extension of the univariate FFT-based method in [9].

Let ξ be a primitive q-th root of unity and let $u \cdot v$ denote the dot product of vectors u and v. We have that

$$P_i(y) = \sum_{a\in\mathbb{F}_q^{\ell_i}} W_{ia}\xi^{a\cdot y} = q^{-\ell_i} + \sum_{a\in\mathbb{F}_q^{\ell_i}:a\neq 0} W_{ia}\xi^{a\cdot y},$$

where $W_{ia} = q^{-\ell_i}\sum_{x\in\mathbb{F}_q^{\ell_i}} P_i(x)\xi^{-a\cdot x}$. The numbers W_{ia} are called the *Fourier spectrum of* P_i. Given an input vector of length q^n, the FFT computes the Fourier spectrum with time complexity $O(nq^n)$ for bounded q (e.g., see [7]). The cost of computing the spectrum for all $P_1, \ldots, P_N$ is then $O(\sum_{i=1}^{N} \ell_i q^{\ell_i})$ operations. By assumption, $P_i(y)$ are close to $q^{-\ell_i}$, so $q^{\ell_i}\sum_{a\in\mathbb{F}_q^{\ell_i}:a\neq 0} W_{ia}\xi^{a\cdot y}$ are small. Since $\ln(1+\varepsilon) \approx \varepsilon$ for small ε, we have

$$\ln P_i(y) = \ln(q^{-\ell_i} + \sum_{a\neq 0} W_{ia}\xi^{a\cdot y}) = \ln(1 + q^{\ell_i}\sum_{a\neq 0} W_{ia}\xi^{a\cdot y}) - \ell_i \ln q \approx q^{\ell_i}\sum_{a\neq 0} W_{ia}\xi^{a\cdot y} - \ell_i \ln q,$$

where $\sum_{a\neq 0}$ is a short notation for $\sum_{a\in\mathbb{F}_q^{\ell_i}:a\neq 0}$. Thus, the statistic $S(x)$ defined by (6) may be approximated as

$$\sum_{i=1}^{N} \ln P_i(A_i x) \approx \sum_{i=1}^{N} q^{\ell_i}\sum_{a\neq 0} W_{ia}\xi^{a\cdot(A_i x)} - \sum_{i=1}^{N} \ell_i \ln q = \sum_{b\in\mathbb{F}_q^n} C(b)\xi^{b\cdot x} - \sum_{i=1}^{N} \ell_i \ln q,$$

where $C(b) = \sum_{i=1}^{N} q^{\ell_i}\sum_{a\neq 0, aA_i=b} W_{ia}$, $b \in \mathbb{F}_q^n$. The non-zero $C(b)$ may be computed in $\sum_{i=1}^{N} q^{\ell_i}$ operations. The values $\sum_b C(b)\xi^{b\cdot x}$ for all $x \in \mathbb{F}_q^n$ may be computed with the FFT in $O(nq^n)$ operations. We have to keep the values $C(b)$ in order to apply the FFT, so the space complexity is q^n. Therefore, the overall time complexity of the attack is $O(\sum_{i=1}^{N} \ell_i q^{\ell_i} + nq^n)$, which is still larger than q^n. We apply this method to Grain-v1 in Section 9.2. It requires a significantly lower number of the keystream bits compared to [38], though with higher time complexity.

3 New Method

The new method may work faster than the multivariate correlation attack above and, in particular, its time complexity is lower than q^n. That is the main contribution of this study. The fast correlation attack in [30] applies to LFSR-based stream ciphers and exploits low-weight parity-checks, which may not exist or be scarce within the available length-N keystream. Since the system (4) is generally irrelevant to LFSRs and linear recurrences, relations of a more general nature are used instead.

Let B_r be an $r \times n$ matrix over $\mathbb{F}_q$ of rank r, where $1 \leq r \leq n$, and $Y = B_r X$. Also, $\langle\cdot\rangle$ will denote the linear space spanned by the rows of the specified matrices. We show how to test the values of Y.

Definition 1. *Let B_r be as above and A_i be the matrices in* (4)*. A set of indices $I \subseteq \{1, \ldots, N\}$ such that*

$$\langle A_i, i \in I\rangle \cap \langle B_r\rangle \neq \langle 0\rangle \tag{7}$$

is called a relation modulo B_r *and its* weight *is $|I|$. If its weight is small, I is said to be* short.

Let $t_{rI} > 0$ be the dimension of the space (7). This space is spanned by the rows of a matrix $T_{rI}B_r$, where T_{rI} is a matrix of size $t_{rI} \times r$ of rank t_{rI}. If I is a short relation, we may efficiently compute the conditional probability distribution p_{rI} given by

$$p_{rI}(v) = \Pr\left((T_{rI}B_r)X = v \,|\, A_i X = X_i, i \in I\right), \quad v \in \mathbb{F}_q^{t_{rI}}. \tag{8}$$

Let Y_I denote a random variable on $\mathbb{F}_q^{t_{rI}}$ with the distribution p_{rI} and let $\mathcal{I}_r$ be a set of relations modulo B_r. Then

$$T_{rI}Y = Y_I, \quad I \in \mathcal{I}_r,$$

is a system of equations of the same type as (4), but of a smaller dimension $r \le n$. Since X is uniformly distributed on $\mathbb{F}_q^n$, the random variable Y is uniformly distributed on $\mathbb{F}_q^r$. The multivariate method in Section 2 is then applied to solve the new system. That is, $b_r = B_rX$ is tested with

$$p_{rI}(b_{rI}) \neq 0, \quad I \in \mathcal{I}_r, \tag{9}$$

$$S_r(b_r) = \sum_{I\in\mathcal{I}_r} \ln p_{rI}(b_{rI}) > c_r, \tag{10}$$

where $b_{rI} = T_{rI}b_r$ and c_r is a threshold defined by the success probability of not rejecting the correct solution. Alternatively, we may use the FFT to compute the values of the statistic S_r if the probabilities $p_{rI}(v)$ are close enough to $q^{-t_{rI}}$.

For matrices B_r of large rank r, a straightforward application of the multivariate methods is inefficient since we need to run over q^r vectors b_r. The following test-and-extend algorithm is used instead. We choose a sequence of matrices B_r of rank $r = 1, \ldots, n$, such that B_r is a sub-matrix of B_{r+1} and obtain a set $\mathcal{I}_r$ of relations. The algorithm has two variations:

1. **Tree search.** The search starts at $r = 1$. The candidates for B_rX are tested with (9) and (10). The survivors are extended to candidates for $B_{r+1}X$ and tested. We continue in this fashion to obtain the survivors at level n. The cost of computing the values of the statistic for B_rX is proportional to $|\mathcal{I}_r|$. The method is implemented by traversing a tree.

2. **Hybrid method.** The search starts at $r = r_0$ for some parameter $r_0 \ge 1$. The FFT is applied to compute the values of the statistic for $B_{r_0}X$. Up to around 2^{r_0} relations may be used within essentially the cost of one application of the FFT. The candidates for $B_{r_0}X$ are ranked according to the value of the statistic. After that, the tree traversal is done as in the first variation starting with the most probable candidates for $B_{r_0}X$. As an option, one can brute force the values of $X = x$ such that $B_{r_0}x$ are most probable.

In Sections 4 and 5, we show how to compute short relations (7) and their probability distributions (8), respectively. The success probability of the algorithm and its complexity are studied in Section 6. Implementation details are presented in Section 7. In Section 8 the algorithm is applied to equations (2) constructed from some instances of the filter generator. Experimentally, the complexity of finding the solution is lower than q^n, even for some hard instances of (2), where the number of equations N is relatively low.

Vectorial parity-checks used in vectorial FCAs are similar to relations introduced here. However, the latter have a more general nature and may be seen as a generalisation of the former (which in turn generalise non-vectorial parity-checks). Additionally, the new method is not a decoding procedure as the ones used in FCAs. Thus, it is not possible to make a direct comparison with decoding algorithms in FCAs. (Moreover, the authors in [39] state that it is difficult to make a comparison of vectorial and non-vectorial decoding algorithms.)

A detailed example of the new method applied on a toy filter generator is in Appendix A. We refer to Section A.1 first, where the application of both variants of the method as described above is presented. Then, Section A.2 discusses the strategy to construct the matrices B_r and the sets $\mathcal{I}_r$. We refer to the latter after having read Sections 6–8.1.

4 Relations Modulo B_r

In this section, we present two methods to find short relations modulo B_r of weight d.

4.1 Brute Force

The relation (7) is equivalent to a system of homogeneous linear equations

$$\sum_{i\in I} v_iA_i = vB_r, \tag{11}$$

where the variables are vectors $v_i \in \mathbb{F}_q^{\ell_i}$, $i \in I$, and $v \in \mathbb{F}_q^r$ such that $v \neq 0$. The system incorporates n equations in $\sum_{i\in I} \ell_i + r$ variables from $\mathbb{F}_q$. One has to solve $\binom{N}{d}$ such systems to find all relations (7) modulo B_r of weight $\leq d$.

Let $\ell_i = \ell$ for $1 \leq i \leq N$. We may expect to find at least one relation (7) if $N > (d/e)\, q^{\frac{n-d\ell-r+1}{d}}$. Indeed, there are $q^{\ell d} - 1$ non-zero vectors in the left hand sides of (11) for every $I = \{i_1, \ldots, i_d\}$ if dependencies between the rows of A_i, $i \in I$, are neglected. The probability that one random vector hits the space $\langle B_r\rangle$ is q^{r-n}. If a vector belongs to $\langle B_r\rangle$, then its multiples by non-zero constants belong to $\langle B_r\rangle$ too. For $\ell d + r < n$, the probability that two non-collinear vectors for the same I hit $\langle B_r\rangle$ is negligible. The average number of the relations (11) and therefore (7) is around

$$\frac{\binom{N}{d}(q^{\ell d} - 1)}{q^{n-r}(q-1)}. \tag{12}$$

For small d and large N, we have $\binom{N}{d} \approx \frac{N^d}{d!}$. That implies the bound for N. The expression (12) is a rather accurate estimate for the actual number of relations modulo B_r for the parameters in Section 8.2.

4.2 Lattice Reduction

Assume that q is a small prime number. Let A be a vertical concatenation of the matrices $A_1, \ldots, A_N$. Thus, A is a matrix with $m = \sum_{i=1}^N \ell_i$ rows, n columns and integer entries. Let L denote a lattice of all integer vectors v of length m such that $vA \in \langle B_r\rangle$ modulo q. Clearly, if (11) holds, then

$$(0, \ldots, 0, v_{i_1}, 0 \ldots, 0, v_{i_d}, 0 \ldots, 0) \in L.$$

That is a relatively short vector in the lattice since its norm is $\leq \frac{q}{2}(\sum_{i\in I} \ell_i)^{1/2}$.

The rank of the lattice L is m and the volume is q^{n-r}, the basis is easy to construct. A reduction algorithm (e.g., [26]) is applied to compute the reduced basis. Short vectors are extracted and checked. If d is small enough, a short relation (7) is found. Since we may want many short relations, the initial basis of L is modified and the reduction algorithm is applied again.

5 Computing the Distributions p_{rI}

In this section, we present four different methods to compute the probabilities (8). To simplify notation, let $I = \{1, \ldots, d\}$ and $\mathcal{C}$ denote the event $A_iX = X_i, i \in I$. Also, let V be a matrix of size $t \times n$ and rank t such that the rows of V are in the space generated by the rows of $A_1, \ldots, A_d$. Then

$$p_{rI}(v) = \Pr(VX = v \,|\, \mathcal{C}),$$

where $V = T_{rI}B_r$. The results are summarised in Table 3, where $R = \sum_{i=1}^d q^{\ell_i}$ and $\ell_i = \mathrm{rank}(A_i)$. The term R appears in all methods because the corresponding computations depend on all $\sum_{i=1}^d q^{\ell_i}$ probability values.

Method	Formula	Complexity	Comments
Section 5.1	(13)	$dq^n + R$	-
Section 5.2	(14)	$dq^{\mathrm{rank}(A)} + R$	$A = (A_1, \ldots, A_d)$
Section 5.4	(16)	$dq^{\mathrm{rank}(W)} + R$	$\langle A_1\rangle, \ldots, \langle A_d\rangle$ lin. indep. mod $\langle W\rangle$ and $\langle V\rangle \subseteq \langle W\rangle$
Section 5.5	(17)	$dq^{2\,\mathrm{rank}(V)} + R$	$A_1, \ldots, A_d$ lin. indep.

Table 3: Summary of the methods for computing $\Pr(VX = v \,|\, \mathcal{C})$.

The first three methods are universal and the third one is the fastest of the three. The convolution method in Section 5.5 may be even faster. That works if the rows of $A_1, \ldots, A_d$ are linearly independent. Remark that, even if $A_1, \ldots, A_d$ are linearly independent, the matrix W of smallest rank such that $\langle A_1\rangle, \ldots, \langle A_d\rangle$ are linearly independent modulo $\langle W\rangle$ and $\langle V\rangle \subseteq \langle W\rangle$ may be $A = (A_1, \ldots, A_d)$. For instance, let A_1, A_2, A_3 be linearly independent rows ($\ell_1 = \ell_2 = \ell_3 = 1$) and $V = A_1 + A_2 + A_3$. Then $W = (A_1, A_2, A_3)$ and $\mathrm{rank}(W) = 3$. The method from Section 5.5 is faster in that case as $\mathrm{rank}(V) = 1$.

5.1 Basic Formula

By the conditional probability formula,

$$p_{rI}(v) = \Pr(VX = v, \mathcal{C}) / \Pr(\mathcal{C}).$$

Since $X, X_1, \ldots, X_d$ are independent and X is uniformly distributed on $\mathbb{F}_q^n$, we have

$$\begin{aligned} \Pr(VX = v, \mathcal{C}) &= \sum_{x \in \mathbb{F}_q^n : Vx = v} \Pr(X = x, X_1 = A_1 x, \ldots, X_d = A_d x) \\ &= \frac{1}{q^n} \sum_{x \in \mathbb{F}_q^n : Vx = v} \prod_{j=1}^{d} P_j(A_j x). \end{aligned} \tag{13}$$

In order to compute $p_{rI}(v)$, it is enough to compute only $\Pr(VX = v, \mathcal{C})$ for each $v \in \mathbb{F}_q^t$ since $\Pr(\mathcal{C}) = \sum_v \Pr(VX = v, \mathcal{C})$. The whole computation takes dq^n operations.

5.2 Change of Variables

Let $k = \dim_{\mathbb{F}_q} \langle A_1, \ldots, A_d \rangle$ and let U be a $(k \times n)$-matrix constructed with linearly independent rows of $A_1, \ldots, A_d$. Then $A_j = A'_j U$ and $V = V'U$ for some matrices A'_j and V'. Let $Z = UX$. So $A_j X = A'_j Z$ and $VX = V'Z$, and (13) implies

$$\Pr(VX = v, \mathcal{C}) = \frac{1}{q^k} \sum_{z \in \mathbb{F}_q^k : V'z = v} \prod_{j=1}^{d} P_j(A'_j z), \tag{14}$$

There are at most q^k terms in the sums (14) for all v and each term is a product of d numbers. Therefore, the cost of the computation is dq^k operations.

5.3 Independence in $A_1, \ldots, A_d$ modulo $\langle W \rangle \supseteq \langle V \rangle$

Another method for computing probabilities $p_{rI}(v)$ is presented in this section and the following one. It may be efficient even if $k = \dim_{\mathbb{F}_q} \langle A_1, \ldots, A_d \rangle$ is large. Let W be a matrix of size $l \times n$ over $\mathbb{F}_q$ and of rank l. The linear spaces

$$\langle A_1 \rangle, \ldots, \langle A_d \rangle \tag{15}$$

are called linearly independent modulo $\langle W \rangle$ if $\sum_{i=1}^{d} a_i \in \langle W \rangle$ and $a_i \in \langle A_i \rangle$ imply $a_i \in \langle W \rangle$. We will show how to construct a matrix W of lowest rank such that $\langle V \rangle \subseteq \langle W \rangle$ and (15) are linearly independent modulo $\langle W \rangle$. Then in Section 5.4, we will give a formula to compute

$$\Pr(WX = w, \mathcal{C})$$

for every $w \in \mathbb{F}_q^l$. The probabilities $\Pr(VX = v, \mathcal{C})$ are then easy to deduce. The complexity of the computation is $dq^{\mathrm{rank}(W)}$ operations.

Let U be a space generated by all $b_1, \ldots, b_d$ such that $b_i \in \langle A_i \rangle$ and $b_1 + \cdots + b_d \in \langle V \rangle$. Let W be a matrix whose rows form a basis of U. Then the spaces (15) are linearly independent modulo W. Suppose $\sum_{i=1}^{d} a_i \in \langle W \rangle$ and $a_i \in \langle A_i \rangle$. We need to show that $a_i \in \langle W \rangle$. One has $\sum_{i=1}^{d} a_i \in \sum_{i=1}^{d} b_i + \langle V \rangle$, for some $b_i \in \langle A_i \rangle \cap \langle W \rangle$ by the definition of W. Then $\sum_{i=1}^{d} (a_i - b_i) \in \langle V \rangle$ and therefore $(a_i - b_i) \in \langle W \rangle$. Hence, $a_i \in \langle W \rangle$ for $i = 1, \ldots, d$. The spaces (15) are linearly independent. The rank of W is the lowest by construction. The following statement is then true.

Lemma 1. *W is a lowest rank matrix such that $\langle V \rangle \subseteq \langle W \rangle$ and (15) are linearly independent.*

We now show how to construct a basis of U. That may be done by solving a system of linear equations. Let $b_{i1}, \ldots, b_{it_i}$ be a basis for $\langle A_i \rangle / \langle V \rangle$, $i = 1, \ldots, d$. So $b_i = \sum_{j=1}^{t_i} \gamma_{ij} b_{ij} \in \langle A_i \rangle / \langle V \rangle$ for $\gamma_{i1}, \ldots, \gamma_{it_i} \in \mathbb{F}_q$. Therefore $b_1 + \cdots + b_d \in \langle V \rangle$ if and only if

$$\sum_{i=1}^{d} \sum_{j=1}^{t_i} \gamma_{ij} b_{ij} \in \langle V \rangle.$$

We take a set of linearly independent solutions. Each solution results in $b_1, \ldots, b_d$ and such b_i generate the space U. A basis of U forms the matrix W.

5.4 Formula for $\Pr(WX = w, C)$

Let W be a matrix constructed in Section 5.3 and $\mathrm{rank}(W) = l$. Suppose K is a matrix of size $n\times(n-l)$ and of rank $n-l$ such that $WK = 0$. Hence, $Wx = w$ if and only if $x = x_0 + Ky$, where y is a column vector of length $n-l$ and $Wx_0 = w$. Let V_i be the linear space spanned by the columns of A_iK and

$$\phi : \mathbb{F}_q^{n-l} \to V_1 \times \ldots \times V_d$$

be a linear mapping defined by $\phi(y) = (y_1, \ldots, y_d)$, where $y_i = A_iKy$.

Lemma 2. *The mapping ϕ is surjective and*

$$\begin{aligned}\Pr(WX = w, \mathcal{C}) &= \Pr(WX = w, A_1X = X_1, \ldots, A_dX = X_d)\\ &= \frac{|Ker\,\phi|}{q^n}\prod_{i=1}^{d}\sum_{y_i\in V_i} P_i(w_i + y_i),\end{aligned} \tag{16}$$

where $w_i = A_ix_0$.

Proof. Let's prove that ϕ is surjective. If not, then the values of ϕ belong to a proper subspace of $V_1\times\ldots\times V_d$. So there are $v_i \in \mathbb{F}_q^{\ell_i}$ such that $\sum_i v_iA_iKy = 0$ for every $y\in\mathbb{F}_q^{n-l}$ and there are non-zero vectors among $v_1A_1K, \ldots, v_dA_dK$. The equality $\sum_i v_iA_iKy = 0$ holds for any y if and only if $(\sum_i v_iA_i)K = 0$, and so $\sum_i v_iA_i \in \langle W\rangle$. By the definition of W, the latter implies $v_iA_i \in \langle W\rangle$. Hence $v_1A_1K = \cdots = v_dA_dK = 0$, which is a contradiction. Therefore ϕ is surjective.

Let's prove (16). By (13),

$$\Pr(WX = w, \mathcal{C}) = \frac{1}{q^n}\sum_{\substack{x\in\mathbb{F}_q^n:\\ Wx=w}}\prod_{i=1}^{d} P_i(A_ix) = \frac{1}{q^n}\sum_{\substack{y\in\mathbb{F}_q^{n-\ell}:\\ W(x_0+Ky)=w}}\prod_{i=1}^{d} P_i(A_ix_0 + A_iKy).$$

Hence,

$$\Pr(WX = w, \mathcal{C}) = \frac{|\mathrm{Ker}\,\phi|}{q^n}\sum_{\substack{y_1\in V_1,\\ \ldots\\ y_d\in V_d}}\prod_{i=1}^{d} P_i(w_i + y_i) = \frac{|\mathrm{Ker}\,\phi|}{q^n}\prod_{i=1}^{d}\sum_{y_i\in V_i} P_i(w_i + y_i).$$

□

Let r be the rank of the system of linear equations $\phi(y) = (0, \ldots, 0)$. So $|\mathrm{Ker}\,\phi| = q^{n-l-r}$. The values $\sum_{y_i\in V_i} P_i(w_i + y_i)$ may be precomputed for any i and $w_i \in \mathbb{F}_q^{\ell_i}$. It takes at most $\sum_{i=1}^{d} q^{\ell_i}$ operations. After that, the cost is dq^l operations. The overall cost is then $dq^l + \sum_{i=1}^{d} q^{\ell_i}$ operations. Recall that $l = \mathrm{rank}(W) \le k = \dim_{\mathbb{F}_q}\langle A_1, \ldots, A_d\rangle$. If $l < k$, this method is more efficient than the one in Section 5.2.

5.5 Convolution Formula

Let the rows in $A_1, \ldots, A_d$ be linearly independent. The probabilities $\Pr(VX = v \,|\, \mathcal{C})$ may be computed in $\sum_i q^{\ell_i} + dq^{2t}$ operations, where $t = \mathrm{rank}(V)$. Since $\langle V\rangle \subseteq \langle A_1, \ldots, A_d\rangle$, we can represent $V = \sum_{i=1}^{d} V_iA_i$, where V_i are matrices of size $(t\times\ell_i)$. This representation is unique and may be found by solving a system of linear equations.

Lemma 3.

$$\Pr(VX = v \,|\, A_1X = X_1, \ldots, A_dX = X_d) = \Pr\left(\sum_{i=1}^{d} V_iX_i = v\right). \tag{17}$$

Proof. Since the rows in $A_1, \ldots, A_d$ are linearly independent, $A_1X, \ldots, A_dX$ are independent uniformly distributed random variables. By the conditional probability formula,

$$\begin{aligned}\Pr(VX = v \,|\, A_1X = X_1, \ldots, A_dX = X_d) &= \frac{\Pr\left(\sum_{i=1}^{d} V_iA_iX = v, A_1X = X_1, \ldots, A_dX = X_d\right)}{\Pr(A_1X = X_1, \ldots, A_dX = X_d)}\\ &= \frac{\Pr\left(\sum_{i=1}^{d} V_iX_i = v, A_1X = X_1, \ldots, A_dX = X_d\right)}{\Pr(A_1X = X_1, \ldots, A_dX = X_d)},\end{aligned}$$

where

$$\Pr\left(A_1X = X_1, \dots, A_dX = X_d\right) = \prod_{i=1}^{d} \Pr(A_iX = X_i) = \prod_{i=1}^{d} 1/q^{\ell_i} = q^{-\sum_{i=1}^{d}\ell_i}$$

and

$$\begin{aligned}
\Pr\left(\sum_{i=1}^{d} V_iX_i = v, A_1X = X_1, \dots, A_dX = X_d\right) &= \sum_{\substack{v_1,\dots,v_d:\\ \sum_{i=1}^{d} V_iv_i = v}} \prod_{i=1}^{d} \Pr(A_iX = v_i) \prod_{i=1}^{d} P_i(v_i)\\
&= q^{-\sum_{i=1}^{d}\ell_i} \times \sum_{\substack{v_1,\dots,v_d:\\ \sum_{i=1}^{d} V_iv_i = v}} \prod_{i=1}^{d} P_i(v_i)\\
&= q^{-\sum_{i=1}^{d}\ell_i} \times \Pr\left(\sum_{i=1}^{d} V_iX_i = v\right),
\end{aligned}$$

with $v_i \in \mathbb{F}_2^{\ell_i}$. Therefore,

$$\Pr(VX = v \mid A_1X = X_1, \dots, A_dX = X_d) = \Pr\left(\sum_{i=1}^{d} V_iX_i = v\right).$$

□

It takes q^{ℓ_i} linear algebra operations to compute the distribution of V_iX_i . Then, $\Pr\left(\sum_{i=1}^{d} V_iX_i = v\right)$ may be computed iteratively by a convolution type formula because V_iX_i are independent. That takes dq^{2t} operations. The overall cost of computing the distribution $\Pr(VX = v \mid \mathcal{C})$ is $\sum_{i=1}^{d} q^{\ell_i} + dq^{2\,\mathrm{rank}(V)}$. According to Section 5.3, $\langle W\rangle = \langle V_1A_1, \dots, V_dA_d\rangle$ since the rows in $A_1, \dots, A_d$ are linearly independent. The cost to compute the conditional distribution $\Pr(WX = w \mid A_1X = X_1, \dots, A_dX = X_d)$ is $\sum_{i=1}^{d} q^{\ell_i} + dq^{\mathrm{rank}(W)}$. The conditional distribution on VX may be computed within the same cost since $\langle V\rangle \subseteq \langle W\rangle$. So, the convolution method is preferable if the rows $A_1, \dots, A_d$ are linearly independent and $\mathrm{rank}(V) < \mathrm{rank}(W)/2$.

6 Success Probability and Complexity of the Tree Search

To simplify notation, we may assume that $\mathcal{I}_1 = \mathcal{I}_2 = \cdots = \mathcal{I}_n = \mathcal{I}$. Every relation I for B_r is a relation for B_{r+1} according to the definition of B_r and the definition (7) of a relation. So $\mathcal{I}_r \subseteq \mathcal{I}_{r+1}$. However, a relation I modulo B_{r+1} may not be a relation modulo B_r. In the latter case, we can still consider I as a trivial relation for B_r because we have $\langle 0\rangle$ in the right hand side of (7). Then $t_{rI} = 0$ and $p_{rI}(0) = 1$ for such I. Thus we can formally augment the set $\mathcal{I}_r$ and get $\mathcal{I}_r = \mathcal{I}_{r+1}$.

6.1 Success Probability

The execution of the algorithm is a success if $b_r = B_rx$ was not rejected for every $r = 1, \dots, n$, where $X = x$ is the correct solution to (4). The success probability β is defined by

$$\beta = \Pr(S_r(b_r) \geq c_r,\ r = 1, \dots, n \mid A_iX = X_i, i = 1, \dots, N).$$

We will show how to compute the thresholds $c_1, \dots, c_n$ given β. Let

$$\mathcal{S} = \begin{pmatrix} S_1(b_1)\\ \vdots\\ S_n(b_n)\end{pmatrix} = \sum_{I\in\mathcal{I}} \mathcal{S}_I, \quad \mathcal{S}_I = \begin{pmatrix} \ln p_{1I}(b_{1I})\\ \vdots\\ \ln p_{nI}(b_{nI})\end{pmatrix}, \quad c = \begin{pmatrix} c_1\\ \vdots\\ c_n\end{pmatrix},$$

where $b_{rI} = T_{rI}b_r$ by the definition of the statistic S_r in (10). The inequalities $S_r(b_r) \geq c_r$ may be written entry-wise as $\mathcal{S} \geq c$. So $\beta = \Pr(\mathcal{S} \geq c \mid A_iX = X_i, i = 1, \dots, N)$.

As B_r is a submatrix of B_{r+1} in its first r rows, we choose the matrices T_{rI} in (7) such that T_{rI} is a submatrix of $T_{r+1\,I}$ in its first t_{rI} rows and first r columns as below

$$T_{r+1\,I} = \begin{pmatrix} & 0\\ T_{rI} & \vdots\\ & 0\\ * & * \end{pmatrix}.$$

So, $b_{rI} = T_{rI}\, b_r$ is a subvector of $b_{r+1\,I} = T_{r+1\,I}\, b_{r+1}$ in its first t_{rI} entries. The mean of $\mathcal{S}_I$ is

$$\mu_I = \begin{pmatrix} \mu_{1I} \\ \mu_{2I} \\ \vdots \\ \mu_{nI} \end{pmatrix}, \quad \mu_{rI} = \sum_{v \in \mathbb{F}_q^{t_{rI}}} p_{rI}(v) \ln p_{rI}(v).$$

The mean of $\mathcal{S}$ is therefore $\mu = \sum_{I \in \mathcal{I}} \mu_I$. Let Q_I be the covariance matrix of $\mathcal{S}_I$. The entry in the row i and the column $j \geq i$ of Q_I is

$$\sum_{v \in \mathbb{F}_q^{t_{jI}}} p_{jI}(v) \ln p_{iI}(v_i) \ln p_{jI}(v) - \mu_{iI}\mu_{jI},$$

where v_i is the vector in the first t_{iI} entries of v. This is because $j \geq i$ and $b_{iI} = T_{iI} b_i$ is the vector in the first t_{iI} coordinates of $b_{jI} = T_{jI} b_j$.

The distribution of $\mathcal{S}_I$ only depends on the distribution of X_i, $i \in I$. If any distinct relations $I, J \in \mathcal{I}$ are disjoint, then $\mathcal{S}_I$, $I \in \mathcal{I}$, are independent and the covariance matrix of $\mathcal{S}$ is $Q = \sum_{I \in \mathcal{I}} Q_I$. In practice, the sets I are small (of size at most d) random-looking subsets of $\{1, \ldots, N\}$. They are mostly pairwise disjoint. For the same reason, for large enough $|\mathcal{I}|$, the sum $\mathcal{S} = \sum_{I \in \mathcal{I}} \mathcal{S}_I$ approximately follows the multivariate normal distribution $\mathbf{N}(\mu, Q)$. Our experiments with the filter generator in Section 8 fit well this assumption. Given β, the threshold c such that $\Pr(\mathbf{N}(\mu, Q) \geq c) = \beta$ can be computed.

6.2 Number of Tree Nodes at Level r

The complexity of the algorithm is defined by the number of nodes visited during the search. At level r a current node b_r is tested with (9) and (10). The number of nodes at level r is the number of survivors b_{r-1} times q. We show how to compute the number of incorrect survivors b_r.

Let X be taken from the uniform distribution on $\mathbb{F}_q^n$. Therefore, $b_r = B_r X$ is uniformly distributed on $\mathbb{F}_q^r$ and $b_{rI} = T_{rI} b_r$ is uniformly distributed on $\mathbb{F}_q^{t_{rI}}$. Let $\mathcal{E}_{rI}$ denote the event $p_{rI}(b_{rI}) \neq 0$. Then, $\Pr(\mathcal{E}_{rI}) = K_{rI}/q^{t_{rI}}$, where K_{rI} is the size of the support for the distribution p_{rI}. Let $\mathcal{E}_r$ be the joint event $\{\mathcal{E}_{rI}, I \in \mathcal{I}\}$. If any distinct $I, J \in \mathcal{I}$ are disjoint, then the events $\mathcal{E}_{rI}$ are independent. In practice, that is likely to happen, so we may assume

$$\varepsilon_r = \Pr(\mathcal{E}_r) = \prod_{I \in \mathcal{I}} K_{rI}/q^{t_{rI}}.$$

Let

$$\mathcal{S}(r) = \begin{pmatrix} S_1(b_1) \\ \vdots \\ S_r(b_r) \end{pmatrix}, \quad c(r) = \begin{pmatrix} c_1 \\ \vdots \\ c_r \end{pmatrix},$$

where c_i are found from $\Pr(\mathbf{N}(\mu, Q) \geq c) = \beta$ in Section 6.1. The current b_r passes the tests up to level r if and only if $\mathcal{E}_r$ holds and $\mathcal{S}(r) > c(r)$. The probability of this event is

$$\Pr(\mathcal{S}(r) > c(r), \mathcal{E}_r) = \Pr(\mathcal{E}_r) \cdot \Pr(\mathcal{S}(r) > c(r) \,|\, \mathcal{E}_r).$$

We will show how to compute $\alpha(r) = \Pr(\mathcal{S}(r) > c(r) \,|\, \mathcal{E}_r)$. We can write $\mathcal{S}(r) = \sum_{I \in \mathcal{I}} \mathcal{S}_I(r)$, where

$$\mathcal{S}_I(r) = \begin{pmatrix} \ln p_{1I}(b_{1I}) \\ \vdots \\ \ln p_{rI}(b_{rI}) \end{pmatrix}.$$

Let $\mu_{rI} = \begin{pmatrix} \mu_{1I} \\ \vdots \\ \mu_{rI} \end{pmatrix}$ be the mean vector and let Q_{rI} be the covariance matrix of $\mathcal{S}_I(r)$. Since b_{jI} is a vector in the first t_{jI} entries of b_{rI}, then

$$\mu_{jI} = \frac{\sum_{v_r \in \mathbb{F}_q^{t_{rI}} : p_{rI}(v_r) \neq 0} \ln p_{jI}(v_j)}{K_{rI}},$$

where v_j is the first t_{jI} entries of v_r. Notice that $p_{rI}(v_r) \neq 0$ implies $p_{jI}(v_j) \neq 0$. The entry in the row i and the column j of the covariance matrix Q_{rI} is

$$\sum_{v_r \in \mathbb{F}_q^{t_{rI}} : p_{rI}(v_r) \neq 0} \frac{\ln p_{iI}(v_i) \ln p_{jI}(v_j)}{K_{rI}} - \mu_{iI}\mu_{jI}.$$

For large $|\mathcal{I}|$, the random variable $\mathcal{S}(r) = \sum_{I \in \mathcal{I}} \mathcal{S}_I(r)$ approximately follows a multivariate normal distribution $\mathbf{N}(\mu_r, Q_r)$, where $\mu_r = \sum_{I \in \mathcal{I}} \mu_{rI}$ and $Q_r = \sum_{I \in \mathcal{I}} Q_{rI}$. Therefore $\alpha(r) \approx \Pr(\mathbf{N}(\mu_r, Q_r) > c(r))$. The number of incorrect b_r which pass the test at level r is approximately

$$\varepsilon_r \cdot \alpha(r) \cdot q^r.$$

6.3 Complexity

For $N > (d/e)\, q^{\frac{n-d\ell-r+1}{d}}$, according to Section 4.1, we may expect non-trivial relations modulo B_r of weight at most d. Short relations are computed by brute force or lattice reduction. The search for relations is fully parallelisable. For small d, the distributions p_{rI} are relatively easy to compute and more likely to be non-uniform. However, we do not expect many useful relations if N is moderate and r is small. For larger r, we can have numerous useful relations. On the other hand, computing the distributions p_{rI} may be tedious for larger d and the distributions tend to be uniform. We need $|\mathcal{I}_r|$ arithmetic operations with real numbers to compute the statistic S_r in (10) for each visited node at level r. So the complexity of the tree search is $\sum_{r=0}^{n-1} \varepsilon_r \cdot \alpha(r) \cdot q^{r+1} \cdot |\mathcal{I}_{r+1}|$ arithmetic operations, where we set $\varepsilon_0 = 1, \alpha(0) = 1$.

7 Algorithm Implementation Details

The algorithm comprises two stages: precomputation and tree search. Let a small d be a parameter. In the first stage, a set of matrices B_r of rank $r = 1, \ldots, n$ are chosen; these matrices are such that B_r is a submatrix of B_{r+1} in its first r rows. Then, we obtain relations modulo B_r of weight at most d (see Section 4) and construct the sets $\mathcal{I}_r$ together with the computation of the probability distributions p_{rI}, $I \in \mathcal{I}_r$, defined by (8) (see Section 5). Finally, the thresholds c_r are calculated such that the correct solution is found with a predefined success probability β after the algorithm terminates (see Section 6). That defines the statistical tests (9), (10) for $1 \leq r \leq n$. In the second stage, the candidate solutions for $B_r X$ are tested, the survivors are extended to candidate solutions for $B_{r+1}X$ and tested, for $1 \leq r \leq n-1$. This stage is implemented with a tree search. By (10), the complexity of computing the statistic S_r is proportional to $|\mathcal{I}_r|$. We can afford using only a limited number of the relations $\mathcal{I}_r$ with this variation. Therefore, to construct the sets $\mathcal{I}_r$, all obtained relations are ranked and the best ones are selected; this is explained in Section 7.1. Section 7.2 presents the tree search.

Alternatively, the FFT may be used (i.e., hybrid method). We choose a parameter $1 \leq r_0 \leq n$. In the first stage we construct the sets $\mathcal{I}_r$ and compute p_{rI}, $I \in \mathcal{I}_r$, as above, however, we do it for $r = r_0 + 1, \ldots, n$. We also obtain a large set $\mathcal{I}_{r_0}$ of relations modulo B_{r_0}. In the second stage, the values of the statistic $S_{r_0}(b)$ are computed for all $b \in \mathbb{F}_q^{r_0}$ with the FFT (see Section 2.4). The candidates $b = B_{r_0}X$ are ranked according to the values $S_{r_0}(b)$. Then, starting with the most probable candidates, they are extended to candidate solutions for $B_{r_0+1}X$ and tested, the survivors are further extended and tested, etc. This variation requires space of order q^{r_0} to execute the FFT. We can use up to q^{r_0} relations in $\mathcal{I}_{r_0}$ within the cost of one FFT application. Experimentally, that reduces the size of the tree search. Since $|\mathcal{I}_{r_0}|$ is large, there may be dependencies between the summands in the definition (10) of the statistic S_r for $r = r_0$ and a normal approximation to the distribution of S_{r_0} may not be very accurate.

7.1 Ranking Relations

The complexity of the tree search for every level r is influenced by the number of relations $\mathcal{I}_r$ as the statistic (10) incorporates $|\mathcal{I}_r|$ summands. The available relations may be ranked according to the size of the support and the entropy (alternatively, the quadratic imbalance) of the distribution p_{rI} on $\mathbb{F}_q^{t_{rI}}$. Inferior relations are then filtered out. The size of the support of $I \in \mathcal{I}_r$ is the number K_{rI} of $v \in \mathbb{F}_q^{t_{rI}}$

such that $p_{rI}(v) \neq 0$. The normalised q-ary entropy is

$$H(p_{rI}) = -\sum_{v \in \mathbb{F}_q^{t_{rI}}} p_{rI}(v) \log_q p_{rI}(v) - t_{rI}.$$

Let $I, J \in \mathcal{I}_r$. We say that I is a better distinguisher than J (i.e., further away from being uniform) if

1. $\frac{K_{rI}}{q^{t_{rI}}} < \frac{K_{rJ}}{q^{t_{rJ}}}$, or
2. $H(p_{rI}) < H(q_{rJ})$ if $\frac{K_{rI}}{q^{t_{rI}}} = \frac{K_{rJ}}{q^{t_{rJ}}}$.

A number of best relations are used to construct the statistic S_r in (10).

7.2 Tree Search

Let $b = (a_1, a_2, \dots, a_n) \in \mathbb{F}_q^n$. For $1 \leq r \leq n$, we denote $b_r = (a_1, a_2, \dots, a_r)$, so that $b_n = b$. To simplify notation, we will use a predicate R_r. We say $R_r(b_r) = 1$ if both conditions (9) and (10) are satisfied, and $R_r(b_r) = 0$ otherwise. The task is to find b such that

$$R_1(b_1) = 1, \dots, R_n(b_n) = 1. \tag{18}$$

The solving algorithm is implemented by traversing a tree, where b_r is tested at level r. If $R_r(b_r) = 1$, then b_r is extended to b_{r+1} and tested. If $R_r(b_r) = 0$, that branch is not explored and the search backtracks. Whenever the last level n is reached, the value of b_n is a solution to (18). Generally, the tree search finds candidate solutions to (4). In our experiments with the filter generator, the correct solution is always found. In some cases, it is unique at an early level $r < n$.

8 Key Recovery Attacks for the Filter Generator

In this section we apply the new method for cryptanalysis of the filter generator.

8.1 Constructing B_r and relations

For the filter generator, the matrix B_n, and therefore its sub-matrices B_r, and the relations in $\mathcal{I}_r$ were chosen according to the following principles:

1. Each row of B_n is randomly taken from the vectors $aA_i, i = 1, \dots, N$, where $a = (a_1, \dots, a_\ell)$ and the linear Boolean function $a(x_1, \dots, x_\ell) = a_1x_1 + \dots + a_\ell x_\ell$ is one of the best linear approximations to the function f. The vector aA_i belongs to the space generated by the rows of A_i. There are at least r relations I modulo B_r of weight 1, thus providing with a few good distributions $p_{r,I}$.

2. Even though we may find numerous relations, only a bounded number of them may be used for the tree search. We try to choose $\mathcal{I}_r$ such that $I \cap J = \emptyset$ for distinct $I, J \in \mathcal{I}_r$, that is, all the relations are pairwise disjoint. In that case, the distribution of the statistic S_r may be well approximated with the Central Limit Theorem. Also, the sets $\mathcal{I}_r$ are chosen to be disjoint. The tests (9) and (10) may be considered independent for $r = 1, \dots, n$. So the statistics S_r, $r = 1, \dots, n$, are independent and the covariance matrix Q for their joint distribution is diagonal. That allows our experimental results to be as close as possible to the theoretical analysis based on the normal approximation to the distribution of $\mathcal{S}_r$.

3. In contrast, when using the FFT at stage r_0 (hybrid method), we can afford a large number of relations in $\mathcal{I}_{r_0}$ with almost no additional cost. That significantly reduces the time complexity of the whole attack. However, the Central Limit Theorem (CLT) does not seem to hold for the statistic S_{r_0}, since the sum (10) representing the statistic may contain a large number of dependent terms. At least in the case when b is distributed uniformly on $\mathbb{F}_q^{r_0}$ (as in Section 6.2 where the number of wrong survivors is computed), the distribution of S_{r_0} is normal, but the parameters are different from those calculated with the CLT. Hence, the complexity of the FFT based attack is generally more difficult to predict.

4. Let I be a relation modulo B_r. Then, $j \in I$ is called *irrelevant* if $v_j = 0$ for every solution v_i, $i \in I$, and $v \neq 0$ to (11). That means that the distribution $p_{r,I}$ does not depend on X_j, even if $j \in I$. The other indices in I are called *relevant* modulo B_r. Two relations I_1 and I_2 modulo B_r are *equivalent* if their set of relevant indices coincide. For each B_r, we apply the ranking criteria in Section 7.1 on the classes of equivalent relations and choose a suitable number of them to create $\mathcal{I}_r$.

Section A.2 illustrates the principles above for the toy example in Section A.1.

8.2 Experimental Results on the Filter Generator

We now present results of the new method applied to four hard instances of the filter generator. The method requires a significantly lower number of keystream bits than FCAs, methods based on the Berlekamp-Massey algorithm and fast algebraic attacks as in [11, 10]. So, we mainly compare the efficiency of the new method with brute force. The latter requires $2^n - 1$ trials of the LFSR initial state. On each candidate, we clock the LFSR and generate n bits of the keystream. Therefore, brute force takes essentially $n2^n$ bit operations, where we neglected the cost of clocking the LFSR.

In the first two experiments, $n = 40$ and the filtering functions f depend on $\ell = 5$ and 7 variables, respectively. We were able to explicitly recover the LFSR initial state with $N = 5\,000$ keystream bits and significantly faster than brute force. The best complexity was achieved with a combination of FFT and tree search, that is, $2^{32.06}$ and $2^{35.19}$ additions of reals, respectively, to compute the values of the statistics. The results closely fit the theoretical prediction. In the last two experiments, $n = 64$ and 80, respectively, $\ell = 5$ and $N = 10\,000$. The tree search was executed up to some intermediate level. The complexity was then extrapolated to the whole tree. Again, the best result was achieved with a combination of FFT and tree search, that is, $2^{57.39}$ and $2^{70.95}$ additions of reals, respectively.

In the experiments below, we used instances of the filter generator which employ "components" from the existing literature, such as the widely used degree-40 feedback polynomial in [24] and the Boolean function from Grain-v1. In the first three experiments, we used feedback polynomials with high weight and input indices k_i that maximise the memory (i.e., $k_\ell - k_1 \approx n$). In the last experiment, we follow closely the definition of the LFSR in Grain-v1, but maximise the memory when choosing the last input to the filtering function. Under various criteria (for example [16]), the devices are hard instances of the filter generator.

The statistical software R [34] was used to get the vector c which defines the tests (10) and the probabilities $\alpha(r)$ in Sections 6.1 and 6.2.

8.2.1 Experiments

In all experiments the matrices B_r were created as in Section 8.1, we applied brute force to obtain a set $\mathcal{I}$ of relations modulo B_{r_1}, for some value r_1, and created the (disjoint) sets $\mathcal{I}_r$ by choosing relations from $\mathcal{I}$. Table 4 shows the values used in the experiments. Then, for all experiments, we computed the covariance matrix and mean vector for the multivariate distributions to get the vector c of thresholds using a success probability $\beta = 0.9$.

Experiment 1. The polynomial g is a common choice in the literature, f is the one used in Grain-v1, the input spacings to f are coprime and span the whole register:

- $g(x) = x^{40} + x^{38} + x^{33} + x^{32} + x^{29} + x^{27} + x^{25} + x^{21} + x^{19} + x^{17} + x^{12} + x^{11} + x^{9} + x^{5} + x^{3} + x + 1;$
- $f(x_1, \ldots, x_5) = x_2 + x_5 + x_1x_4 + x_3x_4 + x_4x_5 + x_1x_2x_3 + x_1x_3x_4 + x_1x_3x_5 + x_2x_3x_5 + x_3x_4x_5;$
- $(k_1, \ldots, k_5) = (0, 7, 15, 26, 39).$

The tree search found a unique solution corresponding to the correct initial state. Figure 2 shows the number of theoretical and experimental survivors from the tree search. The maximum of theoretical survivors is $2^{28.59}$ at $r = 30$. Experimentally, it was $2^{28.34}$. Since $|\mathcal{I}_{30}| = 300$, the theoretical complexity is $2^{36.82}$ and in practice it was $2^{36.57}$ additions of reals to compute the values of the statistics in the right hand side of (10). We applied the hybrid approach with $r_0 = 20$ using all $2\,269$ relations modulo B_{20} with non-uniform distributions. The complexity of the FFT is $O\left(2^{24.32}\right)$ operations. Then, the correct initial state was recovered after executing the tree search on $32\,603 \approx 2^{15}$ candidate solutions starting at level $r = 20$. Figure 2 shows the number of survivors with the hybrid approach at levels $r \geq 20$. The maximum number of survivors is $2^{23.84}$ at $r = 30$. Since $|\mathcal{I}_{30}| = 300$, the complexity of the tree search

	Linear approx.	r_1	d	$\|\mathcal{I}\|$	$\|\mathcal{I}_r\|$
Experiment 1	$x_1 + x_3 + x_4$	10	3	15 000	50 for $r = 1, \ldots, 10$ 150 for $r = 11, \ldots, 20$ 300 for $r = 21, \ldots, 40$
Experiment 2	$x_1 + x_4 + x_5 + x_6 + x_7$	5	3	15 000	50 for $r = 1, \ldots, 10$ 150 for $r = 11, \ldots, 20$ 300 for $r = 21, \ldots, 40$
Experiment 3	$x_4 + x_5$	32	3	100 000	100 for $r = 1, \ldots, 20$ 250 for $r = 21, \ldots, 30$ 400 for $r = 31, \ldots, 50$ 500 for $r = 51, \ldots, 64$
Experiment 4	$x_4 + x_5$	40	3	49 657	50 for $r = 1, \ldots, 20$ 200 for $r = 21, \ldots, 45$ 400 for $r = 46, \ldots, 60$ 500 for $r = 61, \ldots, 80$

Table 4: Details of the experiments.

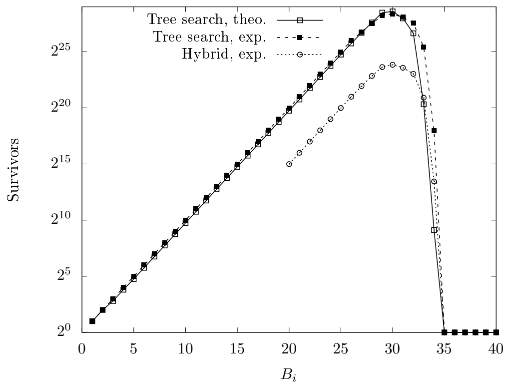

Figure 2: Number of survivors for experiment 1.

is $2^{32.06}$ additions with reals. The hybrid approach performs better in this case compared to only using the tree search.

Experiment 2. Device taken from [28], the authors did not specify the input spacings to f, in our case, they are coprime and span the whole register.

- $g(x) = x^{40} + x^{38} + x^{33} + x^{32} + x^{29} + x^{27} + x^{25} + x^{21} + x^{19} + x^{17} + x^{12} + x^{11} + x^{9} + x^{5} + x^{3} + x + 1$;
- $f(x_1, \ldots, x_7) = 1 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_1x_7 + x_2(x_3 + x_7) + x_1x_2(x_3 + x_6 + x_7)$;
- $(k_1, \ldots, k_7) = (0, 3, 8, 15, 26, 31, 39)$.

The tree search found 14 solutions which included the one corresponding to the correct initial state. Figure 3 shows the number of theoretical and experimental survivors from the tree search. The maximum of theoretical survivors is $2^{30.31}$ at $r = 32$. Experimentally, it was $2^{27.83}$. Since $|\mathcal{I}_{30}| = 300$, the theoretical complexity is $2^{38.54}$ operations and in practice it was $2^{36.06}$. We applied the hybrid approach with $r_0 = 20$ using all 261 relations modulo B_{20} with non-uniform distributions. The complexity of the FFT is negligible as above. Then, the correct initial state was recovered after executing the tree search on $259\,039 \approx 2^{18}$ candidate solutions starting at level $r = 20$. Figure 3 shows the number of survivors with the hybrid approach at levels $r \geq 20$. The maximum number of survivors is $2^{26.96}$ at $r = 32$. Since $|\mathcal{I}_{30}| = 300$, the complexity of the tree search is $2^{35.19}$ operations. The hybrid approach performs better in this case compared to only using the tree search.

Experiment 3. The polynomial g was chosen at random with high weight, f is the one used in Grain-v1, the input spacings to f are coprime and span the whole register.

- $g(x) = x^{64} + x^{62} + x^{55} + x^{49} + x^{44} + x^{42} + x^{37} + x^{24} + x^{23} + x^{20} + x^{16} + x^{15} + x^{10} + x^{8} + x^{6} + x^{2} + 1$;
- $f(x_1, \ldots, x_5) = x_2 + x_5 + x_1x_4 + x_3x_4 + x_4x_5 + x_1x_2x_3 + x_1x_3x_4 + x_1x_3x_5 + x_2x_3x_5 + x_3x_4x_5$;

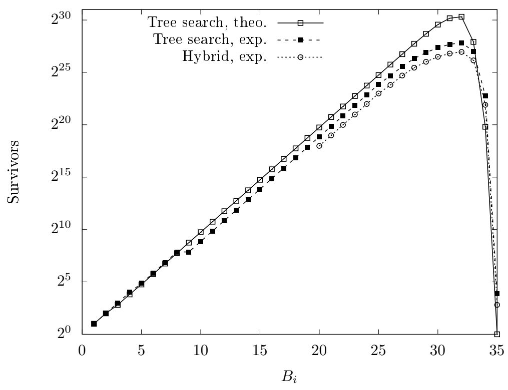

Figure 3: Number of survivors for experiment 2.

- $(k_1, \ldots, k_5) = (0, 22, 43, 61, 63)$.

We estimated the theoretical complexity first and, given the number of expected survivors, we executed the tree search up to level 36 only. Figure 4 shows the number of theoretical and partial experimental survivors from the tree search. The maximum of theoretical survivors is $2^{50.67}$ at $r = 52$. Since $|\mathcal{I}_{52}| = 500$, the theoretical complexity is $2^{59.64}$ operations. At level 36, we got $2^{35.75}$ survivors from the tree search and $2^{35.73}$ survivors theoretically. Since the number of experimental survivors follows very closely the theoretical curve, we expect the experimental complexity to be about $2^{59.64}$. We applied the hybrid approach using 1 115 relations with B_{20}. These are all the relations in $\mathcal{I} \bmod B_{20}$ whose support have a non-uniform probability distribution (at level 20). Due to the potential high number of survivors, we executed the tree search part up to level 36 as well. For this partial experiment, however, we knew in advance the candidate at level 20 corresponding to the correct initial state. Otherwise, we would not have been able to know where to stop the tree search and it would be equivalent to brute force all candidates at level 20. Figure 4 shows the number of survivors with the hybrid approach. We got $2^{32.42}$ candidates at level 36. Hence, the hybrid approach also performs better than using the tree search only. As in the previous experiments, the complexity of the FFT is negligible compared to the tree search part. We assume that the number of survivors for the hybrid method follows the behaviour of the tree search only, as in the previous experiments. In the worst case, the tree search part of the hybrid method will not reject any candidates up to level 52, that is, there should be $2^{48.42}$ survivors at this level. Since $|\mathcal{I}_{52}| = 500$, the worst case complexity is about $2^{57.39}$ operations.

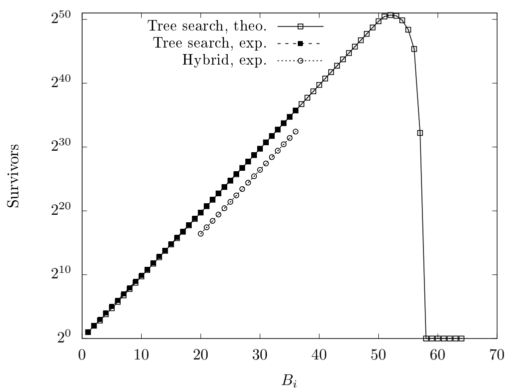

Figure 4: Number of survivors for experiment 3.

Experiment 4. The polynomial g, the function f and the indices k_i are taken from Grain-v1. In that cipher, the last input to f comes from its NFSR (see Section 9); here it comes from the last cell of the LFSR ($k_5 = 79$) to increase the memory.

- $g(x) = x^{80} + x^{62} + x^{51} + x^{38} + x^{23} + x^{13} + 1$;
- $f(x_1, \ldots, x_5) = x_2 + x_5 + x_1x_4 + x_3x_4 + x_4x_5 + x_1x_2x_3 + x_1x_3x_4 + x_1x_3x_5 + x_2x_3x_5 + x_3x_4x_5$;

- $(k_1, \ldots, k_5) = (3, 25, 46, 64, 79)$.

We also executed the tree search up to level 36 given the high theoretical complexity. Figure 5 shows the number of theoretical and experimental survivors (up to level 36) from the tree search. The maximum of theoretical survivors is $2^{63.9}$ at B_{65} giving the theoretical complexity $2^{72.86}$ (since $|\mathcal{I}_{65}| = 500$). At level 36, we got $2^{34.72}$ survivors from the tree search and $2^{35.72}$ survivors theoretically. Since the number of experimental survivors follows very closely the theoretical curve, we can expect the experimental complexity to be about $2^{71.86}$ operations. We applied the hybrid approach using all 9 845 relations modulo B_{20} whose support have a non-uniform probability distribution (at level 20). We executed the tree search up to level 36 as well and, as in experiment 3, we knew in advance the correct candidate at level 20. Figure 5 shows the number of survivors with the hybrid approach. We got $2^{32.98}$ candidates at level 36. Hence, the hybrid approach also performs better than using the tree search only. As in the previous experiments, the complexity of the FFT is negligible compared to the tree search part. We assume that the number of survivors for the hybrid method follows the behaviour of the tree search only method, as in the previous experiments. In the worst case, the tree search part of the hybrid method will not reject any candidates up to level 65, that is $2^{61.98}$ survivors at level 65. Since $|\mathcal{I}_{65}| = 500$, the worst case complexity is about $2^{70.95}$ operations.

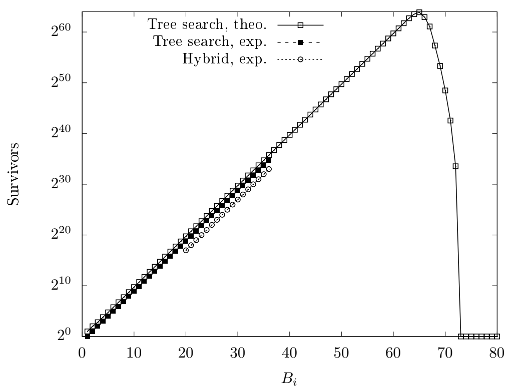

Figure 5: Number of survivors for experiment 4.

8.2.2 Analysis of experimental results

The relations (7) may be seen as a generalisation of the parity-checks used in FCAs. Some FCAs perform a partial brute force on a subset Γ of the LFSR's initial state bits as in [9]. We call parity-checks used in that work parity-checks modulo Γ. The expected number of weight-d parity-checks modulo Γ given the length-N keystream is $2^{|\Gamma|-n}\binom{N}{d-1}$, according to [9]. With the same weight, the set of relations modulo B_r, for an appropriate matrix B_r, incorporates parity-checks modulo Γ, where $|\Gamma| = r$. However, for the same number of keystream bits, the expected number (12) of relations modulo B_r is significantly higher. Table 5 compares these numbers for some explicit parameters. The value of a relation I is defined by the quality of the distribution (8). A large pool of the relations is constructed during precomputation, then we choose those to use during the attack.

That explains why our method requires less keystream bits to recover the LFSR initial state compared to FCAs and is still faster than brute force. Table 6 shows a comparison of the data and time complexity of our method with some reported FCAs. Recall that $p = \Pr(s_i = z_i)$. For experiment 1, there is no result reported for the same parameters, but we compare it against the attack in [28] since f yields $1 - p = 0.375$. For experiment 2 we used the same device as in [28]. Experiment 3 can be compared to the attack in [8]. Even though the parameters n and $1 - p$ are not the same, we can notice that our method requires less keystream bits when compared to the experiments with $n = 60$. The closest comparison for experiment 4 may be the result with $n = 70$ from [8]; in our experiment, the length of the LFSR is larger and our method requires less keystream bits. It is reported that the method in [8] has error probability of 0.6 for $n = 60$ and 0.4 for $n = 70$; in our case, the error probability is 0.1.

n	d	N	$r = \|\Gamma\|$	# parity-checks mod Γ
40	3	$5 \cdot 10^3$	0	0
			15	0
			25	$2^{8.58}$
40	3	$8 \cdot 10^4$	0	0
			15	$2^{6.58}$
			25	$2^{16.58}$
89	3	2^{28}	32	0

(a) Expected number of parity-checks mod Γ of weight d.

n	d	N	ℓ	r	# relations mod B_r
40	3	$5 \cdot 10^3$	5	0	$2^{9.28}$
			5	15	$2^{24.28}$
			5	25	$2^{34.28}$
40	3	$5 \cdot 10^3$	7	0	$2^{15.28}$
			7	15	$2^{30.28}$
			7	25	$2^{40.28}$
40	3	$8 \cdot 10^4$	5	15	$2^{36.29}$
			7	15	$2^{42.29}$
89	3	2^{28}	5	32	$2^{13.42}$
			7	32	$2^{45.42}$

(b) Expected number of relations mod B_r of weight d.

Table 5: Comparison of the expected number of parity-checks and relations.

	$n = \deg(g)$	d	$1-p$	N	Time complexity
Exp. 1	40	3	0.375	$5 \cdot 10^3$	2^{32}
[28]	40	5	0.375	$1.7 \cdot 10^4$	2^{28}
Exp. 2	40	3	0.375	$5 \cdot 10^3$	2^{35}
[28]	40	5	0.375	$1.7 \cdot 10^4$	2^{28}
Exp. 3	64	3	0.375	$1 \cdot 10^4$	2^{57}
[8]	60	3	0.300	$6.3 \cdot 10^4$	2^{42}
[8]	60	3	0.400	$6 \cdot 10^5$	2^{44}
Exp. 4	80	3	0.375	$1 \cdot 10^4$	2^{71}
[8]	70	3	0.350	$1.12 \cdot 10^6$	2^{44}

Table 6: Comparison of attacks against the filter generator.

9 Application to Grain ciphers

Ciphers in the Grain family are bit oriented synchronous stream ciphers for hardware implementation. Their main components are an LFSR, an NFSR and an output function constructed with a nonlinear Boolean function h and linear terms added to h; see Figure 6.

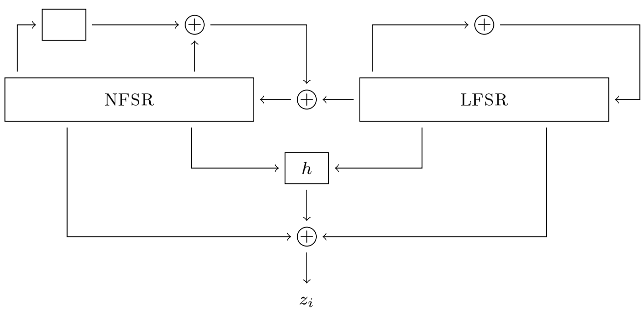

Figure 6: Overview of the components in the Grain family of ciphers.

In this section, we show how to construct a system of equations (4) for the toy Grain-like cipher described in [38] and Grain-v1. We apply the basic multivariate correlation attack from Sections 2.2 and

2.4 to these ciphers. It requires a significantly lower number of keystream bits compared to [38], though with higher time complexity. For the ciphers in this section, the resulting relations presented a high number of right hand sides and therefore required a more efficient strategy for storing their probability distributions. So, we did not apply the test-and-extend algorithm in this case. That is a future direction for this work. For both ciphers, we also find linear combinations of LFSR bits with higher correlations than those indicated in [38].

In [36], Shi et al. applied a FCA against SNOW-V [13] and SNOW-Vi [14] by finding a linear approximation with high correlation. The latter is obtained with the so-called technique of approximation to composite functions together with the aid of an automatic search model based on the SAT/SMT technique (see the paper for details). The technique in [36] may be used with the Grain ciphers and check whether it yields linear approximations with even higher correlation. Additionaly, in a similiar fashion as described below for the Grain ciphers, our new technique may be applied to SNOW-V and SNOW-Vi using the correlations found in [36]. However, we do not follow these directions in this work.

9.1 Grain toy cipher

The cipher consists of LFSR and NFSR of length 24 bits each. The LFSR and NFSR feedback at time t is computed by

$$\begin{aligned} s_{t+24} &= s_t + s_{t+1} + s_{t+2} + s_{t+7}, \\ b_{t+24} &= s_t + b_t + b_{t+5} + b_{t+14} + b_{t+20}b_{t+21} + b_{t+11}b_{t+13}b_{t+15}, \end{aligned}$$

respectively. The keystream bit is computed by

$$z_t = h(s_{t+3}, s_{t+7}, s_{t+15}, s_{t+19}, b_{t+17}) + \sum_{j\in A} b_{t+j}, \tag{19}$$

where $A = \{1, 3, 8\}$ and

$$\begin{aligned} h(x_0, x_1, x_2, x_3, x_4) = x_1 + x_4 + x_0x_3 + x_2x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + \\ x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4 \end{aligned}$$

Assume the N-bit keystream $z_1, \ldots, z_N$ is given. Let $X = (s_1, s_2, \ldots, s_{24})^T$ be the LFSR unknown initial state and

$$\begin{aligned} A_tX = (s_{t+3}, s_{t+7}, s_{t+15}, s_{t+19}, s_{t+8}, s_{t+12}, s_{t+20}, s_{t+24}, s_{t+17}, s_{t+21}, \\ s_{t+29}, s_{t+33}, s_{t+27}, s_{t+31}, s_{t+39}, s_{t+43}, s_{t+1} + s_{t+3} + s_{t+8})^T, \end{aligned}$$

where A_t is a 17×24-matrix. That is, A_tX incorporates 17 linear functions in X. We will construct a multivariate probability distribution on A_tX for a random variable X_t and get a system of equations $A_tX = X_t$, $t = 1, \ldots, N$, as in Section 2.1.

Let $T = \{0, 5, 14, 24\}$ as in [38]. We may have two distributions on A_tX depending on the bit $Z_t = \sum_{i\in T} z_{t+i}$. From the definition of the NFSR feedback

$$\sum_{i\in T, j\in A} b_{t+j+i} = \sum_{j\in A} s_{t+j} + \sum_{j\in A} b_{t+20+j}b_{t+21+j} + b_{t+11+j}b_{t+13+j}b_{t+15+j}.$$

So (19) implies

$$\begin{aligned} Z_t + \sum_{i\in T} h(s_{t+3+i}, s_{t+7+i}, s_{t+15+i}, s_{t+19+i}, b_{t+17+i}) + \sum_{j\in A} s_{t+j} = \\ \sum_{j\in A} b_{t+20+j}b_{t+21+j} + b_{t+11+j}b_{t+13+j}b_{t+15+j} \end{aligned} \tag{20}$$

and

$$s_{t+17} + \sum_{i\in T} b_{t+17+i} = b_{t+37}b_{t+38} + b_{t+28}b_{t+30}b_{t+32}. \tag{21}$$

The distribution of X_t on A_tX is then computed as a uniform distribution conditioned by the relations (20) and (21). The distribution is non-uniform. To be specific, let $a = (a_1, \dots, a_{17})$ be a 17-bit string and we want to compute $\Pr(A_tX = a)$. By $A_tX = a$, (20) and (21) the following 22 bits of

$$u = (Z_t, a_1, \dots, a_{17}, b_{t+17}, b_{t+22}, b_{t+31}, b_{t+41}) \tag{22}$$

uniquely define 3 bits of

$$v = (b_{t+22}, \textstyle\sum_{j\in A} b_{t+20+j}b_{t+21+j} + b_{t+11+j}b_{t+13+j}b_{t+15+j},\ b_{t+37}b_{t+38} + b_{t+28}b_{t+30}b_{t+32}). \tag{23}$$

So $\phi(u) = v$ for a 22-bit to 3-bit mapping ϕ. Each v has the same number 2^{19} of pre-images u under ϕ. The distribution p_v on (23) is precomputed by running over 15 variables involved in the right hand side. This induces a distribution $2^{-19}p_{\phi(u)}$ on (22). Under condition that Z_t is fixed by $\varepsilon = 0$ or 1 we have

$$\mathbf{Pr}(X_t = a|Z_t = \varepsilon) = 2^{-18} \sum_{b_{t+17}, b_{t+22}, b_{t+31}, b_{t+41}, Z_t=\varepsilon} p_{\phi(u)},$$

where the sum is run over all values of $b_{t+17}, b_{t+22}, b_{t+31}, b_{t+41}$ and $Z_t = \varepsilon$. Therefore $A_tX = X_t$, $t = 1, \dots, N$.

We apply the FFT-based method in Section 2.4 to recover X. By Section 2.2, we find the parameters of the limit distributions as

$$\mu_{01} = -11.782815, \quad \sigma_{01}^2 = 0.00137196 \quad \text{and} \quad \mu_{11} = -11.784191, \quad \sigma_{11}^2 = 0.00138229.$$

By the formulae in Section 2.3, for $c = -358\,013.3911$ and $N = 30\,382 \approx 2^{14.89}$, the number of incorrect survivors is < 1 on average and the success probability is $\beta = 0.9999$. The condition (5) is fulfilled. The FFT is used to compute the values of the statistic in (6), thus recovering X. The complexity of the attack is proportional to $2^{17}N + 24 \cdot 2^{24} \approx 2^{31.89}$ operations. The cipher state is 48 bits long. According to [38], with $N = 2^{23.25}$ the whole state may be recovered with the number of operations and memory size of order N.

Let p_0, p_1 be a probability distribution, then $\delta = p_0 - p_1$ is called its correlation. With the FFT we find all linear combinations of the entries of A_tX with non-zero correlations. Table 7 shows absolute values of non-zero correlations δ and the number of linear combinations N_δ with the same δ. The data does not depend on Z_t. It is stated in [38] that there are 1024 linear combinations with highest absolute value of the correlation $2^{-10.41503}$. However that is not correct according to Table 7. There are linear combinations with even a higher correlation. For instance, the absolute value of the correlation of

$$s_{t+7} + s_{t+19} + s_{t+12} + s_{t+24} + s_{t+17} + s_{t+21} + s_{t+31} + s_{t+43} + s_{t+1} + s_{t+3} + s_{t+8} \tag{24}$$

is $2^{-9.83007}$. The reason for the discrepancy is the relation (21) which was ignored in [38]. So we can slightly improve the results of that paper, though we do not follow this direction here.

δ	N_δ
$\frac{9437184}{2^{33}} = 2^{-9.83007...}$	128
$\frac{6291456}{2^{33}} = 2^{-10.41503...}$	768
$\frac{4718592}{2^{33}} = 2^{-10.83007...}$	512
$\frac{3145728}{2^{33}} = 2^{-11.41503...}$	3968
$\frac{1572864}{2^{33}} = 2^{-12.41503...}$	3584

Table 7: Correlations for Grain toy cipher

We verified experimentally the correlation value of (24) as follows. Let s denote (24) with $t = 0$. We randomly chose 2^{30} different initial states for the cipher (i.e, LFSR and NFSR initial states). For each initial state, we computed $Z_0 = z_0 + z_5 + z_{14} + z_{24}$, and when $Z_0 = 0$, we computed s and kept track of the number of times $s = Z_0$. We got that $Z_0 = 0$ occurred $536\,879\,412 \approx 2^{29.000022}$ times and among those, $s = 0$ occurred $268\,737\,466 = 2^{28.001622}$ times. With this, we obtained $2^{-9.816232}$ as experimental correlation.

9.2 Grain-v1

We apply a similar method to Grain-v1. The LFSR and NFSR feedback at time t are computed by

$$\begin{aligned}
s_{t+80} = {} & s_t + s_{t+13} + s_{t+23} + s_{t+38} + s_{t+51} + s_{t+62}, \\
b_{t+80} = {} & s_t + b_{t+62} + b_{t+60} + b_{t+52} + b_{t+45} + b_{t+37} + b_{t+33} + b_{t+28} + b_{t+21} + b_{t+14} + \\
& b_{t+9} + b_t + b_{t+63}b_{t+60} + b_{t+37}b_{t+33} + b_{t+15}b_{t+9} + b_{t+60}b_{t+52}b_{t+45} + \\
& b_{t+33}b_{t+28}b_{t+21} + b_{t+63}b_{t+45}b_{t+28}b_{t+9} + b_{t+60}b_{t+52}b_{t+37}b_{t+33} + \\
& b_{t+63}b_{t+60}b_{t+21}b_{t+15} + b_{t+63}b_{t+60}b_{t+52}b_{t+45}b_{t+37} + b_{t+33}b_{t+28}b_{t+21}b_{t+15}b_{t+9} + \\
& b_{t+52}b_{t+45}b_{t+37}b_{t+33}b_{t+28}b_{t+21}.
\end{aligned}$$

respectively. The keystream bit is computed as

$$z_t = h(s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64}, b_{t+63}) + \sum_{j \in A} b_{t+j}, \tag{25}$$

where $A = \{1, 2, 4, 10, 31, 43, 56\}$ and

$$\begin{aligned}
h(s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64}, b_{t+63}) = {} & s_{t+25} + b_{t+63} + s_{t+3}s_{t+64} + s_{t+46}s_{t+64} + \\
& s_{t+64}b_{t+63} + s_{t+3}s_{t+25}s_{t+46} + s_{t+3}s_{t+46}s_{t+64} + \\
& s_{t+3}s_{t+46}b_{t+63} + s_{t+25}s_{t+46}b_{t+63} + s_{t+46}s_{t+64}b_{t+63}.
\end{aligned}$$

Let $X = (s_1, s_2, \ldots, s_{80})^T$ be the LFSR unknown initial state and

$$\begin{aligned}
A_t X = {} & (s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64}, s_{t+17}, s_{t+39}, s_{t+60}, s_{t+78}, s_{t+24}, s_{t+67}, s_{t+85}, s_{t+31}, \\
& s_{t+53}, s_{t+74}, s_{t+92}, s_{t+40}, s_{t+62}, s_{t+83}, s_{t+101}, s_{t+48}, s_{t+70}, s_{t+91}, s_{t+109}, s_{t+55}, \\
& s_{t+77}, s_{t+98}, s_{t+116}, s_{t+63}, s_{t+106}, s_{t+124}, s_{t+65}, s_{t+87}, s_{t+108}, s_{t+126}, s_{t+105}, \\
& s_{t+144}, s_{t+1} + s_{t+2} + s_{t+4} + s_{t+10} + s_{t+31} + s_{t+43} + s_{t+56}),
\end{aligned}$$

where A_t is a 37×80-matrix. That is $A_t X$ incorporates 37 linear functions in X. We will construct a multivariate probability distribution on $A_t X$ for a random variable X_t and get a system of equations $A_t X = X_t$, $t = 1, \ldots, N$ as in Section 2.1.

Let $T = \{0, 14, 21, 28, 37, 45, 52, 60, 62, 80\}$ as in [38]. We may have two distributions on $A_t X$ depending on $Z_t = \sum_{i \in T} z_{t+i}$. From the definition of the NFSR

$$\sum_{i \in T, j \in A} b_{t+i+j} = \sum_{j \in A} s_{t+j} + \sum_{j \in A} g'(b^{(t+j)}), \tag{26}$$

where

$$\begin{aligned}
g'(b^{(t)}) = {} & b_{t+33} + b_{t+9} + b_{t+63}b_{t+60} + b_{t+37}b_{t+33} + b_{t+15}b_{t+9} + b_{t+60}b_{t+52}b_{t+45} + \\
& b_{t+33}b_{t+28}b_{t+21} + b_{t+63}b_{t+45}b_{t+28}b_{t+9} + b_{t+60}b_{t+52}b_{t+37}b_{t+33} + \\
& b_{t+63}b_{t+60}b_{t+21}b_{t+15} + b_{t+63}b_{t+60}b_{t+52}b_{t+45}b_{t+37} + \\
& b_{t+33}b_{t+28}b_{t+21}b_{t+15}b_{t+9} + b_{t+52}b_{t+45}b_{t+37}b_{t+33}b_{t+28}b_{t+21}.
\end{aligned}$$

So (25) and (26) imply

$$Z_t + \sum_{i \in T} h(s_{t+3+i}, s_{t+25+i}, s_{t+46+i}, s_{t+64+i}, b_{t+63+i}) + \sum_{j \in A} s_{t+j} = \sum_{j \in A} g'(b^{(t+j)}) \tag{27}$$

and

$$s_{t+63} + \sum_{i \in T} b_{t+63+i} = g'(b^{(t+63)}). \tag{28}$$

Let $a = (a_1, \ldots, a_{37})$ be a 37-bit vector, we want to compute $\Pr(A_t X = a)$. By (27) and (28), the following 48 bits of

$$u = (Z_t, a_1, \ldots, a_{37}, b_{t+63}, b_{t+77}, b_{t+84}, b_{t+91}, b_{t+100}, b_{t+108}, b_{t+115}, b_{t+123}, b_{t+125}, b_{t+143}) \tag{29}$$

uniquely define 9 bits of

$$v = (b_{t+77}, b_{t+84}, b_{t+91}, b_{t+100}, b_{t+108}, b_{t+115}, b_{t+123}, \sum_{j\in A} g'(b^{(t+j)}), g'(b^{(t+63)})\,). \tag{30}$$

So $\phi(u) = v$ for a 48-bit to 9-bit mapping ϕ. Each v has 2^{39} pre-images u under ϕ. The distribution p_v on (30) is pre-computed. This induces a distribution $2^{-39}p_{\phi(u)}$ on (29). The last entry in A_tX above incorporates 6 different variables (s_{t+31} appears in position 12 as well). Hence, under the condition that Z_t is fixed by $\epsilon = 0$ or 1, we have

$$\Pr(X_t = a \mid Z_t = \epsilon) = 2^{-38} \sum_{\substack{b_{t+63}, b_{t+77}, b_{t+84}, b_{t+91} \\ b_{t+100}, b_{t+108}, b_{t+115}, b_{t+123} \\ b_{t+125}, b_{t+143}, Z_t=\epsilon}} p_{\phi(u)}.$$

The distribution p_v was pre-computed as follows. The expression for (30) incorporates 64 variables. Some of the variables are fixed by constants, then $\sum_{j\in A} g'(b^{(t+j)})$ and $g'(b^{(t+63)})$ are represented as sums of "independent" polynomials with fewer variables. Independence means that each of the rest variables appears in one polynomial only. The distributions relevant to the independent polynomials are computed separately. Finally, they are combined to get p_v. We computed p_v by fixing $b_{t+38}, b_{t+46}, b_{t+64}, b_{t+65}, b_{t+71}$ and b_{t+91}. The largest computation corresponded with a polynomial in 23 variables.

By Section 2.2, we find the parameters of the limit distributions as

$$\mu_{0,1} = -25.646445680717974846, \quad \sigma_{0,1}^2 = 3.204164923186231 \cdot 10^{-15}$$

and

$$\mu_{1,1} = -25.646445680717978051, \quad \sigma_{1,1}^2 = 3.204164923189462 \cdot 10^{-15}.$$

By the formulae in Section 2.3, for $c = -326687075514236406.749337$ and $N = 2^{53.5}$, the number of incorrect survivors is < 1 on average and the success probability is $\beta = 0.9991$. The condition (5) is fulfilled. The FFT is used to compute the values of the statistic in (6), thus recovering X. The complexity of the attack is proportional to $2^{37}N + 80 \cdot 2^{80} \approx 2^{90.5}$ operations. The internal state of the cipher is 160 bits long. According to [38], with $N = 2^{75.11}$ the whole state may be recovered with time complexity and space complexity of order N.

With the FFT applied to $f(v) = p_v$, we find linear combinations of the entries of A_tX with non-zero correlations. That would require space/memory for 2^{37} elements. Due to the memory limitation, we adopted the following strategy. The Fourier transform on $f : \mathbb{F}_2^n \to \mathbb{R}$ at point u is

$$\widehat{f}(u) = \sum_{x\in\mathbb{F}_2^n} (-1)^{u\cdot x} f(x),$$

where $u \cdot x$ is the dot product of u and x. Let $n = n_1 + n_2$ with $n_1 > 0$ and $n_2 > 0$, then

$$\begin{aligned}\widehat{f}(u) &= \sum_{x_1\in\mathbb{F}_2^{n_1}} \sum_{x_2\in\mathbb{F}_2^{n_2}} (-1)^{(u_1,u_2)\cdot(x_1,x_2)} f(x_1,x_2) = \sum_{x_1\in\mathbb{F}_2^{n_1}} (-1)^{u_1\cdot x_1} \sum_{x_2\in\mathbb{F}_2^{n_2}} (-1)^{u_2\cdot x_2} f(x_1,x_2) \\ &= \sum_{x_1\in\mathbb{F}_2^{n_1}} (-1)^{u_1\cdot x_1} g_{u_2}(x_1) = \widehat{g}_{u_2}(u_1),\end{aligned} \tag{31}$$

where $u_i \in \mathbb{F}_2^{n_i}$ and (u_1, u_2), (x_1, x_2) denote the concatenation of vectors and $g_{u_2}(x_1) = \sum_{x_2\in\mathbb{F}_2^{n_2}} (-1)^{u_2\cdot x_2} f(x_1x_2)$. We can compute the Fourier-Hadamard spectrum of f by using equation (31). For every $u_2 \in \mathbb{F}_2^{n_2}$, we evaluate $g_{u_2}(x_1)$ in all points x_1 by running over x_2, and then we apply the FFT to compute $\widehat{f}(u) = \widehat{g}_{u_2}(u_1)$. The total complexity is $2^{n_2}(2^n + n_1 2^{n_1}) = 2^n(2^{n_2} + n_1)$ operations.

The time complexity of the method above is higher compared to that of the FFT ($n2^n$ operations). However, since we are interested in certain points u (e.g., where $\widehat{f}(u) \neq 0$ or $|\widehat{f}(u)| \geq t$ for a threshold t), we can choose n_1 and n_2 such that the computations of $\widehat{g}_{u_2}$ can be done with the available space/memory and discard the irrelevant data. The computation is parallelisable which is additional to the parallelisation that can be implemented within the FFT for computing $\widehat{g}_{u_2}$.

For this application, we chose $n_2 = 9$ and we parallelised the computation of the 2^9 possible values for u_2. Each computation of the Fast Walsh-Hadamard transform is therefore applied to a vector of length 2^{28}. Since each element was stored on a 64-bit precision floating-point number, the total memory requirement was 2^{34} bits. For each value of u_2, we only kept the values of u_1 such that $|\widehat{f}(u)| >$

2^{-36}, where $u = (u_1, u_2)$. In other words, we only kept the linear combinations of A_tX given by u whose correlation's absolute value is greater than 2^{-36}. The authors in [38] found 442 368 such linear combinations, however, we found 443 264. As in the toy example above, we attribute this discrepancy to the omission of (28) in [38]. There are 171 different correlation values among the 443 264 linear combinations we found. Table 8 shows some of the highest and lowest values. As an example,

$$s_{t+3} + s_{t+25} + s_{t+64} + s_{t+39} + s_{t+60} + s_{t+78} + s_{t+24} + s_{t+85} + s_{t+53} + s_{t+92} +$$
$$s_{t+62} + s_{t+83} + s_{t+101} + s_{t+70} + s_{t+109} + s_{t+77} + s_{t+116} + s_{t+63} + s_{t+106} + s_{t+124} +$$
$$s_{t+87} + s_{t+126} + s_{t+105} + s_{t+144} + s_{t+1} + s_{t+2} + s_{t+4} + s_{t+10} + s_{t+31} + s_{t+43} + s_{t+56}$$

has the highest correlation $2^{-35.46890046}$.

δ	N_δ
$2^{-35.46890046...}$	64
$2^{-35.50019546...}$	64
$2^{-35.54760452...}$	128
$2^{-35.55461504...}$	640
$2^{-35.57682560...}$	64

(a) Highest correlations for Grain-v1.

δ	N_δ
$2^{-35.98275923...}$	128
$2^{-35.98646706...}$	1280
$2^{-35.99121484...}$	256
$2^{-35.99186310...}$	640
$2^{-35.99726377...}$	640

(b) Lowest correlations for Grain-v1.

Table 8: The correlations for Grain-v1.

10 Conclusions

We introduced new methods for cryptanalysis of LFSR-based stream ciphers. The cryptanalysis is presented as a more general problem of finding solutions to systems of linear equations with associated probability distributions on possible right hand sides. We described the multivariate correlation attack and then, the test-and-extend algorithm. The latter has lower time complexity and comprises two stages, precomputation and main computation. In the precomputation stage, we find relations modulo B and compute the probability distributions induced by these relations. The second stage has two variations: tree search and hybrid variant. The first one finds the initial state of the LFSR (in general, candidate solutions to the systems of linear equations) by traversing a tree along with a statistical test to decide which branches to discard. The second variant also traverses a tree, however, the tree search is started at a further level on the tree following the ranking given by the statistic associated to the nodes and computed with the FFT.

We applied the test-and-extend algorithm to a variety of hard instances of the filter generator. In some experiments, we successfully recovered the correct initial state of the LFSR. For the other cases, our cryptanalytic results are theoretical only. In all cases, the hybrid variant outperformed the simple tree search. This new method allows successful recovery of the initial state requiring a lower number of keystream bits compared to other published attacks. We also applied the multivariate correlation method to a toy Grain-like cipher and Grain-v1. Again, we recover the LFSR's initial state for the ciphers using less keystream bits compared to the best known attack. On the other hand, the time complexity to recover the whole cipher state (LFSR and NFSR states) is higher using our method. In the case of Grain-v1, our results are theoretical only. Additionally, for both ciphers, we found linear combinations of LFSR sequence bits with a higher correlation than those reported in [38]. Particularly, the correlations for Grain-v1 were obtained by computing the FFT on a large input vector; we used a simple method to parallelise this computation, which, to the best of our knowledge, has not been reported.

Acknowledgements

Some computations for the various experiments were performed on resources provided by UNINETT Sigma2 - the National Infrastructure for High Performance Computing and Data Storage in Norway.

References

[1] M. Ågren, M. Hell, T. Johansson, and W. Meier. Grain-128a: A New Version of Grain-128 with Optional Authentication. *International Journal of Wireless and Mobile Computing*, 5(1):48–59, 12 2011.

[2] M. Ågren, C. Löndahl, M. Hell, and T. Johansson. A survey on fast correlation attacks. *Cryptography and Communications*, 4(3):173–202, 12 2012.

[3] R. Anderson. Searching for the optimum correlation attack. In B. Preneel, editor, *Fast Software Encryption*, volume 1008 of *Lecture Notes in Computer Science*, pages 137–143, Berlin, Heidelberg, 1995. Springer.

[4] P. Billingsley. *Probability and Measure*. Wiley Series in Probability and Statistics. John Wiley and Sons, New York, 3 edition, 1995.

[5] A. Biryukov and A. Shamir. Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. In T. Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 1–13, Berlin, Heidelberg, 2000. Springer.

[6] A. Canteaut and M. Trabbia. Improved Fast Correlation Attacks Using Parity-Check Equations of Weight 4 and 5. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 573–588, Berlin, Heidelberg, 2000. Springer.

[7] C. Carlet. *Boolean Functions for Cryptography and Coding Theory*. Cambridge University Press, Cambridge, 2021.

[8] V. Chepyzhov, T. Johansson, and B. Smeets. A Simple Algorithm for Fast Correlation Attacks on Stream Ciphers. In G. Goos, J. Hartmanis, J. van Leeuwen, and B. Schneier, editors, *Fast Software Encryption*, volume 1978 of *Lecture Notes in Computer Science*, pages 181–195, Berlin, Heidelberg, 2001. Springer.

[9] P. Chose, A. Joux, and M. Mitton. Fast Correlation Attacks: An Algorithmic Point of View. In L. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 209–221, Berlin, Heidelberg, 2002. Springer.

[10] N. Courtois. Fast Algebraic Attacks on Stream Ciphers with Linear Feedback. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 176–194, Berlin, Heidelberg, 2003. Springer.

[11] N. Courtois and W. Meier. Algebraic Attacks on Stream Ciphers with Linear Feedback. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359, Berlin, Heidelberg, 2003. Springer.

[12] F. Didier. Attacking the Filter Generator by Finding Zero Inputs of the Filtering Function. In K. Srinathan, C. Pandu Rangan, and M. Yung, editors, *Progress in Cryptology – INDOCRYPT 2007*, volume 4859 of *Lecture Notes in Computer Science*, pages 404–413, Berlin, Heidelberg, 2007. Springer.

[13] P. Ekdahl, T. Johansson, A. Maximov, and J. Yang. A new SNOW stream cipher called SNOW-V. *IACR Transactions on Symmetric Cryptology*, 2019(3):1–42, 9 2019.

[14] P. Ekdahl, A. Maximov, T. Johansson, and J. Yang. SNOW-Vi: an extreme performance variant of SNOW-V for lower grade CPUs. In C. Pöpper, M. Vanhoef, L. Batina, and R. Mayrhofer, editors, *WiSec '21*, pages 261–272, New York, 2021. Association for Computing Machinery.

[15] S. Fauskanger and I. Semaev. Separable Statistics and Multidimensional Linear Cryptanalysis. *IACR Transactions on Symmetric Cryptology*, 2018(2):79–110, 6 2018.

[16] J. D. Golić. On the Security of Nonlinear Filter Generators. In D. Gollmann, editor, *Fast Software Encryption*, volume 1039 of *Lecture Notes Computer Science*, pages 173–188, Berlin, Heidelberg, 1996. Springer.

[17] J. D. Golić, A. Clark, and E. Dwason. Generalized Inversion Attack on Nonlinear Filter Generators. *IEEE Transactions on Computers*, 49(10):1100–1109, 10 2000.

[18] J. D. Golić and P. Hawkes. Vectorial Approach to Fast Correlation Attacks. *Designs, Codes and Cryptography*, 35(1):5–19, 4 2005.

[19] J. D. Golić and G. Morgari. Vectorial Fast Correlation Attacks. Cryptology ePrint Archive, Paper 2004/247, 2004.

[20] M. Hell, T. Johansson, A. Maximov, and W. Meier. A Stream Cipher Proposal: Grain-128. In *2006 IEEE International Symposium on Information Theory*, pages 1614–1618, New Jersey, 2006. IEEE.

[21] M. Hell, T. Johansson, A. Maximov, and W. Meier. *The Grain Family of Stream Ciphers*, volume 4986 of *Lecture Notes in Computer Science*, pages 179–190. Springer, Berlin, Heidelberg, 2008.

[22] M. Hell, T. Johansson, and W. Meier. Grain: A Stream Cipher for Constrained Environments. *International Journal of Wireless and Mobile Computing*, 2(1):86–93, 5 2007.

[23] T. Johansson and F. Jönsson. Fast Correlation Attacks Based on Turbo Code Techniques. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 181–197, Berlin, Heidelberg, 1999. Springer.

[24] T. Johansson and F. Jönsson. Improved Fast Correlation Attacks on Stream Ciphers via Convolutional Codes. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 347–362, Berlin, Heidelberg, 1999. Springer.

[25] T. Johansson and F. Jönsson. Fast Correlation Attacks through Reconstruction of Linear Polynomials. In M. Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 300–315, Berlin, Heidelberg, 2000. Springer.

[26] A. Lenstra, H. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 12 1982.

[27] S. Leveiller, J. Boutros, P. Guillot, and G. Zémor. Cryptanalysis of Nonlinear Filter Generators with $\{0,1\}$-Metric Viterbi Decoding. In B. Honary, editor, *Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 402–414, Berlin, Heidelberg, 2001. Springer.

[28] S. Leveiller, G. Zémor, P. Guillot, and J. Boutros. A New Cryptanalytic Attack for PN-generators Filtered by a Boolean Function. In K. Nyberg and H. Heys, editors, *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 232–249, Berlin, Heidelberg, 2003. Springer.

[29] W. Meier. Fast Correlation Attacks: Methods and Countermeasures. In A. Joux, editor, *Fast Software Encryption*, volume 6733 of *Lecture Notes in Computer Science*, pages 55–67. Springer, Berlin, Heidelberg, 2011.

[30] W. Meier and O. Staffelbach. Fast Correlation Attacks on Certain Stream Ciphers. *Journal of Cryptology*, 1(3):159–176, 10 1989.

[31] W. Meier and O. Staffelbach. Nonlinearity Criteria for Cryptographic Functions. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology – EUROCRYPT '89*, volume 434 of *Lecture Notes in Computer Science*, pages 549–562, Berlin, Heidelberg, 1990. Springer.

[32] M. Mihaljević, M. Fossorier, and H. Imai. Fast Correlation Attack Algorithm with List Decoding and an Application. In M. Matsui, editor, *Fast Software Encryption*, volume 2355 of *Lecture Notes in Computer Science*, pages 196–210, Berlin, Heidelberg, 2002. Springer.

[33] H. Molland, J. E. Mathiassen, and T. Helleseth. Improved Fast Correlation Attack Using Low Rate Codes. In K. Paterson, editor, *Cryptography and Coding*, volume 2898 of *Lecture Notes in Computer Science*, pages 67–81, Berlin, Heidelberg, 2003. Springer.

[34] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021.

[35] H. Raddum and I. Semaev. Solving Multiple Right Hand Sides linear equations. *Designs, Codes and Cryptography*, 49(1):147–160, 12 2008.

[36] Z. Shi, C. Jin, J. Zhang, T. Cui, L. Ding, and Y. Jin. A Correlation Attack on Full SNOW-V and SNOW-Vi. In O. Dunkelman and S. Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022*, volume 13277 of *Lecture Notes in Computer Science*, pages 34–56, Berlin, Heidelberg, 2022. Springer.

[37] T. Siegenthaler. Decrypting a Class of Stream Ciphers Using Ciphertext Only. *IEEE Transactions on Computers*, C-49(1):81–85, 1 1985.

[38] Y. Todo, T. Isobe, W. Meier, K. Aoki, and B. Zhang. Fast Correlation Attack Revisited. In H. Shacham and A. Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, volume 10992 of *Lecture Notes in Computer Science*, pages 129–159, Berlin, HeidelbergBerlin, Heidelberg, 2018. Springer.

[39] Z. Zhou, D. Feng, and B. Zhang. Vectorial Decoding Algorithm for Fast Correlation Attack and Its Applications to Stream Cipher Grain-128a. *IACR Transactions on Symmetric Cryptology*, 2022(2):322–350, 6 2022.

A Toy example

A.1 Application of the new method

Here we illustrate the idea of the new method by applying it to a small device. Let the keystream $z_1, \ldots, z_{11} = 1,0,1,0,0,0,0,0,0,1,0$ be produced by a filter generator where $g(x) = x^7 + x^5 + x^2 + x + 1$, $f(x_1, x_2, x_3) = x_1 + x_1x_2 + x_2x_3$ and $(k_1, k_2, k_3) = (0, 2, 5)$. We will find its initial state X. We have

$$M = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \quad \text{and} \quad \Lambda = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Then the matrices $A_i = \Lambda M^{i-1}$, $i = 1, \ldots, 11$ are

$$A_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad A_3 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix},$$

$$A_4 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}, \quad A_5 = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}, \quad A_6 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix},$$

$$A_7 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}, \quad A_8 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}, \quad A_9 = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix},$$

$$A_{10} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}, \quad A_{11} = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

The truth table of f is in Table 9. The distributions $P_{(0)} = (1/4, 1/4, 1/4, 0, 0, 0, 1/4, 0)$ and $P_{(1)} = (0, 0, 0, 1/4, 1/4, 1/4, 0, 1/4)$ correspond to $f(a) = 0$ and $f(a) = 1$, respectively. Hence, the distributions of the random variables X_i are $P_i = P_{(0)}$, for $i = 2, 4, 5, 6, 7, 8, 9, 11$, and $P_i = P_{(1)}$, for $i = 1, 3, 10$. That defines the system (4).

a	$(0,0,0)$	$(0,0,1)$	$(0,1,0)$	$(0,1,1)$	$(1,0,0)$	$(1,0,1)$	$(1,1,0)$	$(1,1,1)$
$f(a)$	0	0	0	1	1	1	0	1

Table 9: Truth table of $f(x_1, x_2, x_3) = x_1 + x_1x_2 + x_2x_3$.

Let the matrices B_r, $r = 1, \ldots, 7$, be obtained by taking the first r rows of

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

We will use short relations of weight $d = 2$. To show the idea of relations, let us consider B_3 and $I = \{1, 2\}$: we have that $\langle A_1, A_2 \rangle \cap \langle B_3 \rangle = \langle T_{3\,I} B_3 \rangle$, where $T_{3\,I} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and therefore $t_{3\,I} = 2$. We define the sets

$$\begin{aligned}
\mathcal{I}_1 &= \{\{5,7\}, \{1,5\}\}, \\
\mathcal{I}_2 &= \{\{1,6\}, \{1,7\}, \{2,5\}\}, \\
\mathcal{I}_3 &= \{\{8,10\}, \{2,11\}, \{3,7\}, \{4,8\}\}, \\
\mathcal{I}_4 &= \{\{2,4\}, \{4,6\}, \{2,6\}, \{2,7\}, \{2,10\}\}, \\
\mathcal{I}_5 &= \{\{5,11\}, \{1,2\}, \{2,9\}, \{7,11\}, \{6,7\}, \{1,11\}, \{4,5\}, \{10,11\}, \{1,8\}, \{5,8\}\}, \\
\mathcal{I}_6 &= \{\{1,10\}, \{3,4\}, \{5,9\}, \{8,11\}, \{4,10\}, \{3,6\}, \{3,9\}\}, \\
\mathcal{I}_7 &= \{\{8,9\}\},
\end{aligned}$$

and compute the probability distribution of those relations. We set the success probability $\beta = 0.9$ and get the thresholds

$$(c_1, \ldots, c_7) = (-2.8344, -8.8069, -15.4057, -17.0976, -39.5219, -28.2609, -4.0000).$$

The applcation of the tree search yielded a unique candidate solution $b = (0, 0, 1, 0, 0, 1, 1)^T$; the tree traversal is depicted in Figure 7. Solving the linear system $B_7 X = b$ yields $X = (1, 0, 1, 1, 0, 1, 0)^T$, which is the correct initial state.

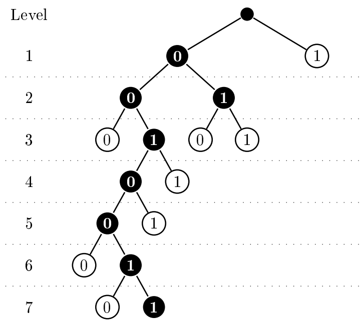

Figure 7: Tree traversal for the toy example. Filled nodes represent the survivor nodes. Non-filled nodes represent rejected nodes whose branch is not traversed.

We also applied the hybrid approach with $r_0 = 3$. Since this is a small example, we used all the available relations modulo B_3 for the given keystream to compute the statistic at that level:

$$\mathcal{I} = \left\{ \begin{array}{l} \{1,2\},\{1,3\},\{1,5\},\{1,6\},\{1,7\},\{1,8\},\{1,9\},\{1,10\},\{1,11\}, \\ \{2,3\},\{2,4\},\{2,5\},\{2,6\},\{2,7\},\{2,9\},\{2,10\},\{2,11\},\{3,4\}, \\ \{3,6\},\{3,7\},\{3,9\},\{3,10\},\{3,11\},\{4,5\},\{4,6\},\{4,8\},\{4,10\}, \\ \{4,11\},\{5,6\},\{5,7\},\{5,8\},\{5,9\},\{5,11\},\{6,7\},\{6,10\},\{7,8\}, \\ \{7,9\},\{7,10\},\{7,11\},\{8,9\},\{8,10\},\{8,11\},\{9,10\},\{9,11\},\{10,11\} \end{array} \right\}.$$

After applying the FFT at level 3, the candidates were sorted as follows according to the value of the statistic:

$$(0,1,0), \quad (1,1,0), \quad (1,1,1), \quad (0,1,1), \quad (0,0,1), \quad (1,0,0), \quad (0,0,0), \quad (1,0,1).$$

Neither of the first four candidates survived at level 4. The fifth candidate is the one corresponding to the correct initial state, which was recovered, and the tree search stopped at this point. The complexity of the hybrid method is defined by the FFT. Due to the size of this example, the hybrid approach is the worst. For bigger instances, however, the hybrid approach yields the best results (e.g., the results in Section 8.2).

A.2 Constructing B_r and relations

We now illustrate the principles in Section 8.1 (i.e., how to obtain the matrices B_r and the sets $\mathcal{I}_r$) for the toy example.

- The matrix $B = B_7$ is constructed by randomly taking vectors from $(1, 1, 0)A_i$, where $x_1 + x_2$ is a good linear approximation to f.
- The 45 weight-2 relations modulo B_3 in the set $\mathcal{I}$ were found using the method in Section 4.1. As explained at the beginning of Section 6, they can be seen as relations modulo B_r for $r = 1, \ldots, 7$. Some $I, J \in \mathcal{I}$ taken modulo B_r may be equivalent for some values of r and not equivalent for others. For example, $I = \{1, 2\}$ and $J = \{1, 8\}$. When $r = 1$ we have

  $$\begin{pmatrix}0\end{pmatrix} B_1 = \begin{pmatrix}0 & 0 & 0\end{pmatrix} A_1 + \begin{pmatrix}0 & 0 & 0\end{pmatrix} A_2, \quad \begin{pmatrix}0\end{pmatrix} B_1 = \begin{pmatrix}0 & 0 & 0\end{pmatrix} A_1 + \begin{pmatrix}0 & 0 & 0\end{pmatrix} A_8.$$

  When $r = 2$ we have

  $$\begin{pmatrix}0 & 1\end{pmatrix} B_2 = \begin{pmatrix}1 & 1 & 0\end{pmatrix} A_1 + \begin{pmatrix}0 & 0 & 0\end{pmatrix} A_2, \quad \begin{pmatrix}0 & 1\end{pmatrix} B_2 = \begin{pmatrix}1 & 1 & 0\end{pmatrix} A_1 + \begin{pmatrix}0 & 0 & 0\end{pmatrix} A_8.$$

Finally, when $r = 3$ we get

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} B_3 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} A_1 + \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} A_2, \quad \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} B_3 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} A_1 + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix} A_8.$$

No indices in I or J are relevant modulo B_1 and the only relevant index of I and J at level 2 is 1, so they are equivalent modulo B_1 and B_2. They are no longer equivalent modulo B_r, $r \geq 3$, since their set of relevant indices are distinct.

- The sets $\mathcal{I}_r$, $r = 1, \ldots, 7$, are disjoint, though not all relations within one set $\mathcal{I}_r$ are pairwise disjoint.