Improved Fast Correlation Attacks Using Parity-Check Equations of Weight 4 and 5

Anne Canteaut[1] and Michaël Trabbia[1,2]

[1] INRIA projet CODES
B.P. 105
78153 Le Chesnay Cedex - France
Anne.Canteaut@inria.fr
[2] Ecole Polytechnique
91128 Palaiseau Cedex - France
michael.trabbia@enst.fr

Abstract. This paper describes new techniques for fast correlation attacks, based on Gallager iterative decoding algorithm using parity-check equations of weight greater than 3. These attacks can be applied to any key-stream generator based on LFSRs and it does not require that the involved feedback polynomial have a low weight. We give a theoretical analysis of all fast correlation attacks, which shows that our algorithm with parity-check equations of weight 4 or 5 is usually much more efficient than correlation attacks based on convolutional codes or on turbo codes. Simulation results confirm the validity of this comparison. In this context, we also point out the major role played by the nonlinearity of the Boolean function used in a combination generator.

1 Introduction

Stream ciphers form an important class of secret-key encryption schemes. They are widely used in applications since they present many advantages: they are usually faster than common block ciphers and they have less complex hardware circuitry. Moreover, their use is particularly well-suited when errors may occur during the transmission because they avoid error propagation. In a binary additive stream cipher the ciphertext is obtained by adding bitwise the plaintext to a pseudo-random sequence s, called the *running-key* (or the *key stream*). The running-key is produced by a pseudo-random generator whose initialization is the secret key shared by the users. Most attacks on such ciphers therefore consist in recovering the initialization of the pseudo-random generator from the knowledge of a few ciphertext bits (or of some bits of the running-key in known-plaintext attacks).

Linear feedback shift registers (LFSRs) are the basic components of most key-stream generators since they are appropriate to hardware implementations, produce sequences with good statistical properties and can be easily analyzed. Different classes of key-stream generators can be distinguished depending on the techniques used for combining the constituent LFSRs [16]: combination generators, filter generators, clock-controlled generators.... In all these systems, the secret key usually consists of the initial states of the constituent LFSRs. The secret key has then $\sum_{i=1}^{n} L_i$ bits, where L_i denotes the length of the i-th LFSR and n is the number of involved LFSRs. Any key-stream generator based on LFSRs is vulnerable to *correlation attacks*. These cryptanalytic techniques introduced by Siegenthaler [20] are "divide-and-conquer" methods: they exploit the existence of a statistical dependence between the running-key and the output of one constituent LFSR for recovering the initialization of each LFSR separately. The secret key can then be recovered with only $\sum_{i=1}^{n} 2^{L_i}$ tests.

A classical method for generating a running-key is to combine n LFSRs by a nonlinear Boolean function f. Such a combination generator is depicted in Figure 1. For combination generators, the original correlation attack presented by Siegenthaler can be prevented by using a *correlation-immune* combining function [19]. In this case, the running-key is statistically independent of the output of each constituent LFSR; any correlation attack should then consider several LFSRs together. More generally, a correlation attack on a set of k LFSRs, namely LFSR $i_1, \ldots,$ LFSR i_k, exploits the existence of a correlation between the running-key s and the output σ of a smaller combination generator, which consists of the k involved LFSRs combined by a Boolean function g with k variables (see Fig. 2). Since $Pr[s_n \neq \sigma_n] = Pr[f(X_1, \ldots, X_n) \neq g(X_{i_1}, \ldots, X_{i_k})] = p_g$, this attack only succeeds when $p_g < 0.5$. The number k of involved LFSRs should then be strictly greater than the correlation immunity order t of the combining function f. This cryptanalysis therefore requires that all $2^{\sum_{j=1}^{t+1} L_{i_j}}$ initial states be examined; it becomes infeasible when the correlation-immunity order t of the combining function is high.

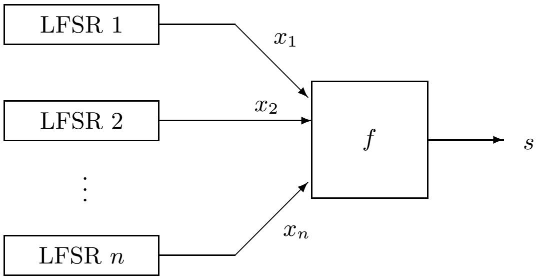

Fig. 1. Combination generator

The fast correlation attack proposed by Meier and Staffelbach [9, 10] relies on the same principle but it avoids examining all possible initializations of $(t+1)$ LFSRs together. Let us again consider the sequence σ produced by LFSR $i_1, \ldots,$

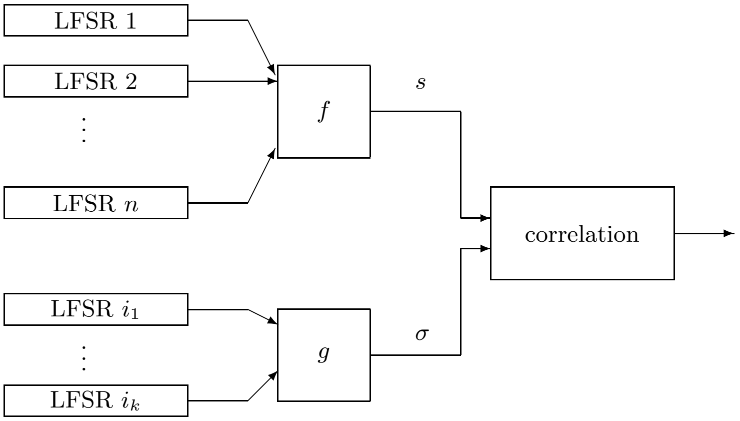

Fig. 2. Correlation attack involving k constituent LFSRs

LFSR i_k combined by g. The sequence σ obviously corresponds to the output of a unique LFSR of length L; the length and the feedback polynomial P of this LFSR can be derived from the feedback polynomials of the constituent LFSRs. Note that $L \leq g(L_{i_1}, \ldots, L_{i_k})$ where the function g is evaluated over integers. Equality notably holds when the feedback polynomials of the involved LFSRs are primitive and when their degrees are coprime [17]. Any subsequence of length N of σ is then a codeword of a linear code $\mathcal{C}$ of length N and dimension L defined by the feedback polynomial P. The running-key subsequence $(s_n)_{n<N}$ can then be seen as the result of the transmission of $(\sigma_n)_{n<N}$ through the binary symmetric channel with error probability (or crossover probability) $p = Pr[s_n \neq \sigma_n]$. The attack therefore aims at recovering L consecutive bits of σ (i.e., the initial state of the equivalent LFSR) from the knowledge of N bits of s. This can be done by decoding $(s_n)_{n<N}$ relatively to $\mathcal{C}$.

From the attacker's point of view, the main problem is to make the cryptanalysis feasible even if a small number N of bits of the running-key (or of the

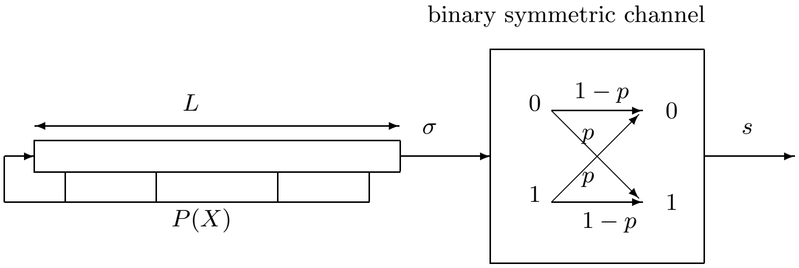

Fig. 3. Model for a fast correlation attack

ciphertext) is known. Shannon's channel coding theorem [18] gives a theoretical lower bound on N depending on the error-probability p: $N \geq LC(p)$, where $C(p)$ is the capacity of the binary symmetric channel with error-probability p, i.e., $C(p) = 1 + p\log_2(p) + (1-p)\log_2(1-p)$. Unfortunately no efficient general decoding algorithm is known for achieving the channel capacity. This means that practical correlation attacks require that the known running-key sequence be much longer than this theoretical bound. Any improvement of fast correlation attacks then consists in finding an efficient decoding procedure for the code $\mathcal{C}$, when N is as close as possible to Shannon's limit.

Meier and Staffelbach attack [10] uses the iterative decoding process due to Gallager for low-density parity-check codes [3]. Any polynomial P^{2^i} actually provides a parity-check equation for $\mathcal{C}$ as far as its degree is less than N. It follows that the received sequence $(s_n)_{n<N}$ can be decoded with Gallager algorithm when the feedback polynomial P has a low weight and when the error probability p is not too high. Several minor improvements of this original attack were proposed in [21, 12, 1, 13, 15] but these papers did not introduce any important modification of the basic underlying concepts. Johansson and Jönsson recently proposed two new techniques for fast correlation attacks: the main idea is to derive from $(s_n)_{n<N}$ a sequence which can be seen as a corrupted version of a word of a convolutional code [6] or of a turbo code [7]. These new attacks increase the highest achievable error probability p for given values of L and N (L is the length of the LFSR generating σ and N is the number of known bits of the running-key). Moreover, they do not require that the feedback polynomial P have a low weight. We here show that Gallager iterative decoding algorithm with parity-check equations of weight 4 or 5 is usually much more efficient than all previous attacks: it successfully decodes very high error probabilities with a feasible time and memory complexity, and it does not require that the feedback polynomial P have a low weight. As an example, for a LFSR of length $L = 40$ and an error-probability $p = 0.3$, the best previously known attack [7] requires the knowledge of $N = 40,000$ bits of s whereas our algorithm with parity-check equations of weight 5 is successful with only $9,770$ bits.

The paper is organized as follows. Section 2 focuses on the particular case of combination generators. Here, we prove that the lowest possible Hamming distance between a fixed t-resilient function and any Boolean function with $(t+1)$ variables is achieved by an affine function. It follows that the nonlinearity of the combining function plays a major role in the resistance of a combination generator to correlation attacks. The rest of the paper presents a new general method for fast correlation attacks which can be applied to any type of keystream generators based on LFSRs. The preprocessing step and the decoding step of the algorithm are respectively described in Section 3 and 4. Section 5 gives a theoretical analysis of the recent attacks proposed by Johansson and Jönsson. Most notably, we point out that our attack using parity-check equations of weight 4 or 5 has better performance than the attacks based on convolutional codes or on turbo codes. Section 6 finally presents some simulation results which confirm the validity of the previous comparison.

2 Approximation of a t-resilient function by a function with $t+1$ variables

This section is devoted to the special case of combination generators. It focuses on the choice of the Boolean function g which is used for combining the k LFSRs involved in the correlation attack (see Fig. 2). The attack will be even more efficient that the correlation between the running-key s and the sequence σ is high. This equivalently means that $Pr[s_n \neq \sigma_n]$ should be as small as possible. The Boolean function g with k variables should then minimize $p_g = Pr[f(X_1, \ldots, X_n) \neq g(X_{i_1}, \ldots, X_{i_k})]$. Moreover, since the length of the equivalent LFSR considered in a fast correlation attack is usually given by $L = g(L_{i_1}, \ldots, L_{i_k})$, it is obviously required that the degree of g be not too high.

We first recall some basic properties of Boolean functions (see e.g. [8], [11] and [2] for details). In the following, $\mathcal{F}_n$ denotes the set of all Boolean functions with n variables, i.e., the set of all functions from $\mathbf{F}_2^n$ into $\mathbf{F}_2$. A Boolean function is balanced if its output is uniformly distributed; balancedness is then an obvious requirement for combining functions. A Boolean function $f \in \mathcal{F}_n$ is t-th order correlation-immune if the probability distribution of its output is unaltered when any t input variables are fixed [19]. Balanced t-th order correlation-immune functions are called t-resilient. Note that a t-th order correlation-immune function is k-th order correlation-immune for any $k \leq t$. From now on, the correlation-immunity order of a function f then refers to the highest integer t such that f is t-th order correlation-immune. The *Walsh transform* of a Boolean function $f \in \mathcal{F}_n$ is the Fourier transform of the corresponding sign function $\chi_f(x) = (-1)^{f(x)}$:

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

where $x \cdot y$ denotes the usual dot product between two n-bit vectors x and y. The Walsh coefficient $\widehat{\chi_f}(u)$ estimates the Hamming distance between f and the affine function $u \cdot x + \varepsilon$, $\varepsilon \in \mathbf{F}_2$:

$$Pr[f(X_1, \ldots, X_n) \neq u \cdot X + \varepsilon] = \frac{1}{2} - \frac{(-1)^\varepsilon}{2^{n+1}} \widehat{\chi_f}(u) \; .$$

The nonlinearity of f, $\mathcal{NL}(f)$, corresponds to its Hamming distance to the set of affine functions:

$$\mathcal{NL}(f) = 2^{n-1} - \frac{1}{2} \max_{u \in \mathbf{F}_2^n} |\widehat{\chi_f}(u)| \; .$$

We now consider the combination generator depicted in Figure 1 and we assume that the combining function f is t-resilient. The running-key produced by the combination generator is then independent of any set of t constituent LFSRs. The smallest number of LFSRs involved in a correlation attack is therefore $t+1$. We now prove that, in this case, the Boolean function g with $(t+1)$ variables which provides the best approximation to f (i.e., which minimizes p_g) is an affine function.

Theorem 1. *Let f be a t-resilient function with n variables and let T be a subset of $\{1, \dots, n\}$ of cardinality $(t+1)$, $T = \{i_1, \dots, i_{t+1}\}$. The lowest possible value over all $g \in \mathcal{F}_{t+1}$ of*

$$p_g = Pr[f(X_1, \dots, X_n) \neq g(X_{i_1}, \dots, X_{i_{t+1}})]$$

is achieved by the affine function

$$g(x_{i_1}, \dots, x_{i_{t+1}}) = \sum_{i \in T} x_i + \varepsilon$$

with $\varepsilon = 0$ if $\widehat{\chi_f}(1_T) > 0$ and $\varepsilon = 1$ otherwise, where 1_T denotes the n-bit vector whose i-th component equals 1 if and only if $i \in T$.

Moreover, we have

$$\min_{g \in \mathcal{F}_{t+1}} p_g = \frac{1}{2} - \frac{1}{2^{n+1}} |\widehat{\chi_f}(1_T)| \ .$$

Proof: For any vector $x \in \mathbf{F}_2^n$, $x = (y, z)$ refers to the decomposition of x with respect to T, i.e., y is the $(t+1)$-bit vector composed of all $x_i, i \in T$. Let $p_T(y)$, $y \in \mathbf{F}_2^{t+1}$, denote the probability $p_T(y) = Pr[f(Y, Z) = 1 | Y = y]$. For any $g \in \mathcal{F}_{t+1}$ we have

$$\begin{aligned} p_g &= Pr[f(Y, Z) \neq g(Y)] \\ &= \sum_{y \in g^{-1}(0)} Pr[f(Y, Z) = 1 | Y = y] + \sum_{y \in g^{-1}(1)} Pr[f(Y, Z) = 0 | Y = y] \\ &= \sum_{y \in g^{-1}(0)} p_T(y) + \sum_{y \in g^{-1}(1)} (1 - p_T(y)) \ . \end{aligned}$$

It follows that p_g is minimal if and only if

$$g(x) = \begin{cases} 0 & \text{if } p_T(x) < 1/2, \\ 1 & \text{if } p_T(x) > 1/2 \ . \end{cases} \tag{1}$$

Note that the value of $g(x)$ can be arbitrarily chosen when $p_T(x) = \frac{1}{2}$. For any $j \in T$, e_j denotes the $(t+1)$-bit vector whose all coordinates are zero except the j-th one. For any $y \in \mathbf{F}_2^{t+1}$ and any $j \in T$, we have

$$\begin{aligned} p_T(y) + p_T(y + e_j) &= Pr[f(Y, Z) = 1 | Y = y] + Pr[f(Y, Z) = 1 | Y = y + e_j] \\ &= 2\,(Pr[f(Y, Z) = 1 | \forall i \in T, Y_i = y_i] Pr[Y_j = y_j] + \\ &\quad Pr[f(Y, Z) = 1 | \forall i \in T \setminus \{j\}, Y_i = y_i, Y_j \neq y_j] Pr[Y_j \neq y_j]) \\ &= 2 Pr[f(Y, Z) = 1 | \forall i \in T \setminus \{j\}, Y_i = y_i] = 1 \end{aligned}$$

where the last equality comes from the fact that f is t-resilient and that the set $T \setminus \{j\}$ has cardinality t. Let $g \in \mathcal{F}_{t+1}$ be such that p_g is minimal. Since $p_T(x) + p_T(x + e_j) = 1$ for any $x \in \mathbf{F}_2^{t+1}$ and for any $j \in T$, Condition (1) implies that

$$g(x) + g(x + e_j) \equiv 1 \bmod 2$$

when $p_T(x) \neq \frac{1}{2}$. Moreover, we can assume that this relation is satisfied for any $x \in \mathbf{F}_2^{t+1}$, because the value of $g(x)$ can be arbitrarily chosen when $p_T(x) = \frac{1}{2}$. It follows that, for any $x \in \mathbf{F}_2^{t+1}$,

$$g(x) = g(0) + \sum_{i \in T} x_i \ .$$

Since g is an affine function, p_g is given by

$$p_g = \frac{1}{2} - \frac{(-1)^{g(0)}}{2^{n+1}} \widehat{\chi_f}(1_T) \ .$$

This probability is then minimized when $(-1)^{g(0)}$ and $\widehat{\chi_f}(1_T)$ have the same sign. □

It follows that, in a fast correlation attack involving $(t+1)$ LFSRs, the same combining function g minimizes both the error probability p_g and the length of the LFSR generating σ. In this context, the feedback polynomial P of this equivalent LFSR is the least common multiple of the feedback polynomials P_i of the considered LFSRs [22]. Note that we generally have $P = \prod_{i \in T} P_i$ since all these feedback polynomials are usually primitive. The running-key s can be seen as the result of the transmission of the sequence σ generated by this LFSR through the binary symmetric channel with error probability

$$p = \frac{1}{2} - \frac{1}{2^{n+1}} |\widehat{\chi_f}(1_T)| \ .$$

A similar cryptanalytic method applies to a ciphertext-only attack. In this case, we make use of the redundancy of the plaintext sequence m, i.e., $Pr[m_n = 0] = p_0 > 1/2$. The attack now considers that the ciphertext sequence c results of the transmission of σ through the binary symmetric channel with error-probability

$$p = Pr[c_n \neq \sigma_n] = \frac{1}{2} - \frac{(2p_0 - 1)}{2^{n+1}} |\widehat{\chi_f}(1_T)| \ .$$

Theorem 1 points out the importance of the nonlinearity of the combining function f: any known-plaintext correlation attack on $(t+1)$ LFSRs should decode an error probability $p \geq \mathcal{NL}(f)/2^n$. The use of highly nonlinear combining function may then prevent this attack. In this case, an acceptable error probability can only be obtained when $(t+2)$ constituent LFSRs are involved and when the degree of g is at least 2. But this dramatically increases the length of the equivalent LFSR and it makes any correlation attack infeasible.

3 Generating parity-check equations

We now come back to the general cryptanalysis of any key-stream generator based on LFSRs. A fast correlation attack aims at recovering L consecutive bits of σ from the knowledge of N bits of s (see Fig. 3). We use that any N-bit subsequence of σ is a codeword of a linear code $\mathcal{C}$ of length N and dimension L defined by the feedback polynomial P. The preprocessing step of the attack then consists in generating some parity-check equations for $\mathcal{C}$ (i.e., linear relations involving some bits of $(\sigma_n)_{n<N}$) in such a way that they provide an efficient decoding procedure. Here, we use a fast decoding algorithm due to Gallager [3] for low-density parity-check codes. In this context, the preprocessing step consists in searching for all linear equations involving d bits of the sequence $(\sigma_n)_{n<N}$:

$$\sigma_n + \sum_{j=1}^{d-1} \sigma_{i_j} = 0 \ .$$

These equations exactly correspond to the polynomials $Q(X)P(X)$ of weight d and of degree at most N, where P is the feedback polynomial of the LFSR generating σ. The cyclic structure of LFSR sequences implies that the set of all parity-check equations of weight d involving σ_i does not depend on i. It is therefore sufficient to find all polynomials $Q(X)P(X)$ of weight d whose constant term equals 1. These polynomials can be found with the following algorithm:

- Compute all residues $q_i(X) = X^i \bmod P(X)$ for $1 \leq i < N$ and store their values in a table T defined by

$$\forall 0 \leq a < 2^L, \ T[a] = \{i, q_i(X) = a\} \ .$$

- For each set of $d-2$ elements of $\{1, \cdots, N-1\}$
  compute $A = 1 + q_{i_1}(X) + \ldots + q_{i_{d-2}}(X)$
  for any $j \in T[A]$, $1 + X^{i_1} + \ldots + X^{i_{d-2}} + X^j$ is a multiple of P of weight d.

The number of operations required by this algorithm is then roughly

$$\binom{N-1}{d-2} \sim \frac{N^{d-2}}{(d-2)!} \ .$$

We can also use an algorithm based on a “birthday technique” as suggested in [10, Section 5]. This consists in storing in a table the values of all linear combinations of $\lfloor \frac{d-1}{2} \rfloor$ residues $q_i(X)$. The complexity of this algorithm is only $\binom{N}{\lceil \frac{d-1}{2} \rceil}$ but it requires $L\binom{N}{\lfloor \frac{d-1}{2} \rfloor}$ bits of memory. For $d > 4$ the choice of the algorithm used in the preprocessing step then highly depends on the available memory amount. Similar techniques for finding low-weight parity-check equations are presented in [14].

We now want to estimate the number of such parity-check equations of weight d. Let us first assume that P is a primitive polynomial in $\mathbf{F}_2[X]$ of degree L. Then the number $m(d)$ of polynomials $Q(X) = 1 + \sum_{i=1}^{N} q_i X^i$ of weight d such that P divides Q is approximatively

$$m(d) \simeq \frac{N^{d-1}}{(d-1)!2^L} \ . \qquad (2)$$

This approximation is motivated as follows: when d is small, the number of multiples of P of weight d and of degree at most $2^L - 1$ can be approximated [8, p. 129] by

$$A = \frac{2^{(d-1)L}}{d!} \ .$$

Let us now assume that the probability p_d that P divides a polynomial of weight d is uniform. We then deduce that

$$p_d = \frac{A}{\binom{2^L-1}{d}} \simeq \frac{1}{2^L} \ .$$

We similarly obtain that

$$m(d) = p_d \binom{N-1}{d-1} \simeq \frac{N^{d-1}}{(d-1)!2^L} \ .$$

Simulations for $d \leq 6$ show the accuracy of this approximation when N is not too small. Moreover its validity does not depend on the weight of P. As an example the following table compares our approximation with the exact values of $m(3)$ for two polynomials of degree 17: $P_1(X) = 1 + X^3 + X^{17}$ and $P_2(X) = 1+X^2+X^4+X^5+X^6+X^8+X^9+X^{10}+X^{11}+X^{13}+X^{14}+X^{15}+X^{17}$.

N	3000	4000	5000	6000	7000	8000
$m_1(3)$	38	61	95	131	183	238
$m_2(3)$	36	67	95	127	185	243
approximation	34	61	95	137	187	244

Since this approximation is also accurate when P is a product of primitive polynomials, we will now use Formula (2) as an approximation of the number of parity-check equations of weight d involving the i-th bit of $(\sigma_n)_{n<N}$. For the polynomial of degree 40 considered in [6], $P(X) = 1 + X + X^3 + X^5 + X^9 + X^{11} + X^{12} + X^{17} + X^{19} + X^{21} + X^{25} + X^{27} + X^{29} + X^{32} + X^{33} + X^{38} + X^{40}$, we obtain 9607 parity-check equations of weight 4 for $N = 400,000$ and 400 parity-check equations of weight 5 for $N = 10,000$. For these values of N, Formula (2) gives $m(4) = 9701$ and $m(5) = 379$.

4 Decoding procedure

Using the previous parity-check equations we recover $(\sigma_n)_{n<N}$ from $(s_n)_{n<N}$ using Gallager soft-input/soft-output decoding algorithm [3, 4]. It relies on the evaluation, for all $0 \leq i < N$, of the probability that σ_i equals 1 conditional on the known sequence $(s_n)_{n<N}$ and on the event S that all parity-check equations involving σ_i are satisfied. As usual in soft decoding algorithms, all probabilities are expressed in terms of log-likelihood ratios: the log-likelihood ratio of a binary random variable X, $L(X)$, is defined as

$$L(X) = \log \frac{Pr[X = 0]}{Pr[X = 1]} \ .$$

The sign of $L(X)$ corresponds to a hard decision on X ($\text{sign}(L(X)) = (-1)^X$); its magnitude $|L(X)|$ is the reliability of this decision. Here, we have that

$$L(\sigma_i|(s_n)_{n<N}, S) = L(\sigma_i|(s_n)_{n<N}) + L(S|\sigma_i, (s_n)_{n<N}) \ .$$

The second term of the right hand member of this equation can be evaluated with the following approximation (see e.g. [5])

$$L(\sum_{i=1}^{k} X_i) = \left(\prod_{i}^{k} \text{sign}(L(X_i))\right) \min_{1\leq i\leq k} |L(X_i)| \ .$$

The decoding procedure is then as follows:

```
– Initialization: for all $i$ from 0 to $N-1$, $L[i] = \log\frac{1-p}{p}$ .
– Until convergence, repeat:
    For all $i$ from 0 to $N-1$
        $L'[i] = (-1)^{s_i} L[i]$.
        for any parity-check equation involving $\sigma_i$, written as $\sigma_i + \sum_{j\in J} \sigma_j = 0$,
            $L'[i] \leftarrow L'[i] + \left(\prod_{j\in J} (-1)^{s_j}\right) \min_{j\in J} L[j]$ .
    For all $i$ from 0 to $N-1$, $s_i \leftarrow \text{sign}(L'[i])$ and $L[i] \leftarrow |L'[i]|$.
```

The number of parity-check equations required for convergence of this decoding procedure highly depends on their weight d. Figures 4 and 5 present simulations results for $L = 21$ ($P(X) = 1 + X^2 + X^3 + X^5 + X^{10} + X^{11} + X^{12} + X^{14} + X^{21}$). Figure 5 clearly shows that the performance of the attack increases with the weight of the parity-check equations. For $p = 0.4$, the attack requires the knowledge of 16800 bits of s for $d = 3$, 2200 bits for $d = 4$ and 1100 bits for $d = 5$.

Simulations actually provide the following approximation of the minimum value of $m(d)$ for convergence (see Fig. 4):

$$m(d) \geq \frac{K_d}{C_{d-2}(p)} \tag{3}$$

where $C_{d-2}(p)$ is the capacity of the binary symmetric channel with error-probability $p_{d-2} = \frac{1}{2}(1 - (1-2p)^{d-2})$, *i.e.* $C_{d-2}(p) = 1 + p_{d-2}\log_2(p_{d-2}) + (1-p_{d-2})\log_2(1-p_{d-2})$. $K_d \simeq 1$ if $d \geq 4$ and $K_3 \simeq 2$. Combining (2) and (3) we obtain that the correlation attack with parity-check equations of weight d is successful if the number of known bits of s is at least

$$N = 2^{\alpha_d(p) + \frac{L}{d-1}} \text{ with } \alpha_d(p) = \frac{1}{d-1}\log_2\left[(d-1)!\frac{K_d}{C_{d-2}(p)}\right] \ . \tag{4}$$

This formula points out that the influence of L decreases when we use higher-weight parity-check equations. When $m(d)$ satisfies (3) the decoding procedure

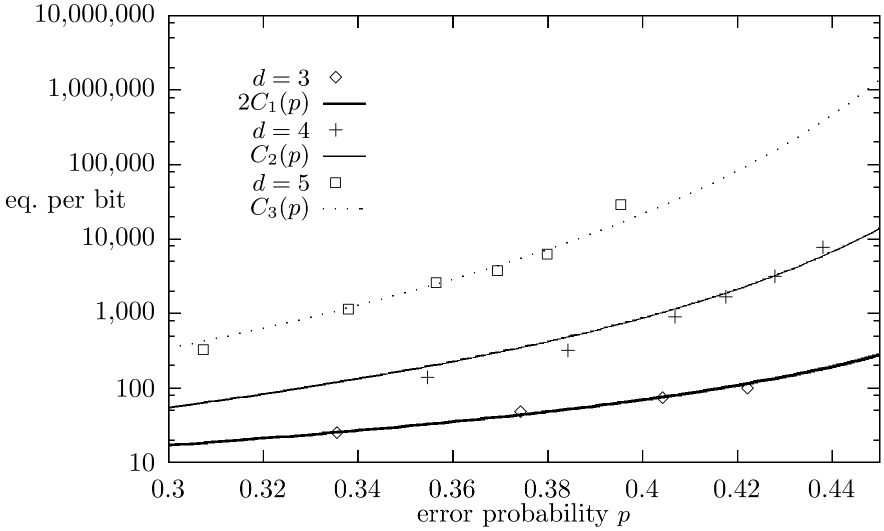

Fig. 4. Number of parity-check equations of weight d per bit required for convergence ($L = 21$)

requires at most 10 iterations. The algorithm then performs approximatively $5(d-1)m(d)N$ operations in average. The amount of involved memory is composed of $(d-1)m(d)$ computer words for storing the parity-check equations and of $2N$ computer words for storing of the sequence $(s_n)_{n<N}$ and the corresponding soft values $(L[n])_{n<N}$.

5 Comparison with previous correlation attacks

We first compare our attack with the correlation attack using convolutional codes described in [6]. This attack associates a convolutional code with memory B to the code $\mathcal{C}$ stemming from the LFSR with feedback polynomial P. The embedded convolutional code is defined by all parity-check equations involving σ_n and $d-1$ bits of σ outside positions $n-1, \ldots, n-B$:

$$\sigma_n + \sum_{i=1}^{B} \beta_i \sigma_{n-i} + \sum_{j=1}^{d-1} \sigma_{i_j} \ .$$

Johansson and Jönsson focus on the case $d = 3$. Using the algorithm described in [6] all these equations can be found with roughly $\frac{N^{d-2}}{(d-2)!}$ operations. Exactly as in Formula (2) the number of such parity-check equations involving the i-th bit of σ is approximatively given by

$$m_B(d) = \frac{N^{d-1} 2^B}{(d-1)! 2^L} \ .$$

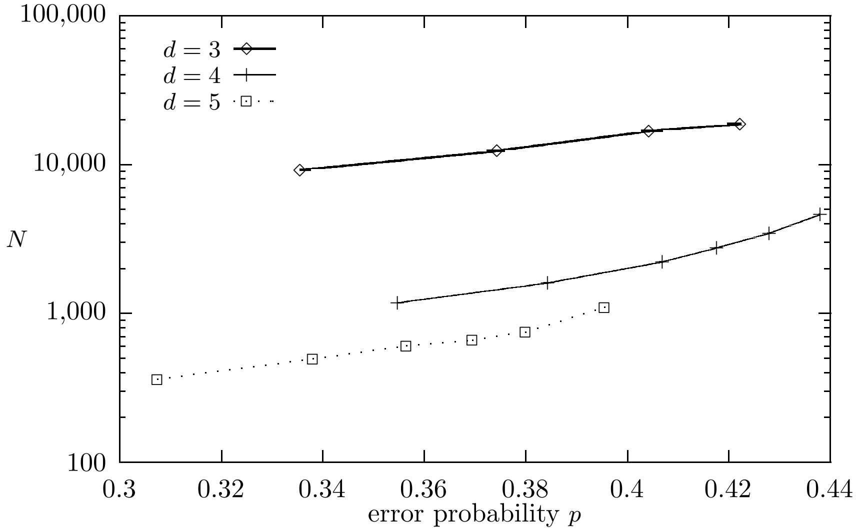

Fig. 5. Number of bits of s required for convergence ($L = 21$)

The decoding step of the attack now consists in deriving a sequence r from $(s_n)_{n<N}$. Decoding r with respect to the convolutional code then provides L consecutive bits of σ. By construction most bits of the corrupted sequence r satisfy

$$Pr[r_n \neq \sigma_n] = \frac{1}{2}(1 - (1 - 2p)^{d-1}) = p_{d-1} \ .$$

This obviously implies that the decoding procedure can not be successful if the transmission rate of the convolutional code R is greater than the capacity of the binary symmetric channel with error probability p_{d-1}. The simulation results presented in [6] actually provide the following maximum value of R for convergence of Viterbi algorithm:

$$R \leq \frac{C_{d-1}(p)}{K'}$$

where K' is a constant which slightly depends on L (we obtain $K' = 3$ for $L = 21$ and $K' = 2.5$ for $L = 40$). Since $R = 1/(m_B(d) + 1)$ we deduce the following convergence condition for Viterbi algorithm:

$$m_B(d) \leq \frac{K'}{C_{d-1}(p)} - 1 \ . \qquad (5)$$

The number of known bits of s required by a correlation attack using a convolutional code with memory B is then

$$N = 2^{\beta_d(p) + \frac{(L-B)}{d-1}} \text{ with } \beta_d(p) = \frac{1}{d-1} \log_2 \left[(d-1)! \frac{K'}{C_{d-1}(p)} \right] \ . \qquad (6)$$

The decoding step using Viterbi algorithm performs $2^B m_B(d)(L+10B)$ operations.

Figure 6 compares the number of bits of s required for a correlation attack using Gallager decoding algorithm with $d = 3$, $d = 4$ (Formula (4)) and for the attack using Viterbi algorithm with $d = 3$, $B = 18$ (Formula (6)). It points out that the use of Gallager algorithm with $d = 4$ provides better performance than the use of Viterbi algorithm with $d = 3$ and $B = 18$. Moreover, this advantage increases for growing p and L. As an example, we now compare our attack with $d = 4$ and the attack using a convolutional code which was presented in [6]($d = 3$). Let N be the minimum number of bits of the running-key which are required by our attack for a given value of p. The correlation attack using Viterbi algorithm only succeeds for these values of N and p when $m_B(3) \geq K'm(4)$, i.e., $B \geq \log_2(N)$ since $K' \simeq 3$. This high value of B makes the complexity of the decoding step with Viterbi algorithm higher than for our attack: the number of operations required for decoding is multiplied by $B + \frac{L}{10}$. Moreover, the memory requirement makes the decoding step intractable for large values of B (B can not exceed 20 or 30 in practice). The only advantage of the attack based on convolutional codes is the lower complexity of the preprocessing step: in our attack the number of operations performed for finding the parity-check equations is multiplied by $\frac{N}{2}$. But this part of the attack is performed once for all while the decoding step should be repeated for each new initialization of the system. A similar comparison can be made for higher weight parity-check equations. Note that, as pointed out by Formula (6), the advantage of increasing

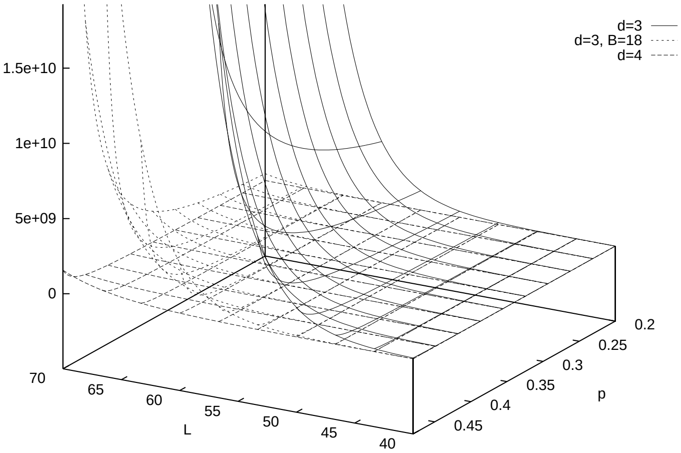

Fig. 6. Number of bits of s required by Gallager algorithm and by Viterbi algorithm

the memory B in the attack based on convolutional codes decreases for higher values of d. Moreover, the value of d in the attack based on convolutional codes is limited due to the time complexity of Viterbi algorithm. Gallager algorithm should therefore be preferred in most situations.

A recent improvement [7] of the attack based on convolutional codes consists in using M parallel convolutional codes with memory B which all share the same information bits. This does not strongly modify the results of the previous comparison. When a turbo code is used, the number of operations performed by the decoding procedure is $6M2^B m_B(d)(L+9B)$ and the memory requirement is roughly the same than in Viterbi algorithm. The processing step now performs around $(L+9B)M\frac{N^{d-2}}{(d-2)!}$ operations.

6 Simulation results

We now present some simulation results of our attack based on a LFSR of length $L = 40$ with feedback polynomial $P(X) = 1 + X + X^3 + X^5 + X^9 + X^{11} + X^{12} + X^{17} + X^{19} + X^{21} + X^{25} + X^{27} + X^{29} + X^{32} + X^{33} + X^{38} + X^{40}$. This polynomial was used for all simulations in [12, 6, 7]. The results obtained by Gallager algorithm with parity-check equations of weight 4 and 5 are presented in Figure 7.

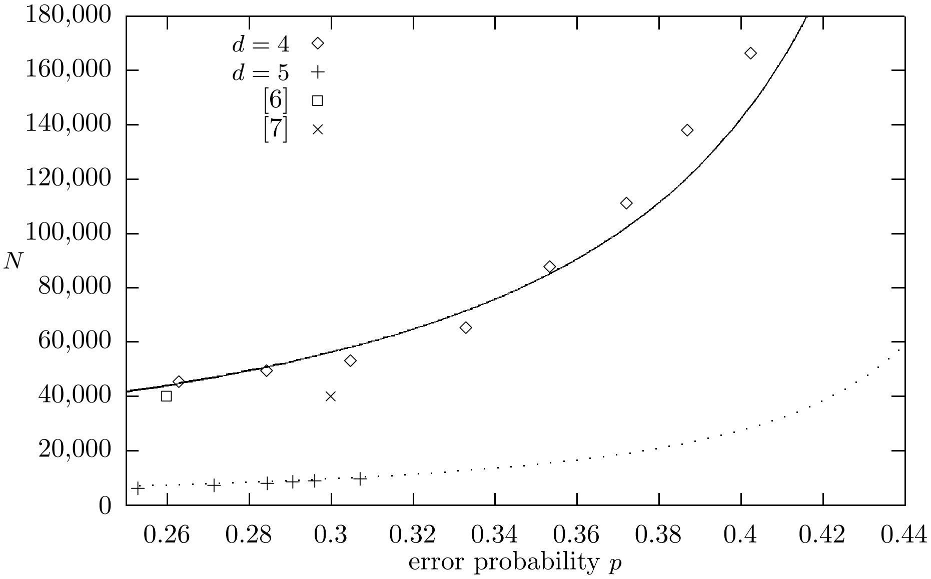

Fig. 7. Number of bits of s required for a fast correlation attack (L=40)

As an example, the following table compares the maximum error-probabilities achieved by the different correlation attacks when $N = 400,000$ bits of s are known:

	our attack	[6] (Viterbi)	[7] (turbo)
	$d = 4$	$d = 3, B = 15$	$d = 3, M = 8, B = 13$
maximum error probability p	0.44	0.40	0.41

For $N = 400,000$ and $p = 0.44$, the preprocessing step and the decoding step of our attack took respectively 9 hours and 1.5 hour on a DEC alpha workstation. Note that the attack based on convolutional codes with $d = 4$ and $B = 16$ can achieve $p = 0.482$, but it requires 2^{53} operations. This error-probability can be achieved by our attack with $d = 5$ and with only $N = 360,000$ bits of s. In this case, the number of operations required by the decoding step is 2^{52}.

Similarly, the correlation attack based on turbo codes achieves $p = 0.3$ for $40,000$ known bits of s (with $B = 15$ and $M = 16$) [7]. We here correct the same error-probability with only $9,770$ bits using parity-check equations of weight 5. In this last case, the preprocessing step takes roughly 30 hours, and the decoding step takes 12 seconds.

7 Conclusions

We have shown that the fast correlation attacks using Gallager iterative decoding algorithm with parity-check equations of weight 4 or 5 are more efficient than the attacks based on convolutional codes or on turbo codes. The performance of our algorithm is only limited by the time complexity of the preprocessing step; however, it is important to note that this part of the attack has to be performed once for all. The different techniques proposed by Johansson and Jönsson could also use higher-weight parity-check equations but the induced improvement is strongly limited by the memory requirement of the decoding procedure. Gallager algorithm should therefore be preferred in most situations. The previous theoretical analysis provides all necessary choices of parameters for practical implementations.

References

1. V. Chepyzhov and B. Smeets. On a fast correlation attack on certain stream ciphers. In *Advances in Cryptology - EUROCRYPT'91*, number 547 in Lecture Notes in Computer Science, pages 176–185. Springer-Verlag, 1991.
2. E. Filiol and C. Fontaine. Highly nonlinear balanced Boolean functions with a good correlation-immunity. In *Advances in Cryptology - EUROCRYPT'98*, number 1403 in Lecture Notes in Computer Science, pages 475–488. Springer-Verlag, 1998.
3. R.G. Gallager. Low-density parity-check codes. *IRE Trans. Inform. Theory*, IT-8:21–28, 1962.
4. R.G. Gallager. *Low-density parity-check codes.* MIT Press, Cambridge, MA, 1963.

5. J. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *IEEE Trans. Inform. Theory*, 42(2):429–445, 1996.
6. T. Johansson and F. Jönsson. Improved fast correlation attack on stream ciphers via convolutional codes. In *Advances in Cryptology - EUROCRYPT'99*, number 1592 in Lecture Notes in Computer Science, pages 347–362. Springer-Verlag, 1999.
7. T. Johansson and F. Jönsson. Fast correlation attacks based on turbo code techniques. In *Advances in Cryptology - CRYPTO'99*, number 1666 in Lecture Notes in Computer Science, pages 181–197. Springer-Verlag, 1999.
8. F.J. MacWilliams and N.J.A. Sloane. *The theory of error-correcting codes.* North-Holland, 1977.
9. W. Meier and O. Staffelbach. Fast correlation attacks on stream ciphers. In *Advances in Cryptology - EUROCRYPT'88*, number 330 in Lecture Notes in Computer Science, pages 301–314. Springer-Verlag, 1988.
10. W. Meier and O. Staffelbach. Fast correlation attack on certain stream ciphers. *J. Cryptology*, pages 159–176, 1989.
11. W. Meier and O. Staffelbach. Nonlinearity criteria for cryptographic functions. In *Advances in Cryptology - EUROCRYPT'89*, number 434 in Lecture Notes in Computer Science, pages 549–562. Springer-Verlag, 1990.
12. M.J. Mihaljević and J.D. Golić. A fast iterative algorithm for a shift register initial state reconstruction given the noisy output sequence. In *Advances in Cryptology - AUSCRYPT'90*, number 453 in Lecture Notes in Computer Science, pages 165–175. Springer-Verlag, 1990.
13. M.J. Mihaljević and J.D. Golić. A comparison of cryptanalytic principles based on iterative error-correction. In *Advances in Cryptology - EUROCRYPT'91*, number 547 in Lecture Notes in Computer Science, pages 527–531. Springer-Verlag, 1991.
14. W. T. Penzhorn. Correlation attacks on stream ciphers: computing low-weight parity checks based on error-correcting codes. In *Fast Software Encryption 96*, number 1039 in Lecture Notes in Computer Science, pages 159–172. Springer-Verlag, 1996.
15. W.T. Penzhorn and G.J. Kühn. Computation of low-weight parity checks for correlation attacks on stream ciphers. In *Cryptography and Coding - 5th IMA Conference*, number 1025 in Lecture Notes in Computer Science, pages 74–83. Springer-Verlag, 1995.
16. R.A. Rueppel. *Analysis and Design of stream ciphers.* Springer-Verlag, 1986.
17. E.S. Selmer. *Linear recurrence relations over finite fields.* PhD thesis, University of Bergen, Norway, 1966.
18. C.E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27, 1948.
19. T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Trans. Inform. Theory*, IT-30(5):776–780, 1984.
20. T. Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Trans. Computers*, C-34(1):81–84, 1985.
21. K. Zeng, C.H. Yang, and T.R.N. Rao. An improved linear syndrome algorithm in cryptanalysis with applications. In *Advances in Cryptology - CRYPTO'90*, number 537 in Lecture Notes in Computer Science. Springer-Verlag, 1990.
22. N. Zierler. Linear recurring sequences. *J. Soc. Indus. Appl. Math.*, 7:31–48, 1959.