A Simple Algorithm for Fast Correlation Attacks on Stream Ciphers

Vladimor V. Chepyzhov[1], Thomas Johansson[2], and Ben Smeets[3]

[1] Institute for Problems of Information Transmission, Russian Academy of Sciences, Moscow, Russia
[2] Dept. of Information Technology, Lund University, P.O. Box 118, 221 00 Lund, Sweden
[3] Ericsson Mobile Communications, 221 83 Lund, Sweden

Abstract. A new simple algorithm for fast correlation attacks on stream ciphers is presented. The advantages of the new approach are at least two. Firstly, the new algorithm significantly reduces the memory requirements compared with some recent proposals [2,3]. This allows more powerful attacks than previously. Secondly, the simplicity of the algorithm allows us to derive theoretical results. We determine the relation between the number of observed symbols, the correlation probability, and the allowed computational complexity, required for a successful attack. Hence, we can get theoretical estimates on the required computational complexity in cases when simulation is not possible.

Keywords. Stream ciphers, correlation attacks, cryptanalysis.

1 Introduction

A good stream cipher should be resistant against a *known-plaintext attack*. Here the cryptanalyst is given a plaintext and the corresponding ciphertext, and the task is to determine a key k. This is usually equivalent to the problem of finding the key k that produced a given running key $z_1, z_2, \ldots, z_N$.

The problem of cryptanalysis often involves recovering (restoring) the initial states of some linear feedback shift registers, LFSRs. As usual it is assumed that the structure of the key generator is known to the cryptanalyst.

It was noticed by Siegenthaler in [1] that it can happen that the observed output sequence (running key) is correlated to the output of a particular (target) LFSR in the generator. Thus it is reasonable to try to apply a so called divide-and-conquer attack, i.e., try to restore the initial state of the target LFSR independently of the other unknown key bits. In such a setting, one may consider the output of the target LFSR to have passed through an observation channel. The nature of such a channel may vary, but here we model the channel by the Binary (Memoryless) Symmetric Channel, BSC, with some error probability $p < 1/2$.

If $p = 1/2 - \varepsilon$ then usually ε is small. In this setting an LFSR output sequence having some fixed length N can be regarded as a binary linear $[N, l]$-code, where l is the degree of the feedback polynomial of the target LFSR. The number of codewords equals the number of initial states of LFSR, that is 2^l. Thus, the cryptanalyst's problem can be reformulated as a decoding problem of this particular code in the presence of a BSC with strong noise. The problem is how to decode this linear $[N, l]$-code with as low decoding complexity as possible.

Meier and Staffelbach presented in [4] how this could be done in a very efficient way if the feedback polynomial has low weight. Essentially, they made use of iterative decoding techniques. Several minor improvements then followed [5, 6,10]. In [2,3], Johansson and Jönsson presented new ideas involving convolutional codes that improved Meier and Staffelbach's results in the case of a general feedback polynomial.

Although we are influenced by [2,3], this paper proceeds in another direction and uses the following idea. Associate with the target LFSR another binary linear (n_2, k)-code with $k < l$. The k information symbols of this code may coincide with the first k symbols of the initial state of the LFSR we want to recover. The codeword of this second code is considered to have passed through another BSC with a "double" noise level $p_2 = 2p(1-p) > p$, or $p_2 = 1/2 - 2\varepsilon^2$. As will be shown in the paper, if the length of the new code can be chosen at least $n_2 = \lceil k/C(p_2) \rceil$, then the decoding of this code leads to the recovery of the first k symbols in the initial state of the LFSR. Since the new code has dimension k, the decoding complexity is decreased from $O(2^l \times l/C(p))$ to $O(2^k \times k/C(p_2))$.

To make our method work, we need to calculate proper parity checks for construction of the second code. This is done in a precomputation step, and the result is stored on disk. In the decoding step, the symbols of the observed sequence are combined according to the parity checks, and the probability of each codeword in the code is calculated, i.e., ML-decoding. Hence, there are no memory requirements in the decoding part (as opposite to [2,3]), and thus it can be performed very efficiently.

The algorithm is most efficient when the observed output sequence is long (as for all other algorithms for fast correlation attacks), and a theoretical treatment determining the relation between the number of observed symbols, the correlation probability, and the allowed computational complexity, is given. Hence, we can get theoretical estimates of the required computational complexity in cases when simulation is not possible.

In the next section we give the model and problem definition, and in Section 3 we give some arguments on ML-decoding of linear block codes. In Section 4 and 5, the algorithm is described and a theoretical treatment is given. Section 6 contains simulation results.

2 The Cryptanalyst's Problem and Definitions

As most other authors [1]-[6], we use the approach of viewing the problem as a decoding problem. Let the target LFSR have length l and let the set of possible LFSR sequences be denoted by $\mathcal{L}$. Clearly, $|\mathcal{L}| = 2^l$ and for a fixed length N the truncated sequences from $\mathcal{L}$ form a linear $[N, l]$ block code [9], referred to as $\mathcal{C}$. Furthermore, the observed keystream sequence $\mathbf{z} = z_1, z_2, \ldots, z_N$ is regarded

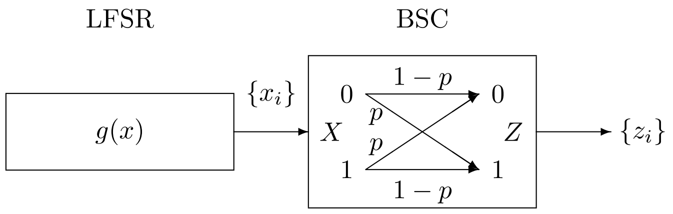

Fig. 1. The correlation attack model for initial state recovery problem.

as the received channel output and the LFSR sequence $\mathbf{x} = x_1, x_2, \ldots, x_N$ is regarded as a codeword from an $[N, l]$ linear block code. Due to the correlation between x_i and z_i, we can describe each z_i as the output of the binary symmetric channel, BSC, when x_i was transmitted. The correlation probability $1-p$, defined by

$$P(x_i = z_i) = 1 - p = 1/2 + \varepsilon,$$

gives p as the crossover probability (error probability) in the BSC. W.l.o.g we can assume $p < 0.5$. This is all shown in Figure 1. The cryptanalyst's problem can now be formulated as follows:

Statement of the problem: Let $p = 1/2 - \varepsilon < 0.5$ and let the feedback polynomial of the LFSR, denoted by $g(D)$, be of degree l. The problem is to restore the linear feedback shift register's initial state $(x_1, x_2, \ldots, x_l)$ from the observed output sequence $\mathbf{z} = (z_1, z_2, \ldots, z_N)$.

In order to derive theoretical results we use the following conjecture. The linear code under consideration is "random" enough to meet the main coding theorem: If the rate $R = k/n$ of a code is less than the capacity $C(p) = 1 - H(p)$ of the BSC then, in the ensemble of random linear (n, k) codes, the decoding error probability approaches zero. Here $H(x)$ is the binary entropy function $H(x) = -x \log_2 x - (1-x)\log_2(1-x)$. This is further described in Section 3.

Siegenthaler in [1] considered an exhaustive search through all the codewords of the above $[N, l]$-code as the decoding procedure. This algorithm is optimal because it is a maximum likelihood (ML) decoding. In [1,6] it is demonstrated that the probability of success is more than 1/2 if $N > n_0$, where n_0 is the *critical length*

$$n_0 = \lceil l/C(p) \rceil .$$

The complexity of this algorithm is about $O\left(2^l \cdot l/C(p)\right)$. The idea of fast correlation attacks is to avoid the factor 2^l and derive algorithms with complexity of order $O(2^{\alpha l})$ with respect to some $\alpha < 1$.

Let us briefly recall some results from coding theory. Since each symbol x_i is a linear combination of the l initial values we see that the set of words

$$(x_1, x_2, \ldots, x_N)$$

forms the linear code $\mathcal{C}$ and we call these words *codewords*. The word

$$(x_1, x_2, \ldots, x_l)$$

is called the information word and the symbols $x_1, x_2, \ldots, x_l$ are called information symbols. We have that

$$\begin{cases} c_l x_1 + c_{l-1}x_2 + \ldots + c_1 x_l + c_0 x_{l+1} = 0 \\ c_l x_2 + c_{l-1}x_3 + \ldots + c_1 x_{l+1} + c_0 x_{l+2} = 0 \\ \ldots \\ c_l x_{N-l} + c_{l-1}x_{N-l+1} + \ldots + c_1 x_{N-1} + c_0 x_N = 0 \end{cases}, \tag{1}$$

where c_i are coefficients of the generator polynomial $g(D) = c_0 + c_1 D + \ldots + c_l D^l$ $(c_0 = c_l = 1)$. We define the $(N-l) \times N$ matrix

$$H = \begin{pmatrix} c_l & c_{l-1} & c_{l-2} & \cdots & c_0 & 0 & \cdots & 0 \\ 0 & c_l & c_{l-1} & \cdots & c_1 & c_0 & \cdots & \vdots \\ 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & c_l & c_{l-1} & \cdots & c_1 & c_0 \end{pmatrix}.$$

Thus, our code consists of all codewords such that $H\,(x_1, x_2, \ldots, x_N)^T = 0$. The matrix H is called the parity check matrix of the LFSR code. It follows from (1) that

$$\begin{cases} x_{l+1} = h_{l+1}^1 x_1 + h_{l+1}^2 x_2 + \ldots + h_{l+1}^l x_l \\ x_{l+2} = h_{l+2}^1 x_1 + h_{l+2}^2 x_2 + \ldots + h_{l+2}^l x_l \\ \vdots \\ x_i = h_i^1 x_1 + h_i^2 x_2 + \ldots + h_i^l x_l \\ \vdots \\ x_N = h_N^1 x_1 + h_N^2 x_2 + \ldots + h_N^l x_l \end{cases}. \tag{2}$$

If we denote $h_i(D) = h_i^1 + h_i^2 D + \ldots + h_i^l D^{l-1}$, then clearly

$$h_i(D) = D^{i-1} \bmod g(D) \text{ for } i = 1, 2, \ldots, N.$$

Therefore the code $\mathcal{C}$ has the $(l \times N)$-generator matrix

$$G = \begin{pmatrix} h_1^1 & h_2^1 & \cdots & h_N^1 \\ h_1^2 & h_2^2 & \cdots & h_N^2 \\ \vdots & & \cdots & \\ h_1^l & h_2^l & \cdots & h_N^l \end{pmatrix}. \tag{3}$$

The observed output sequence, which we call the received word and denote

$$\mathbf{z} = (z_1, z_2, \ldots, z_N),$$

is regarded as a very noisy version of the unknown codeword $(x_1, x_2, \ldots, x_N)$. The initial problem can now be reformulated as a decoding problem:

Restatement of the problem: Let $p < 0.5$ and let the generator polynomial, denoted by $g(D)$, be of degree l. Consider the corresponding $[N, l]$ code $\mathcal{C}$.

The problem is to determine the transmitted codeword from the received word $\mathbf{z}$.

Remark. In the theory of LFSR's and stream ciphers, the typical values for p are closed to 1/2, say, $p = 0.25, 0.3, 0.4, 0.45$. However in the theory of error-correcting codes the typical values of p are much smaller, say, for example, $p = 0.1,\ 0.05$. So we have to cope with a much stronger noise than is usual in coding theory. On the other hand, unlike in error-correcting codes, it is sufficient to demonstrate that our algorithm is able to recover the initial state with some nonzero probability, say 1/2. Thus, a much higher error-rate in decoding can be accepted.

3 ML-Decoding of Linear Codes

In this section we review ML-decoding for binary linear codes and their error-correcting capability. The best selection for the information word is obtained by

Maximum Likelihood decoding (ML-decoding):
Let $\mathcal{C} = \{\mathbf{x}\}$ be an (n, k)-code and let $\bar{\mathbf{x}}$ be the transmitted codeword and let $\mathbf{z}$ be the received word. Now put

$$\hat{\mathbf{x}}_0 \overset{def}{=} \arg\min_{\mathbf{x}\in\mathcal{C}} \operatorname{dist}(\mathbf{x}, \mathbf{z}).$$

Here $\operatorname{dist}(\mathbf{x}, \mathbf{y})$ denotes the Hamming distance between $\mathbf{x}$ and $\mathbf{y}$, i.e., the number of ones in the binary vector $\mathbf{x} + \mathbf{y}$. Notice that the previously described exhaustive search algorithm is exactly ML-decoding. Furthermore, let

$$P_e(p) \overset{def}{=} \Pr(\hat{\mathbf{x}}_0 \neq \bar{\mathbf{x}}) \quad \text{-error probability of ML-decoding.}$$

Note that $P_e(p)$ does not depend on $\bar{\mathbf{x}}$ because the code $\mathcal{C}$ is linear. It is known that ML-decoding has the smallest error probability among all decoding algorithms. So it is optimal. Using a random coding argument, Gallager, Berlekamp and Shannon proved the following.

Theorem 1 ([8]). *Let $C = 1 - H(p)$ denote the capacity of the transmission (observation) channel and let the transmission rate $R = k/n$ satisfy $R < C$. Then*

$$E\left[P_e(p)\right] \le 2^{-\tau(R)n}, \quad \tau(R) = \tau(R, p),$$

where $E\left[P_e(p)\right]$ is the mathematical expectation of the random value $P_e(p)$ in the ensemble of random linear (n, k)-codes. $\tau(R)$ is called the random coding exponent, and $\tau(R) > 0$ for all $R < C$.

Thus we can apply the ML-decoding to the code $\mathcal{C}$ with length N satisfying the inequality $l/N < C(p)$, that is $N > l/C(p)$.

Recall that $p = 1/2 - \varepsilon$. A useful approximation of $C(p)$ is

$$C(p) \approx \varepsilon^2 \cdot 2/\left(\ln 2\right). \tag{4}$$

Simulations show that the critical length $N = n_0 \approx 0.35 \cdot l \cdot \varepsilon^{-2}$ provides the probability of successful decoding close to 1/2, while for $N = 2n_0$ the probability is close to 1 (see [1]). Recall that the complexity of this algorithm has order $O(2^l \cdot l \cdot \varepsilon^{-2})$, so it is very time consuming. Yet we can turn this into something useful as we will see in the next section.

4 Description of a New Fast Correlation Attack

Let $k < l$ be fixed. We shall describe a procedure that recovers the symbols $x_1, x_2, \ldots, x_k$ when observing $z_1, z_2, \ldots, z_N$. In the system (2) we look for pairs of equations such that,

$$h_i^{k+1} = h_j^{k+1},\ h_i^{k+2} = h_j^{k+2}, \ldots, h_i^l = h_j^l,\ 1 \le i \ne j \le N. \tag{5}$$

We find all such pairs of equations. Let the number of such distinct pairs be n_2. Denote the indices of all such pairs as $\{i_1, j_1\}, \{i_2, j_2\}, \ldots, \{i_{n_2}, j_{n_2}\}$.

If i and j satisfies (5), then the sum $x_i + x_j$ is a linear combination of information symbols $x_1, x_2, \ldots, x_k$ only, and is independent of the remaining information symbols $x_{k+1}, x_{k+2}, \ldots, x_l$,

$$x_i + x_j = \left(h_i^1 + h_j^1\right) x_1 + \left(h_i^2 + h_j^2\right) x_2 + \ldots + \left(h_i^k + h_j^k\right) x_k. \tag{6}$$

This means that the sequence

$$(X_1, X_2, \ldots X_{n_2}) = (x_{i_1} + x_{j_1}, x_{i_2} + x_{j_2}, \ldots, x_{i_{n_2}} + x_{j_{n_2}})$$

forms an (n_2, k)-code, referred to as $\mathcal{C}_2$, whose information symbols are $(x_1, x_2, \ldots, x_k)$, i.e., it has dimension k. The generator matrix of the (n_2, k)-code $\mathcal{C}_2$ is

$$G_2 = \begin{pmatrix} h_{i_1}^1 + h_{j_1}^1 & h_{i_2}^1 + h_{j_2}^1 & \cdots & h_{i_{n_2}}^1 + h_{j_{n_2}}^1 \\ h_{i_1}^2 + h_{j_1}^2 & h_{i_2}^2 + h_{j_2}^2 & \cdots & h_{i_{n_2}}^2 + h_{j_{n_2}}^2 \\ \vdots & & \cdots & \\ h_{i_1}^k + h_{j_1}^k & h_{i_2}^k + h_{j_2}^k & \cdots & h_{i_{n_2}}^k + h_{j_{n_2}}^k \end{pmatrix}. \tag{7}$$

We denote

$$Z_1 = z_{i_1} + z_{j_1}, Z_2 = z_{i_2} + z_{j_2}, \ldots, Z_n = z_{i_{n_2}} + z_{j_{n_2}}. \tag{8}$$

Since we observe the output symbols $(z_{i_1}, z_{j_1}, z_{i_2}, z_{j_2}, \ldots, z_{i_{n_2}}, z_{j_{n_2}})$ we can calculate also $(Z_1, Z_2, \ldots, Z_{n_2})$, that is, a word acting as a received word for $\mathcal{C}_2$. If $(e_1, e_2, \ldots, e_N)$ is the noise sequence of the code $\mathcal{C}$, $(e_i = x_i + y_i)$, then clearly the noise sequence $(E_1, E_2, \ldots, E_{n_2})$ for $\mathcal{C}_2$ is

$$E_1 = e_{i_1} + e_{j_1}, E_2 = e_{i_2} + e_{j_2}, \ldots, E_{n_2} = e_{i_{n_2}} + e_{j_{n_2}}.$$

Since our model is the BSC, all the e_i's are independent random binary random variables with error probability p. It is then clear that $E_m = e_{i_m} + e_{j_m}$ for

$i_m \neq j_m$ are also independent binary random variables for all $1 \leq m \leq n_2$ with error probability

$$p_2 = \Pr(e_{i_m} + e_{j_m} = 1) = 2p(1-p) = 1/2 - 2\varepsilon^2.$$

Thus, we have created a new code $\mathcal{C}_2$ with smaller dimension but the BSC over which the codeword is transmitted has a stronger noise p_2, i.e., $p_2 > p$. To restore the symbols $x_1, x_2, \ldots, x_k$ we have to decode the $[n_2, k]$-code C_2 in the BSC with the stronger noise p_2. However, as long as the length of the new code $\mathcal{C}_2$ guarantees unique decoding, this new code will be decoded significantly faster than the LFSR code $\mathcal{C}$.

As before, we apply a simple ML-decoding procedure when decoding $\mathcal{C}_2$. By Theorem 1 we need a code length larger than the critical length n_0, in this case

$$n_2 > k/C(p_2), \tag{9}$$

to get the reliable recovery of the information word. We now are ready to describe the algorithm for the recovery of the initial state.

Algorithm A1.

Data: the length l of target LFSR, and the generator polynomial $g(D)$.

Precomputation.

Fix a computational complexity level by choosing a $k < l$ (for example. $k = l/2$). Construct the generator matrix G (see (3)). Using a sorting algorithm, sort the columns of G with respect to

$$\left(h_i^{k+1}, h_i^{k+2}, \ldots, h_i^{l}\right),\ i = 1, 2, \ldots, N.$$

Find all pairs $\{i, j\}$ that satisfy (5). For each pair, store the indices i, j together with the value of $\left(h_i^1 + h_j^1, h_i^2 + h_j^2, \ldots, h_i^k + h_j^k\right)$. Hence, we have constructed the code $\mathcal{C}_2$ with generator matrix G_2 (see (7)).

Decoding.

Input: The received (observed) vector $(z_1, z_2, \ldots, z_N)$.
Step 1. Compute $(Z_1, Z_2, \ldots, Z_{n_2})$ (see (8)).
Step 2. Decode the code $\mathcal{C}_2$ with the generator matrix (7) using exhaustive search through all the 2^k codewords of $\mathcal{C}_2$, and select the information word $(x_1, x_2, \ldots, x_k)$ with highest probability.

Remark 1. After having restored $(x_1, x_2, \ldots, x_k)$ we need to restore the remaining part of the initial state, $(x_{k+1}, x_{k+2}, \ldots, x_l)$. An obvious way would be to repeat the proposed procedure for some other information bits, say $(x_{k+1}, x_{k+2}, \ldots, x_{2k})$. We can use the same parity checks, since the code is cyclic.

However, with knowledge of the first k information symbols, the remaining problem is much simplified compared to the original problem. Hence we can discard the complexity and the error probability of this step. (E.g. if we restore the 20 first information bits of a length 80 LFSR, we use the obtained values of these 20 first information bits and get a new decoding problem but now only for a length 60 LFSR.)

It is clear that we do not have to restrict ourselves to finding pairs of parity check equations of the form (6), but can consider triples, etc. We describe the general form of the algorithm, that uses sets of t parity check equations whose sums only include the information symbols $(x_1, x_2, \ldots, x_k)$.

Algorithm A2.

Data: the length l of target LFSR and the generator polynomial $g(D)$.

Precomputation.

Fix a computational complexity level by choosing a $k < l$ and a $t \geq 2$. Construct the generator matrix G (see (3)). Sort all columns of G with respect to $\left(h_i^{k+1}, h_i^{k+2}, \ldots, h_i^l\right)$, $i = 1, \ldots, N$.

Then find all sets of t indices $\{i(1), i(2), \ldots, i(t)\}$ that satisfy

$$\sum_{j=1}^{t} h_{i(j)}^m = 0, \text{ for } m = k+1, k+2, \ldots, l.$$

Let the number of such sets be n_t. For each set, store the indices $i(1), i(2), \ldots, i(t)$ together with the value of

$$\left(\sum_{j=1}^{t} h_{i(j)}^1, \sum_{j=1}^{t} h_{i(j)}^2, \ldots, \sum_{j=1}^{t} h_{i(j)}^k\right).$$

Hence, we have constructed an (n_t, k)-code $\mathcal{C}_t$.

Decoding.

Input: The received (observed) vector $(z_1, z_2, \ldots, z_N)$.

Step 1. Compute

$$(Z_1 = \sum_{j=1}^{t} z_{i_1(j)}, Z_2 = \sum_{j=1}^{t} z_{i_2(j)}, \ldots, Z_n = \sum_{j=1}^{t} z_{i_n(j)}).$$

Step 2. Decode the code $\mathcal{C}_t$ using exhaustive search through the all 2^k code words of $\mathcal{C}_t$ and output $(x_1, x_2, \ldots, x_k)$.

Using our model of a BSC and assuming that all the e_i's are the independent random binary values with probability $p = 1/2 - \varepsilon$, it can be shown that

$$E_m = \sum_{j=1}^{t} e_{i_m(j)}$$

are independent random binary variables with error probability

$$p_t = \Pr(\sum_{j=1}^{t} e_{i_m(j)} = 1) = 1/2 - 2^{t-1}\varepsilon^t.$$

We will study this algorithm further in the next section.

5 A Theoretical Analysis of the Proposed Algorithm

We consider theoretical results for algorithm A1 in more detail. We later transfer the results to algorithm A2.

The following lemma shows how many pairs of indices i and j, $1 \leq i, j \leq N$, satisfying the conditions (5), we can expect to find, and how this expectation number n_2 depends on N and k.

Lemma 1 (Birthday paradox). *Let $\xi_1, \xi_2, \ldots, \xi_N$ be random variable that are uniformly distributed on the set of L values. We assume that all this variables are pairwise independent. Denote by ψ the number of pairs $\{i, j\}$ such that $\xi_i = \xi_j$. Then*

$$E(\psi) = \frac{N(N-1)}{2L},$$

where $E(\psi)$ is the mathematical expectation of ψ.

Proof. For every pair $\{i, j\}, i < j$ we denote the random value

$$\pi_{i,j} = \begin{cases} 1 \text{ if } \xi_i = \xi_j \\ 0 \text{ otherwise} \end{cases}$$

The number of these values is $\frac{N(N-1)}{2}$. Since the values ξ_i and ξ_j are independent it follows easily that $\Pr(\pi_{i,j} = 1) = L^{-1}$. Hence $E(\pi_{i,j}) = L^{-1}$. It is clear that

$$\psi = \sum_{\{i,j\}} \pi_{i,j}$$

and therefore

$$E(\psi) = \sum_{\{i,j\}} E(\pi_{i,j}) = \frac{N(N-1)}{2} L^{-1}.$$

□

To apply this lemma we set

$$\xi_i = \left(h_i^{k+1}, h_i^{k+2}, \ldots, h_i^l\right), \ i = l+1, \ldots, l+N.$$

Therefore $L = 2^{l-k}$. From the theory of LFSRs it follows that ξ_i is a good generator of quasi random values with uniform distribution that are pairwise independent.

Corollary 1. *The number n_2 of pairs $\{i, j\}$ that satisfy (5) has expectation*

$$E(n_2) = \frac{N(N-1)}{2} 2^{-(l-k)}. \tag{10}$$

Simulations show that for particular LFSRs the number n_2 is closed to $E(n_2)$ in the formula (10) (see the next section).

Combining (9) and (10) we obtain the length N of the output of the LFSR generator that we have to observe in order to recover the symbols $(x_1, x_2, \ldots, x_k)$ with high probability (close to 1/2).

Theorem 2. *With given k, l, ε, the required length N of the observed sequence* $\mathbf{z}$ *for algorithm A1 to succeed is*

$$N \approx 1/2 \cdot \sqrt{k \cdot (\ln 2)} \cdot \varepsilon^{-2} \cdot 2^{\frac{l-k}{2}}. \tag{11}$$

Proof. (Sketch) By Corollary 1 we can expect the length n_2 of the code $\mathcal{C}_2$ to be roughly $\frac{N(N-1)}{2} 2^{-(l-k)}$. We know from (9) that it is sufficient for n_2 to be at least $k/C(p_2)$ for a high probability of successful decoding. Using the approximation

$$C(p_2) \approx (2\varepsilon^2)^2 \cdot \frac{2}{\ln 2},$$

and the approximation $N(N-1) \approx N^2$ we end up with the expression (11). □

Theorem 2 characterizes the proposed algorithm in a theoretical way. It describes the relation between the number of observed symbols, the correlation probability, and the allowed computational complexity, required for a successful attack. We have described k as the level of computational complexity. Let us now look closer at the exact complexity for a given value of k.

The computational complexity is divided into two parts, one precomputation part and one decoding part. In precomputation, the calculation of all parity checks for the $\mathcal{C}_2$ code is of order $O(N \log N)$. We also need to store the generator matrix G_2 together with the two index positions creating each column in G_2. The storage requirement is at most $n_2(k + 2\log_2 N)$ bits.

The complexity of the decoding step is given as follows.

Corollary 2. *The decoding complexity in algorithm A1 is of the order*

$$2^k \cdot k \cdot \frac{\log_2 2}{8\epsilon^4}.$$

Proof. We run through 2^k codewords with length n_2, where $n_2 \approx k \cdot \frac{\log_2 2}{8\epsilon^4}$.

So by taking a small k, we reduce the 2^k factor but pay in the growth of n_2 and thus also in the length of the observed sequence N.

Important to note is that the decoding part has essentially no memory requirements, since we only keep the most probable information word in memory (however, using some memory can speed up the decoding further). This is in contrast to the algorithms in [2,3], where an extensive amount of memory is used in the decoding part. In fact, it is stated in [3] that the main bottleneck is the memory requirements, and reducing it enables more powerful attacks by the fact that higher computational complexity can be allowed (in [2] they could in some particular cases only use a few hours of computing time, since trying to use more time required too much memory).

Let us now consider algorithm A2. A similar reasoning as above will provide us with the following theorem, similar to Theorem 2.

Theorem 3. *With given k, l, ε, t, the required length N of the observed sequence $\mathbf{z}$ for algorithm A2 to succeed is*

$$N \approx 1/4 \cdot (2kt! \ln 2)^{1/t} \cdot \varepsilon^{-2} \cdot 2^{\frac{l-k}{t}}, \tag{12}$$

assuming $N >> n_t$.

Proof. (Sketch) We can expect the length of the code $\mathcal{C}_t$ to be approximately $\frac{N^t}{t!} 2^{-(l-k)}$. The length should be at least $k/C(p_t)$ for a high probability of successful decoding. Using the approximation

$$C(p_t) \approx (2^{t-1}\varepsilon^t)^2 \cdot \frac{2}{\ln 2},$$

we end up with the expression (12).

The complexity of the decoding step is easily calculated.

Corollary 3. *The decoding complexity for algorithm A2 is of the order*

$$2^k \cdot k \cdot \frac{2 \log_2 2}{(2\epsilon)^{2t}}.$$

Proof. We run through 2^k codewords with length n_t, where $n_t \approx k \cdot \frac{2 \log_2 2}{(2\epsilon)^{2t}}$.

As will be apparent from calculated examples, algorithm A2 is in general more powerful than A1, i.e., using not pairs but sums of three or four columns is more powerful. However, the complexity and storage requirements in the precomputation part is increasing. The calculation of all parity checks for the $\mathcal{C}_t$ code is now of order $O(N^2)$ or higher. The length n_t of G_t in algorithm A2 has increased compared to n_2 in A1. The storage is now in the order of $n_t(k + t \log_2 N)$ bits.

Finally, we must note that for algorithm A2, it can happened that $N >> n_t$ is not true. This will mean that some column positions of G will be used several times when constructing $\mathcal{C}_t$. Hence, the assumption of independence between positions in $\mathcal{C}_t$ is no longer true. The algorithm still performs well but the performance will be slightly worse than stated in Theorem 3 (due to the dependence).

6 Simulations Results

To check the performance as well as the correctness of our assumptions/conjectures, we have made extensive simulations. The simulations were done on a Sun Sparc Ultra-80 computer running under Solaris.

First, we compare the length of the code $\mathcal{C}_2$ with the expected value $E(n_2)$. We consider a random primitive polynomial $g(D)$ of the degree $l = 60$. We set

$k = 20$. The following table reflects the precomputation for algorithm A1.

N	$E(n_2)$	n_2 (we found)	p_0
$3 \cdot 10^7$	409	445	0.25
$5 \cdot 10^7$	1137	1138	0.3
10^8	4547	4567	0.35
$2 \cdot 10^8$	18190	18404	0.4
$3.5 \cdot 10^8$	55707	56142	0.43

Here p_0 is the critical value of the error probability of BSC such that $k/n_2 = C(2p_0(1-p_0))$ when $k = 20$. Notice that the actual values of n_2 are close to the theoretical expected values $E(n_2)$.

The next table shows the probability of decoding error for algorithm A1, depending on the error probability p. This can be compared with the theoretical results in Theorem 2.

(i) $N = 5 \cdot 10^7,\ n_2 = 1138,\ p_0 = 0.3$

p	0.29	**0.3**	0.31	0.33
$P_e(A1)$	0.2	0.3	0.5	0.8

(ii) $N = 3.5 \cdot 10^8,\ n_2 = 56142,\ p_0 = 0.43$

p	0.41	0.42	**0.43**	0.44
$P_e(A1)$	0.01	0.1	0.6	0.9

The following tables show how the parameter k influences the required length N of the observed sequence as well as the time complexity of the decoding algorithm for different p. As before, we choose $l = 60$ and $t = 2$.

(i) $p = 0.3$

k	N	n_2	$P_e(A1)$	Decoding time
20	$5 \cdot 10^7$	1138	0.3	1.5 sec
23	$1.85 \cdot 10^7$	1281	0.4	12 sec
25	$9.7 \cdot 10^6$	1472	0.3	1 min
30	$2 \cdot 10^6$	1800	0.1	30 min

(ii) $p = 0.4$

k	N	n_2	$P_e(A1)$	Decoding time
20	$2 \cdot 10^8$	18404	0.4	20 sec
23	$7.38 \cdot 10^7$	19561	0.5	3 min
25	$3.86 \cdot 10^7$	21329	0.6	14 min
30	$7.45 \cdot 10^6$	25980	0.5	9 h

Notice that all the precomputation, i.e., finding all the corresponding pairs of checks for all k and N in the above table, was completed in approximately 2 hours.

We now consider simulation results for the same polynomial of degree $l = 60$ but with $t = 3$. Here n_3 stands for the number of triples of checks we found for the corresponding parameters k and N. The values of n_3 are close to their theoretical expectations.

(i) $p = 0.3$

k	N	n_3	$P_e(A2)$	Decoding time
20	$3.55 \cdot 10^5$	6670	0.4	8 sec
23	$1.86 \cdot 10^5$	7633	0.4	1 min 20 sec
25	$1.21 \cdot 10^5$	8433	0.3	5 min
28	$6.3 \cdot 10^4$	9466	0.6	1 h

(ii) $p = 0.4$

k	N	n_3	$P_e(A2)$	Decoding time
20	$1.42 \cdot 10^6$	433451	0.5	10 min
23	$7.44 \cdot 10^5$	512108	0.3	1 h 30 min
24	$6 \cdot 10^5$	523734	0.6	4 h
28	$2.5 \cdot 10^5$			

In the last table, the result for $k = 28$ is missing because for $N = 2.5 \cdot 10^5$ the expected number of checks $n_3 \approx 6 \cdot 10^5$ is larger than N. In this case the sums of the corresponding triples of the received sequence are no longer independent and the model of the memoryless BSC does not apply (See the condition of Theorem 3). If we use parameters k for which $n_3(N, k) > N$, we can not expect the performance to follow Theorem 3, although it can still be good.

It is worth also to note that the precomputation time for $t = 3$ grows as N^2. For example, for $k = 20$ and $N = 1.42 \cdot 10^6$ the computer worked 4 days to find all 433451 triples of checks. This means that the precomputation becomes more time consuming when increasing t from $t = 2$ to $t = 3$.

It is interesting to compare the results for $t = 2$ and for $t = 3$. We can see that for equal decoding complexities, the length N can be reduced significantly for $t = 3$. Compare, for example, for $p = 0.3$ the row $k = 23$ ($t = 2$) and the row $k = 20$ ($t = 3$). The reduction factor is 50. The factor is 30 for $p = 0.4$; see the row $k = 25$ ($t = 2$) and the row $k = 20$ ($t = 3$). The price for this reduction is the increase in precomputation complexity.

Finally, we have simulated a set of attacks on LFSRs with $l = 70$ and $p = 0.35$ for different lengths N, and measured the total CPU time for the attacks. The tables below show the result when the parameter k is varying.

(i) $t = 2,\ p = 0.35$

k	N	n_2	$P_e(A1)$	Decoding time
25	$5.48 \cdot 10^8$	4270	0.4	3 min
27	$2.85 \cdot 10^8$	4631	0.5	13 min
28	$2.05 \cdot 10^8$	4944	0.3	30 min
30	$1.06 \cdot 10^8$	5144	0.5	2 h 20 min

(ii) $t = 3,\ p = 0.35$

k	N	n_3	$P_e(A2)$	Decoding time
25	$2.16 \cdot 10^6$	47716	0.5	40 min
26	$1.74 \cdot 10^6$	49860	0.4	1 h 30 min
28	$1.12 \cdot 10^6$	53482	0.4	6 h 20 min

For $t = 2$ the largest precomputation complexity ($k = 25$) is 6 hours. For $t = 3$ the largest precomputation complexity ($k = 25$) is 12 days. Moreover, we were able to reach the value $p = 0.4$ for $t = 2$ observing $4.61{\cdot}10^8$ symbols ($k = 28, n_2 = 24160$). The decoding time is 2 hours. The precomputation time is 10 hours. The same can be done for $t = 3$ observing $4.85{\cdot}10^6$ symbols ($k = 25, E(n_2) = 540414$). The decoding time is 5 hours. The precomputation time will be 2 months.

It should finally be noted that it is easy to get very good estimates on the complexity in other cases by using the derived theoretical results and the simulated values above.

We believe that with a set of parallel PC:s and a few weeks of computation one could use $30 \leq k \leq 35$ and restore LFSRs of length 80-100 using algorithm A2 with $t = 3, 4$.

7 Conclusions

We have demonstrated a new algorithm for fast correlation attacks on stream ciphers. The new algorithm significantly reduces the memory requirements compared with some recent proposals [2,3]. Also, we could derive the relation between the number of observed symbols, the correlation probability, and the allowed computational complexity, required for a successful attack.

Since the algorithm can be very efficiently implemented, the performance (highest error probability for given computational complexity) is better than the algorithms in [2,3]. The performance depends on the computational complexity, and it is not always easy to do a fair comparison between two different algorithms. But the actual simulations that we have done have proved this algorithm to be the fastest.

In conclusion, we think that the simplicity of the proposed algorithm is a great advantage and that this can contribute further progress in the area.

References

1. T. Siegenthaler, "Decrypting a class of stream ciphers using ciphertext only", *IEEE Trans. Comput.*, Vol. C-34, pp. 81-85, 1985.
2. T. Johansson, F. Jönsson, "Improved fast correlation attacks on stream ciphers via convolutional codes", *Proceedings of EUROCRYPT'99*, Springer-Verlag, LNCS 1592, pp. 347-362.
3. T. Johansson, F. Jönsson, "Improved fast correlation attacks on stream ciphers via convolutional codes", *Proceedings of CRYPTO'99*, Springer-Verlag, LNCS 1666, pp. 181-197.

4. W. Meier, and O. Staffelbach, “Fast correlation attacks on certain stream ciphers”, *J. Cryptology*, pp. 159-176, 1989.
5. M. Mihaljevic, and J.Dj. Golić, “A fast iterative algorithm for a shift register initial state reconstruction given the noisy output sequence”, *Proc. Auscrypt’90*, Springer-Verlag, LNCS 453, Eds. J.Seberry and J. Pieprzyk, pp. 165-175, 1990.
6. V. Chepyzhov, and B. Smeets, “On a fast correlation attack on stream ciphers”, *Adv. Crypt.-EUROCRYPT’91*, Brighton, UK, Springer-Verlag, LNCS 547, Ed, D.W. Davies, pp. 176-185, 1991.
7. J.Dj. Golić, “Computation of low-weight parity-check polynomials”, *Electronic Letters*, Vol.32, No. 21, Oct., pp. 1981-1982, 1996.
8. R.G. Gallager, *Information Theory and Reliable Communications*, John Wiley and Sons, Inc. New York, London, Sydney, Toronto, 1968.
9. F. MacWilliams, N. Sloane, *The theory of error correcting codes*, North Holland, 1977.
10. W. Penzhorn, “Correlation attacks on stream ciphers: Computing low weight parity checks based on error correcting codes”, FSE’96, Springer-Verlag, LNCS 1039, pp. 159–172.