J. Cryptology (1989) 1: 159–176

Journal of Cryptology

Fast Correlation Attacks on Certain Stream Ciphers[1]

Willi Meier
HTL Brugg-Windisch, CH-5200 Windisch, Switzerland

Othmar Staffelbach
GRETAG Aktiengesellschaft, Althardstrasse 70,
CH-8105 Regensdorf, Switzerland

Abstract. Suppose that the output of a running key generator employed in a stream cipher is correlated to a linear feedback shift register sequence (LFSR sequence) **a** with correlation probability $p > 0.5$. Then two new correlation attacks (Algorithms A and B) are presented to determine the initial digits of **a**, provided that the number t of feedback taps is small ($t < 10$ if $p \leq 0.75$). The computational complexity of Algorithm A is of order $O(2^{ck})$, where k denotes the length of the LFSR and $c < 1$ depends on the input parameters of the attack, and Algorithm B is polynomial (in fact, even linear) in the length k of the LFSR. These algorithms are much faster than an exhaustive search over all phases of the LFSR, and are demonstrated to be successful against shift registers of considerable length k (typically, $k = 1000$). On the other hand, for correlation probabilities $p \leq 0.75$ the attacks are proven to be infeasible against long LFSRs if they have a greater number of taps (roughly $k \geq 100$ and $t \geq 10$).

Key words. Cryptanalysis, Stream cipher, Correlation, Linear feedback shift register.

1. Introduction

A common type of running key generator employed in stream-cipher systems consists of n (mostly maximum-length) binary linear feedback shift registers (LFSRs) whose output sequences are combined by a nonlinear Boolean function f. Siegenthaler [5]—whose work is based on initial ideas of Blaser and Heinzmann [1]—has shown that if the key stream is correlated to (at least) one of the LFSR sequences, a correlation attack against this individual LFSR-sequence can significantly reduce a brute-force attack. In fact, the output of several combining functions previously proposed in the literature is known to be correlated to some input variables with probabilities p up to 0.75 (this holds, e.g., for the generators of Geffe, Pless, or Bruer). These generators have been broken in [5] for LFSR lengths $k < 50$ (roughly), according to the computational complexity of the attack (based on an

[1] Date received: January 25, 1988. Date revised: March 6, 1989. This work was supported in part by GRETAG Ltd., Regensdorf, Switzerland.

exhaustive search over all phases of the LFSR). But other generators, e.g., certain types of multiplexed sequence generators, are also known to be correlated to LFSR components.

In this paper we derive two types of correlation attacks which are much faster than the above attack and work for $k \gg 50$ if the LFSRs in question have only a few feedback taps (which is sometimes preferred in practice for ease of hardware.) Under suitable conditions, correlation attacks against LFSRs of length $k = 1000$ or even greater are feasible.

In order to set out our results in more detail, assume that N digits of the output sequence $\mathbf{z}$ are given, and correlated with probability $p > 0.5$ to an LFSR sequence $\mathbf{a}$, produced by an LFSR with t taps. We assume that the feedback connection is known. Observe that this is no essential restriction as there are only a very limited number of maximum-length feedback connections with few taps. Hence an exhaustive search over all primitive feedback connections is possible. Our analysis applies to an arbitrary number t of taps but we often restrict analysis to even values of t since irreducible feedback connections (of length greater than 1) necessarily have an even number of taps.

The sequence $\mathbf{z}$ may be viewed as a perturbation of the LFSR sequence $\mathbf{a}$ by a binary asymmetric memoryless noise source (with $\text{Prob}(0) = p$). For the purpose of reconstructing the LFSR sequence $\mathbf{a}$ from $\mathbf{z}$ the following principle is essential to the algorithms (Algorithms A and B): every digit a_n of $\mathbf{a}$ satisfies several linear relations derived from the basic feedback relation, all of them involving t other digits of $\mathbf{a}$. By substituting the corresponding digits of $\mathbf{z}$ in these relations, we obtain equations for each digit z_n, which either may or may not hold. To test whether $z_n = a_n$, we count the number of all equations which hold for z_n. Then the greater the number of these equations which hold, the higher is the probability that z_n will agree with a_n. This is justified by a statistical model introduced in Section 2.

On the basis of this idea, we roughly outline Algorithm A (developed in Section 3): we use the test to search for correct digits (i.e., digits z_n with $z_n = a_n$). This is done by selecting those digits which satisfy the most equations. In this way we obtain an estimate of the sequence $\mathbf{a}$ at the corresponding positions. Under favorable conditions these digits have a high probability of being correct, which means that only a slight modification of our estimate is necesssary. This results in a considerably reduced exhaustive search to sort out sufficiently many correct digits, in order to determine the LFSR sequence $\mathbf{a}$ by solving linear equations.

We give precise conditions under which this procedure is successful, and determine its computational complexity, which in general is of order $O(2^{ck})$, where $c < 1$ is a function of t, p, and N/k. To illustrate this estimate we mention that for $t = 2$ taps, $N/k = 10^6$, and $p \geq 0.6$, the number c is smaller than 0.25, and for $p > 0.67$, c is below 0.001 (see Table 2). This is a considerable improvement compared with exhaustive search, where $c = 1$. On the other hand, for large t ($t \geq 10$) our estimate shows that c comes very close to $H(p)$, where $H(p)$ denotes the binary entropy function. This proves that Algorithm A for large t gives no advantage over (a modified) exhaustive search (see Section 3).

In Algorithm B (developed in Section 4) we do not search for the most reliable digits. Instead we take into account all digits of $\mathbf{z}$, together with their probabilities of being correct. *A priori*, with probability p a digit of $\mathbf{z}$ agrees with the correspond-

ing digit of **a**. Now we assign to each digit z_n of **z** a new probability p^*, which is the probability for $z_n = a_n$, conditioned by the number of equations satisfied. This procedure can be iterated with the varied new probabilities p^* as input to every round. After a few rounds, all those digits of **z** whose probability p^* is lower than a certain threshold are complemented. Under suitable conditions we can expect that the number of incorrect digits decreases. In this case we restart the whole process several times, with the new sequence in place of **z**, until we end up with the original LFSR sequence **a** (see Table 4).

To obtain conditions under which Algorithm B succeeds, a function $F(p, t, N/k)$ is introduced to measure the correction effect. Thus if $F(p, t, N/k) \leq 0$ there is no correction effect and Algorithm B will not be able to reproduce the LFSR sequence **a**. Therefore we get a definite limit to this attack (which is attained for $t \geq 10$ if $p \leq 0.75$). In the other direction, investigations of $F(p, t, N/k)$ show that for $t = 2$ or $t = 4$ taps Algorithm B still remains effective for small correlations, and, in fact, for $t = 2$, even for correlation probabilities quite close to 0.5 (see Tables 3 and 5). This means in particular that correlation to LFSRs with only two feedback taps can be very dangerous. The striking efficiency of Algorithm B, as observed in numerous experiments, is explained by the fact that its computational complexity is of order $O(k)$ (i.e., *linear* in length k of the LFSR, for fixed t, p, and N/k).

Algorithms A and B enable attacks against LFSRs of considerable length (e.g., $k = 1000$ or greater) with software implementation. However, a comparison shows that Algorithm A is preferable if $c \ll 1$ and p is near 0.75, whereas Algorithm B becomes more efficient for probabilities p near 0.5. (Simulations of Algorithm B have been shown to be successful in attacks with $p = 0.55$ even on a personal computer.)

The methods developed for Algorithms A and B allow several generalizations and conclusions (see Section 5). To prevent attacks based on these methods, suitable precautions are necessary. This leads to new design criteria for stream ciphers:

1. Any correlation to an LFSR with less than 10 taps should be avoided.
2. There should be no correlation to a *general* LFSR of length shorter than 100 (especially when the feedback connection is assumed to be known).

In [3] and [4] methods have been proposed to face correlation. In applications of these methods the above criteria have to be taken into consideration.

2. Algebraic and Statistical Foundations

2.1. Preliminaries

Assume that the output sequence **z** of a binary key stream generator is correlated to an LFSR-sequence **a**, and let p denote the correlation probability between **a** and **z**, i.e.,

$$p = \text{Prob}(z_n = a_n) > 0.5.$$

Recall that the LFSR-sequence **a** is given by a linear relation of the form

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}. \tag{2.1}$$

Moreover, $c(X) = c_0 + c_1X + c_2X^2 + \cdots + c_kX^k$ (with $c_0 = 1$) is called the feedback polynomial of the relation. The number t of feedback taps is equal to the number of nonzero terms in $\{c_1, c_2, \ldots, c_k\}$. Moreover, (2.1) may be written as an equation with $(t + 1)$ terms

$$\sum_{\{i:0\le i\le k, c_i\neq 0\}} a_{n-i} = 0. \tag{2.1'}$$

By shifting the sequence **a** we observe that a fixed digit a_n appears in every $t + 1$ positions of (2.1'), i.e., it simultaneously satisfies $t + 1$ equations of the form (2.1') or (2.1), respectively.

Every polynomial multiple of $c(X)$ defines a linear relation for **a**. This applies in particular for powers $c(X)^j$ for exponents $j = 2^i$, where we have $c(X)^j = c(X^j)$. In this way we obtain a variety of linear relations beyond those obtained by shifting, all having the same number t of feedback taps. (The latter property is crucial, as the feasibility of our correlation attacks developed in Sections 3 and 4 depends on the number of taps.) In fact, our attack will test all these linear relations for the given sequence **z** to decide whether z_n agrees with a_n for a given n.

Suppose this a_n is fixed. Then the linear relations as obtained above can be written in the form

$$\begin{aligned} L_1 &= a + b_1 = 0, \\ L_2 &= a + b_2 = 0, \\ &\vdots \\ L_m &= a + b_m = 0, \end{aligned} \tag{2.2}$$

where $a = a_n$ and each b_i, $i = 1, \ldots, m$, is a sum of exactly t different terms of the LFSR sequence **a**. The number m of these relations is determined in Section 3 (see formula (3.3)).

Instead of the LFSR-sequence **a** we may substitute the digits of the given sequence **z** (at the same index positions) in equations (2.2), thus giving rise to expressions

$$L_i = z + y_i, \qquad i = 1, \ldots, m, \tag{2.3}$$

where L_i is not necessarily 0.

2.2. *The Statistical Model*

Motivated by these facts, we introduce the following statistical model which is essential to our method. We begin by replacing the digits in equations (2.2) with a set of binary random variables $A = \{a, b_{11}, b_{12}, \ldots, b_{1t}, b_{21}, b_{22}, \ldots, b_{2t}, \ldots, b_{m1}, b_{m2}, \ldots, b_{mt}\}$, satisfying the corresponding equations

$$\begin{aligned} a + b_{11} + b_{12} + \cdots + b_{1t} &= 0, \\ a + b_{21} + b_{22} + \cdots + b_{2t} &= 0, \\ &\vdots \\ a + b_{m1} + b_{m2} + \cdots + b_{mt} &= 0. \end{aligned} \tag{2.4}$$

In a similar way we introduce a set of binary random variables $Z = \{z, y_{11}, y_{12}, \ldots,$

$y_{1t}, y_{21}, y_{22}, \ldots, y_{2t}, \ldots, y_{m1}, y_{m2}, \ldots, y_{mt}\}$ representing the digits of the sequence $\mathbf{z}$ in (2.3).

The two sets A and Z of random variables are related by

$$\text{Prob}(z = a) = p \quad \text{and} \quad \text{Prob}(y_{ij} = b_{ij}) = p. \tag{2.5}$$

Besides the relations derived from (2.4) and (2.5) these random variables are independent and identically distributed with equal probability 0.5 for being 0 or 1. For $i = 1, \ldots, m$ we derive the random variables

$$b_i = b_{i1} + b_{i2} + \cdots + b_{it}$$

and (2.6)

$$y_i = y_{i1} + y_{i2} + \cdots + y_{it}$$

as well as

$$L_i = z + y_i.$$

Moreover, we denote by s the probability of b_i and y_i to be equal,

$$s = \text{Prob}(y_i = b_i), \tag{2.7}$$

which by (2.5) and (2.6) is independent of the index i. This probability is a function of p and t, $s = s(p, t)$. It can be computed using the recursion

$$\begin{aligned} s(p, t) &= ps(p, t-1) + (1-p)(1 - s(p, t-1)), \\ s(p, 1) &= p. \end{aligned} \tag{2.8}$$

Next we consider the random variables $L_1, L_2, \ldots, L_m$. The probability that the outcome of these random variables vanishes for a given fixed set of exactly h indices is determined by expression (2.9). (Note that $L_i = 0$ implies either $z = a$ and $y_i = b_i$ or $z \neq a$ and $y_i \neq b_i$.)

$$ps^h(1-s)^{m-h} + (1-p)(1-s)^h s^{m-h}. \tag{2.9}$$

For simplicity assume that $L_1 = 0$, $L_2 = 0$, $\ldots$, $L_h = 0$ and $L_{h+1} = 1$, $L_{h+2} = 1$, $\ldots$, $L_m = 1$. Then the corresponding conditional probabilities P (for $z = a$ and $z \neq a$) can be computed as

$$P(z = a | L_1 = \cdots = L_h = 0, L_{h+1} = \cdots = L_m = 1) = \frac{ps^h(1-s)^{m-h}}{ps^h(1-s)^{m-h} + (1-p)(1-s)^h s^{m-h}}, \tag{2.10}$$

$$P(z \neq a | L_1 = \cdots = L_h = 0, L_{h+1} = \cdots = L_m = 1) = \frac{(1-p)(1-s)^h s^{m-h}}{ps^h(1-s)^{m-h} + (1-p)(1-s)^h s^{m-h}}. \tag{2.11}$$

In terms of these random variables and the probabilities thereof, we are now able to set out the basic idea of the method. We consider a random experiment according to our statistical model. Thereby we have access to the outcome of z and y_{ij} (and hence to $L_i = z + y_i$), but not to the outcome of a and b_{ij}. (In our application z and y_{ij} correspond to certain digits of the given sequence, whereas a and b_{ij} refer to the

unknown LFSR sequence. In particular, z corresponds to the fixed digit z_n, and a to the fixed digit a_n we wish to determine.) We start with the *a priori* probability $p > 0.5$ that $z = a$ and count the number h of indices i for which $L_i = 0$. Then we alter the *a priori* probability $p = \text{Prob}(z = a)$ to a new probability p^* according to formula (2.10).

We intuitively expect that this new probability p^* increases if $z = a$ and decreases if $z \neq a$. To justify this idea we compute the expected value of p^* for these two cases.

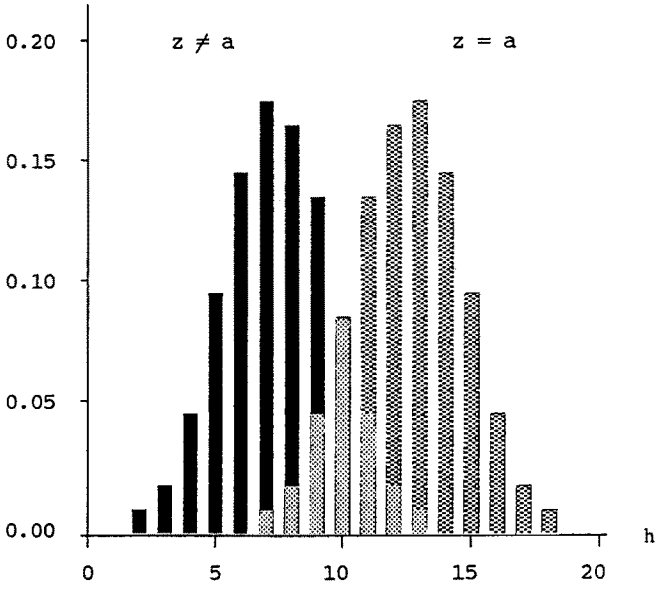
Case 1: $z = a$.

$$E_0[p^*] = E[p^*|z = a] = \sum_{h=0}^{m} \binom{m}{h} \frac{ps^h(1-s)^{m-h}}{ps^h(1-s)^{m-h} + (1-p)(1-s)^h s^{m-h}} s^h(1-s)^{m-h}. \tag{2.12}$$

Case 2: $z \neq a$.

$$E_1[p^*] = E[p^*|z \neq a] = \sum_{h=0}^{m} \binom{m}{h} \frac{ps^h(1-s)^{m-h}}{ps^h(1-s)^{m-h} + (1-p)(1-s)^h s^{m-h}} s^{m-h}(1-s)^h. \tag{2.13}$$

For example, with *a priori* probability $p = 0.75$, $t = 2$ taps, and $m = 20$ relations we get $E_0[p^*] = 0.9$ and $E_1[p^*] = 0.3$. Thus the probability p^* is expected to



p* as function of number h of relations satisfied (p=0.75)

h	p*
0	0.00011
1	0.00030
2	0.00085
3	0.00235
4	0.00649
5	0.01782
6	0.04797
7	0.12278
8	0.27995
9	0.51923
10	0.75000
11	0.89286
12	0.95859
13	0.98469
14	0.99443
15	0.99799
16	0.99927
17	0.99974
18	0.99991
19	0.99997
20	0.99999

Fig. 1

increase in the case where $z = a$ and to decrease in the case where $z \neq a$. But from (2.12) and (2.13) we conclude

$$E[p^*] = pE_0[p^*] + (1-p)E_1[p^*] = \sum_{h=0}^{m} \binom{m}{h} ps^{m-h}(1-s)^h = p \qquad (2.14)$$

which implies that the overall expected value of the new probability p^* remains unchanged.

Figure 1 shows the probability distribution for the number h of relations satisfied under the conditions of the above example ($p = 0.75$, $t = 2$, $m = 20$) in both cases. These are binomial distributions with respect to s and $1 - s$ (see also (3.2)). Moreover, Fig. 1 gives the new probability p^* as a function of h. This example illustrates that the distributions make a clear distinction between the two cases which will give us the main criterion in our attack to decide whether $z = a$ or $z \neq a$.

3. An Efficient Exponential-Time Attack

3.1 Description of the Attack

Suppose that N digits of the sequence $\mathbf{z}$ and the structure of the LFSR (of length k with t feedback taps) are known, and that the output sequence $\mathbf{a}$ of the LFSR is correlated with probability p to the given sequence $\mathbf{z}$. The problem to be solved consists in finding the unknown LFSR sequence $\mathbf{a}$. Basically, this sequence can be constructed out of any k of its digits by solving linear equations for the initial state. (If these equations are linearly dependent we can choose some additional digits to obtain a linearly independent system.) Therefore, to get an estimate of the sequence $\mathbf{a}$ we essentially select k digits of $\mathbf{z}$ with the highest probability p^* (see Section 2), or equivalently k digits which satisfy the most relations (2.2).

The probability $Q(p, m, h)$ that a fixed digit z satisfies at least h of m relations is computed by the formula

$$Q(p, m, h) = \sum_{i=h}^{m} \binom{m}{i} (ps^i(1-s)^{m-i} + (1-p)(1-s)^i s^{m-i}) \qquad (3.1)$$

which is a consequence of (2.9). Moreover, the probability $R(p, m, h)$ that $z = a$ and that at least h of m relations hold, is given by

$$R(p, m, h) = \sum_{i=h}^{m} \binom{m}{i} ps^i(1-s)^{m-i}. \qquad (3.2)$$

Thus the probability for $z = a$, given that at least h of m relations are satisfied, is the quotient $T(p, m, h) = R(p, m, h)/Q(p, m, h)$.

Therefore $Q(p, m, h) \cdot N$ digits are expected to satisfy at least h relations and these digits have probability $T(p, m, h)$ to be correct. For fixed p and m the value $T(p, m, h)$ is increasing with h. Thus in order to find sufficiently many (i.e., at least k) digits with the highest reliability we determine the maximum h with $Q(p, m, h) \cdot N \geq k$.

As a reference guess I_0 for $\mathbf{a}$ we select the digits of $\mathbf{z}$ at the index positions where at least h relations are satisfied. Then $(1 - T(p, m, h)) \cdot Q(p, m, h) \cdot N$ is the expected

number of erroneous digits in I_0. If this number is small, **a** can be found by trying slight modifications of I_0 which are tested by correlating the corresponding phase of the LFSR with the given sequence **z**. A phase is accepted if the correlation exceeds a suitable threshold (see [5]).

To complete our analysis it remains to estimate the average number m of relations available as a function of the length k of the LFSR and the length N of a known portion of the sequence **z** (ciphertext). The relations (2.2) obtained by i iterated squaring operations ($i \geq 0$) have length $2^i k$, and there are $N - 2^i k$ of them. As this number has to be positive, i cannot be larger than the integer part of $\log(N/k)$. (Here log denotes logarithm to the basis 2. For simplicity we also write log for the integer part of $\log_2$ throughout.) As a consequence, the total number of relations can be estimated by

$$\begin{aligned} T &= \sum_{i=0}^{\log(N/k)} (N - 2^i k) = N\left(\log\left(\frac{N}{k}\right) + 1\right) - \sum_{i=0}^{\log(N/k)} 2^i k \\ &= N\left(\log\left(\frac{N}{k}\right) + 1\right) - (2^{\log(N/k)+1} - 1)k \\ &= N \log\left(\frac{N}{k}\right) + N - \left(\frac{2N}{k} - 1\right)k \\ &= N \log\left(\frac{N}{k}\right) + N - 2N + k \\ &= N \log\left(\frac{N}{2k}\right) + k. \end{aligned}$$

Now every relation applies to $t + 1$ digits of **z**. Therefore the average number m of relations per digit is

$$T \cdot \frac{t+1}{N} = \left(\log\left(\frac{N}{2k}\right) + \frac{k}{N}\right)(t+1).$$

In our application $(k/N)(t + 1) \ll 1$. Hence the above formula simplifes to

$$m = m(N, k, t) \approx \log\left(\frac{N}{2k}\right)(t+1). \tag{3.3}$$

3.2. Algorithm A

Step 1. Determine m according to formula (3.3).
Step 2. Find the maximum value of h such that $Q(p, m, h) \cdot N \geq k$.
Step 3. Search for the digits of **z** satisfying at least h relations and use these digits as a reference guess I_0 of **a** at the corresponding index positions.
Step 4. Find the correct guess by testing modifications of I_0 with Hamming distance 0, 1, 2, ..., by correlation of the corresponding LFSR sequence with the sequence **z**

Note that m as computed in step 1 is only a mean value. In general the digits near

the middle of the given part of **z** satisfy more relations than the digits near the boundaries. Therefore in the middle a clearer distinction between correct and incorrect digits is possible. This leads to an improvement of the algorithm, replacing step 3 by step 3′:

Step 3′. Compute the new probability p^* according to formula (2.10) for the given digits of **z** and choose k digits having the highest probability p^*.

The average number of erroneous digits in I_0 (step 3) is computed as $\bar{r} = (1 - T(p, m, h)) \cdot k$. Under favorable conditions (e.g., if $\bar{r} \ll 1$) step 4 is not necessary. This is illustrated by the following example.

Example 1. Assume that **z** has length $N = 5000$ and is correlated with probability $p = 0.75$ to an LFSR of length $k = 100$ having $t = 2$ feedback taps. Then, on average, we obtain $m = 12$ relations to test the digits of **z**. Furthermore, computations of the functions Q and T show that $h \geq 11$ relations are expected to hold for $Q(p, m, 11) \cdot N = 0.02189 \cdot 5000 \approx 109$ digits, and $(1 - T(p, m, 11)) \cdot 109 = 0.001855 \cdot 109 = 0.2 < 1$ digits among these are expected to be incorrect. Thus with high probability all digits selected in step 3 are correct.

3.3. Computational Complexity of Algorithm A

We now determine the computational complexity of the algorithm. Since the computation time for steps 1–3 is negligible we only estimate the average number of trials needed in step 4. Suppose that exactly r among the digits as found in step 3 are incorrect. Then the maximum number of trials in step 4 is

$$A(k, r) = \sum_{i=0}^{r} \binom{k}{i}.$$

For this formula there exists a well-known estimate using the binary entropy function

$$H(0) = H(1) = 0,$$

$$H(x) = -x \log x - (1 - x) \log(1 - x) \qquad (0 < x < 1).$$

Then (see p. 20 of [6])

$$A(k, r) = \sum_{i=0}^{r} \binom{k}{i} \leq 2^{H(\theta)k} \tag{3.4}$$

with $\theta = r/k$. In applications only the average number $\bar{r} = (1 - T(p, m, h)) \cdot k$ for r is available. For large k, the probability that r exceeds $\bar{r}$ is limited roughly by $\frac{1}{2}$. Therefore, replacing r by $\bar{r}$ in (3.4) we obtain an estimate of the number of trials in step 4.

As a consequence the search in Algorithm A has computational complexity $O(2^{ck})$ where $c = H(\bar{r}/k)$ lies in the interval $0 \leq c \leq 1$. The case $c = 1$ corresponds to the exhaustive search over all phases of the shift register. However, under favorable conditions we have $c \ll 1$, which means that the attack is much faster than the exhaustive search. Therefore we investigate the dependency of c as a function of the

Table 1. $c(p, t, N/k)$ for $N/k = 100$.

	t								
p	2	4	6	8	10	12	14	16	∞
0.51	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
0.53	0.994	0.997	0.997	0.997	0.997	0.997	0.997	0.997	0.997
0.55	0.973	0.993	0.993	0.993	0.993	0.993	0.993	0.993	0.993
0.57	0.927	0.985	0.986	0.986	0.986	0.986	0.986	0.986	0.986
0.59	0.846	0.973	0.976	0.976	0.977	0.977	0.977	0.977	0.977
0.61	0.729	0.956	0.964	0.965	0.965	0.965	0.965	0.965	0.965
0.63	0.584	0.930	0.949	0.951	0.951	0.951	0.951	0.951	0.951
0.65	0.432	0.890	0.930	0.934	0.934	0.934	0.934	0.934	0.934
0.67	0.293	0.832	0.905	0.914	0.915	0.915	0.915	0.915	0.915
0.69	0.122	0.750	0.871	0.890	0.893	0.893	0.893	0.893	0.893
0.71	0.062	0.641	0.825	0.860	0.867	0.868	0.869	0.869	0.869
0.73	0.028	0.462	0.761	0.822	0.837	0.840	0.841	0.841	0.841
0.75	0.012	0.314	0.671	0.772	0.800	0.808	0.810	0.811	0.811

Table 2. $c(p, t, N/k)$ for $N/k = 10^6$.

	t								
p	2	4	6	8	10	12	14	16	∞
0.51	0.999	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
0.53	0.976	0.997	0.997	0.997	0.997	0.997	0.997	0.997	0.997
0.55	0.870	0.992	0.993	0.993	0.993	0.993	0.993	0.993	0.993
0.57	0.642	0.982	0.986	0.986	0.986	0.986	0.986	0.986	0.986
0.59	0.362	0.963	0.976	0.976	0.976	0.977	0.977	0.977	0.977
0.61	0.132	0.926	0.963	0.965	0.965	0.965	0.965	0.965	0.965
0.63	0.039	0.856	0.945	0.950	0.951	0.951	0.951	0.951	0.951
0.65	0.007	0.734	0.917	0.932	0.934	0.934	0.934	0.934	0.934
0.67	0.001	0.555	0.875	0.910	0.914	0.915	0.915	0.915	0.915
0.69	0.000	0.327	0.805	0.880	0.891	0.893	0.893	0.893	0.893
0.71	0.000	0.150	0.692	0.836	0.863	0.868	0.868	0.869	0.869
0.73	0.000	0.043	0.515	0.768	0.825	0.838	0.841	0.841	0.841
0.75	0.000	0.009	0.311	0.660	0.771	0.800	0.808	0.811	0.811

input parameters N, k, t, and p. We start with the observation that c is a function of t, p, and N/k only. This can be seen by inspection of the formulas involved in steps 1 and 2 of the algorithm. In high-security applications large values $d = N/k$ have to be considered. Even ratios d of magnitude 10^6 or larger may be reasonable for a cryptographer. Hence for different but fixed N/k we study c as function of t and p (see Tables 1 and 2).

3.4. Discussion

For a sequence $\mathbf{z}$ of length N correlated to the LFSR sequence $\mathbf{a}$ with probability p, the Hamming distance between $\mathbf{a}$ and $\mathbf{z}$ is expected to be $(1 - p) \cdot N$. If $d = N/k$

is small, there may be different phases of **a** with distance less than or equal to $(1-p)\cdot N$, i.e., there are several solutions to the correlation problem. In this case Algorithm A may select a wrong phase of **a**.

With increasing number t of taps, $c(p, t, N/k)$ converges to a limit which is given by $H(p)$ as shown in the last column of Tables 1 and 2. The proof of this fact is outlined as follows: if t tends to infinity the function $s(p, t)$ (see (2.8)) approaches $\frac{1}{2}$. Moreover, from formulas (3.1) and (3.2) we deduce that $T(p, m, h) = R(p, m, h)/Q(p, m, h) = p$ for $s = \frac{1}{2}$. This means that $\theta = r/k$ converges to $1-p$, hence $c(p, \infty, N/k) = H(1-p) = H(p)$. This limit $c = H(p)$ has a cryptologic significance for the correlation attack as described in [5]: if the exhaustive search over all phases is modified to start with the most probable error pattern (see step 4 of Algorithm A), its computational complexity is of order $O(2^{ck})$ instead of $O(2^k)$. The tables show that for $p = 0.75$ this results in a reduction from $c = 1$ to $c = 0.811$.

For $t = 2$ taps and probabilities $p \geq 0.6$ the tables show an enormous improvement over the exhaustive search. Using Algorithm A even correlation attacks against shift registers of length 1000 or more become feasible. (For $N/k = 10^6$ and $p > 0.67$ the entries for c in Table 2 are below 0.0005.)

A comparison of the two tables shows that by increasing the ratio N/k there is a substantial improvement of the attack for $t < 10$ taps. (To further illustrate this point for even higher ratios, take $N/k = 10^9$, $p = 0.57$, and $t = 2$, then the value of c turns out to be 0.408, compared with $H(0.57) = 0.986$.) For $t \geq 10$ taps and probabilities $p \leq 0.75$ the values for c are so close to the asymptotic values that Algorithm A gives no essential advantage over the (modified) exhaustive search. This holds for all values of N/k which may occur in practical applications.

4. A Polynomial-Time Attack

4.1. Description of the Attack

The attack of this section is motivated by the fact that the conditional probability p^* is small if a digit satisfies only a few relations (see Figure 1). This leads to the method of complementing a digit if it satisfies less than a certain number of relations. Under favorable conditions the "corrected" sequence can be expected to have less digits differing from the LFSR sequence **a**. The idea is to repeat this process until we end up reproducing the LFSR sequence **a**.

Our investigations show, however, that there is an alternative and better approach which leaves the whole sequence **z** unchanged in the first instance and instead assigns the new probability p^* to every digit. Then the process of assigning probabilities is iterated before "correcting" the sequence. For either method we need to analyze the effect of complementing digits which satisfy less than a certain number of relations.

The probability that at most h of m relations are satisfied is computed as

$$U(p, m, h) = \sum_{i=0}^{h} \binom{m}{i} (ps^i(1-s)^{m-i} + (1-p)(1-s)^i s^{m-i}). \tag{4.1}$$

Moreover, the probability that $z = a$ and that at most h of m relations are satisfied

is given by

$$V(p, m, h) = \sum_{i=0}^{h} \binom{m}{i} p s^i (1 - s)^{m-i} \tag{4.2}$$

and similarly the probability that $z \neq a$ and that at most h of m relations are satisfied is

$$W(p, m, h) = \sum_{i=0}^{h} \binom{m}{i} (1 - p)(1 - s)^i s^{m-i}. \tag{4.3}$$

Therefore $U(p, m, h) \cdot N$ is the expected number of digits of $\mathbf{z}$ which satisfy at most h relations. If these digits are complemented, $W(p, m, h) \cdot N$ is the number of correctly changed digits and $V(p, m, h) \cdot N$ is the number of erroneously changed digits. Thus the increase of correct digits is the difference $W(p, m, h) \cdot N - V(p, m, h) \cdot N$, and the relative increase is computed as

$$I(p, m, h) = W(p, m, h) - V(p, m, h). \tag{4.4}$$

Hence optimum correction is obtained by choosing $h = h_{\max}$ such that $I(p, m, h)$ is maximum for given p and m.

For our refined method which takes p^* into account, we replace $h_{\max}$ by a corresponding probability threshold on p^*, which is chosen as

$$p_{\text{thr}} = \tfrac{1}{2}(p^*(p, m, h_{\max}) + p^*(p, m, h_{\max} + 1)) \tag{4.5}$$

in order to achieve maximum correction effect. Then the expected number of digits with p^* below p_{thr} is

$$N_{\text{thr}} = U(p, m, h_{\max}) \cdot N. \tag{4.6}$$

If there are only a few digits with p^* below p_{thr} the assignment of new probabilities is iterated. For this we need a generalization of formula (2.8) for $s(p, t)$ to the situation where each of the t digits may have different probabilities $p_1, p_2, \ldots, p_t$:

$$\begin{aligned} s(p_1, \ldots, p_t, t) &= p_t s(p_1, \ldots, p_{t-1}, t - 1) + (1 - p_t)(1 - s(p_1, \ldots, p_{t-1}, t - 1)), \\ s(p_1, 1) &= p_1. \end{aligned} \tag{4.7}$$

This generalization carries over to all other formulas, in particular to formula (2.10) for p^*.

As numerous experiments have revealed, only a very limited number α of such iterations is reasonable. (In many cases $\alpha = 5$ is a suitable choice.)

We are now prepared to formulate the algorithm in detail.

4.2. Algorithm B

Step 1. *Determine* m according to formula (3.3).

Step 2. Find the value of $h = h_{\max}$ such that $I(p, m, h)$ is maximum. Compute p_{thr} and N_{thr} according to (4.5) and (4.6).

Step 3. Initialize the iteration counter $i = 0$.

Step 4. For every digit of $\mathbf{z}$ compute the new probability p^* (see (2.10) and (4.7)) with respect to the individual number of relations satisfied. Determine the number N_w of digits with $p^* < p_{\text{thr}}$.

Step 5. If $N_w < N_{\text{thr}}$ or $i < \alpha$ increment i and go to step 4.
Step 6. Complement those digits of $\mathbf{z}$ with $p^* < p_{\text{thr}}$ and reset the probability of each digit to the original value p.
Step 7. If there are digits of $\mathbf{z}$ not satisfying (2.1) go to step 3.
Step 8. Terminate with $\mathbf{a} = \mathbf{z}$.

In the subsequent discussion which is based on simulations with this algorithm, the outer loops (steps 3–7) are called rounds, whereas the inner loops (steps 4 and 5) are referred to as iterations.

In the first round it was observed that $N_w \approx N_{\text{thr}}$ as expected by theory. (Hence only one iteration of steps 4 and 5 may be necessary.) In higher rounds the errors $(z_n \neq a_n)$ are no longer independent of the relations. Thus our statistical model does not strictly apply anymore. This is reflected by the observation that $N_w \ll N_{\text{thr}}$ in higher rounds. For this reason we iterate the assignment of new probabilities until there are enough digits with p^* below p_{thr}. However, after a few iterations a strong polarization can be observed between digits having probability p^* either very close to 0 or very close to 1. Apart from a few digits, this polarization tends to become stable, which means that we need not iterate any longer. This justifies the termination of a round after a limited number α of iterations.

In performing Algorithm B we observe a steady correction effect which after several rounds terminates with $\mathbf{a} = \mathbf{z}$. To explain this experience (see also Example 2), the statistical dependencies between the different rounds should be taken into account. As this would require a different approach we do not pursue this line further (see also the open problems in Section 5).

Algorithm B allows some modifications, e.g., in step 6 the probabilities could be reset to values higher than the original p, according to the decrease of the expected number of errors after each round. Simulations have revealed, however, that this does not lead to an improvement of Algorithm B.

4.3. Computational Complexity and Limits of the Attack

To estimate the correction effect in Algorithm B, we have to calculate $I_{\max} = I(p, m, h_{\max})$ (step 2) for given p, t, N, and k. First observe that m is a function of t and $d = N/k$ (see (3.3)). Moreover, $h_{\max}$ is a function of p and m. Therefore $I_{\max}$ is a function of p, t, and d, $I_{\max} = I_{\max}(p, t, d)$. The expected number of digits corrected in one iteration is computed as

$$N_c = I_{\max}(p, t, d) \cdot N. \tag{4.8}$$

It is convenient to express N_c as $N_c = F(p, t, d) \cdot k$, where

$$F(p, t, d) = I_{\max}(p, t, d) \cdot d \tag{4.9}$$

is a correction factor independent of k. If $F(p, t, d) \leq 0$ no correction effect can be expected and the attack will fail. On the other hand, if the parameters of the attack result in $F(p, t, d) \geq 0.5$, Algorithm B has appeared to be very successful in most experiments (see Example 2). For fixed t and d we have computed the smallest correlation probability p such that $F(p', t, d) \geq 0.5$ for $p' \geq p$, as shown in Table 3.

From Table 3 we see that for $t < 8$ taps the correlation-probability limit necessary for a successful attack is in a range relevant in practice. In particular for $t = 2$

Table 3. p with $F(p, t, d) = 0.5$.

	t								
d	2	4	6	8	10	12	14	16	18
10	0.761	0.880	0.980	0.980	0.980	0.980	0.980	0.980	0.980
10^2	0.595	0.754	0.824	0.863	0.889	0.905	0.917	0.926	0.934
10^3	0.553	0.708	0.787	0.832	0.861	0.882	0.897	0.908	0.918
10^4	0.533	0.679	0.763	0.812	0.844	0.867	0.883	0.896	0.906
10^5	0.525	0.663	0.748	0.800	0.833	0.857	0.875	0.889	0.900
10^6	0.519	0.650	0.737	0.789	0.825	0.849	0.868	0.883	0.894
10^7	0.515	0.641	0.727	0.781	0.817	0.843	0.862	0.877	0.890
10^8	0.514	0.634	0.720	0.774	0.812	0.838	0.858	0.874	0.886
10^9	0.512	0.628	0.714	0.770	0.807	0.833	0.854	0.870	0.882
10^{10}	0.510	0.621	0.709	0.764	0.802	0.830	0.850	0.866	0.879

this probability comes very close to 0.5. In these cases the algorithm is successful on very long LFSRs. The following example illustrates the efficiency of the attack.

Example 2. Referring to the entry 0.754 in Table 3 for $t = 4$ and $d = 100$ we consider the following situation: $N = 10{,}000$, $k = 100$, $t = 4$, and $p = 0.75$ (instead of 0.754). Then $d = 100$ and $F(p, t, d)$ turns out to be 0.392 (instead of 0.5 of Table 3). The parameters of Algorithm B can be computed as $p_{\text{thr}} = 0.524$, $N_{\text{thr}} = 448$. Thus 448 digits are expected to be changed in the first iteration resulting in a decrease of wrong digits by 39. Table 4 shows the intermediate results after each step. The entry in the third column always indicates the decrease of wrong digits if correction had been applied.

This example deserves some remarks:

In round 1 after iteration 1 the figures are very close to the numbers as predicted by theory. However, after the first iteration in higher rounds the figures are lower than expected, due to statistical dependencies as explained earlier. Nevertheless, we observe an increasing correction effect after some more iterations. As a consequence all errors are eliminated after only a few rounds (5 rounds and 12 iterations in total in Example 2).

The number of rounds/iterations needed is basically dependent on p, t, and d only, but not on the length k of the LFSR. Assume that in the above example N is replaced by $N = 100{,}000$ and k by $k = 1000$. As the corresponding correction factor $F(p, t, d)$ remains unchanged the same number of iterations can be expected to be necessary. Moreover, for every individual digit of the sequence $\mathbf{z}$ the amount of computation needed to calculate the new probability p^* also remains unchanged. Since the sequence $\mathbf{z}$ has ten times as many digits the computational complexity of the algorithm increases by the same factor. As this argument holds in complete generality, we conclude that the computational complexity of Algorithm B grows linearly with the LFSR length k, i.e., is of *order* $O(k)$.

We continue with the observation that in Example 2 even a correction factor $F(p, t, d)$ of less than 0.5 has led to a successful attack. In fact there are examples

Table 4. Protocol of an attack based on Algorithm B.

	Number of digits with $p^* < p_{thr}$	Number of wrong digits with $p^* < p_{thr}$	Decrease of wrong digits	Number of wrong digits after correction
Round 1				
Iteration 1	430	246	62	2500
Iteration 2	615	416	217	2500
Correction ($615 > N_{thr}$)	0	0	0	2283
Round 2				
Iteration 1	70	44	18	2283
Iteration 2	314	254	194	2283
Iteration 3	921	743	565	2283
Correction	0	0	0	1718
Round 3				
Iteration 1	49	48	47	1718
Iteration 2	654	643	623	1718
Correction	0	0	0	1086
Round 4				
Iteration 1	110	110	110	1086
Iteration 2	712	708	704	1086
Correction	0	0	0	382
Round 5				
Iteration 1	86	86	86	382
Iteration 2	342	342	342	382
Iteration 3	382	382	382	382
Correction	0	0	0	0

with $F(p, t, d) = 0.1$ where Algorithm B still leads to a correct solution. To prevent the attack, even smaller values of $F(p, t, d)$ have to be considered. A definite barrier arises for situations with $F(p, t, d) \leq 0$.

For fixed t and d Table 5 shows the largest p such that $F(p', t, d) \leq 0$ for $p' \leq p$. Note that this p is the smallest correlation probability where the attack may be successful at all (for t and d fixed). If N is smaller than the unicity length Algorithm B may converge to a wrong phase as was pointed out in Section 3.4.

Table 5. p with $F(p, t, d) = 0$.

	t								
d	2	4	6	8	10	12	14	16	18
10	0.584	0.739	0.804	0.841	0.864	0.881	0.894	0.904	0.912
10^2	0.533	0.673	0.750	0.796	0.827	0.849	0.865	0.878	0.890
10^3	0.521	0.648	0.727	0.776	0.809	0.833	0.852	0.866	0.878
10^4	0.514	0.629	0.709	0.760	0.795	0.821	0.841	0.856	0.869
10^5	0.511	0.620	0.699	0.752	0.787	0.815	0.834	0.850	0.863
10^6	0.509	0.612	0.692	0.745	0.782	0.809	0.830	0.846	0.860
10^7	0.508	0.605	0.684	0.738	0.775	0.803	0.825	0.842	0.855
10^8	0.507	0.601	0.680	0.733	0.771	0.800	0.821	0.838	0.852
10^9	0.506	0.597	0.676	0.729	0.768	0.797	0.818	0.836	0.850
10^{10}	0.505	0.592	0.671	0.725	0.764	0.793	0.815	0.832	0.847

Table 5 shows that the attack is definitely infeasible for $t \geq 10$ taps with correlation probabilities occurring in practice, even if the ratio N/k is as large as 10^{10}. Note, however, that for $t = 2$ the limit probability comes very near to 0.5.

5. Conclusions, Generalizations, and Open Problems

The attacks as described in Sections 3 and 4 lead to a new design criterion for general stream ciphers with LFSRs as components: any correlation to an LFSR with less than 10 taps should be avoided.

To face correlation attacks the use of correlation-immune functions has been proposed [4]. The question arises whether our attacks also apply to stream ciphers designed with correlation-immune functions. In principle this is possible in the following way. According to a lemma of Xiao and Massey [7] the output of a correlation-immune function of order k is correlated to a sum of $k + 1$ input variables. The sum of the corresponding LFSR sequences is again an LFSR sequence whose feedback polynomial is the product of the individual feedback polynomials. Thus our algorithms could be applied in order to determine this sum. Hereafter the individual LFSR sequences could be extracted by partial fraction decomposition of the formal power series of the sum. However, the attack is limited by the fact that a product of polynomials rarely has low density even if the factors are trinomials. Thus an appropriate application of correlation-immune functions can prevent attacks based on Algorithms A and B.

The application of our attacks extends to the situation where the feedback polynomial has many terms but is a factor of a low density polynomial of moderate degree. It appears to be difficult to decide whether a given polynomial has this property. However, if the degree k of the polynomial is less than 100, it is feasible to find such polynomial multiples using an algorithm of the following kind.

Let $f(x)$ be an arbitrary polynomial of degree k. Consider two sets of polynomials of the form $x^a + x^b$ and $x^c + x^d$, respectively, with $a, b, c, d \leq 2^{k/4}$. There are $2^{k/2}$ polynomials of both kinds. Using a "common birthday argument," with probability roughly 1/2 there are pairs with $x^a + x^b \equiv x^c + x^d \pmod{f(x)}$. Thus we may find a polynomial multiple of $f(x)$ of degree $2^{k/4}$ whose LFSR has only 3 taps. The computational complexity of this algorithm is of order $O(2^{k/2})$. For example, for $k = 80$ we can find polynomials of degree $2^{20} \approx 10^6$ in $O(2^{40})$ steps. This implies that correlation attacks against LFSRs of length shorter than 100 with arbitrary feedback polynomials are possible, provided that the feedback connection is known.

From this observation another important design criterion results in the sense that there should be no correlation to a general LFSR of length shorter than 100 (especially if the feedback connection is assumed to be known). Our algorithms apply to known plaintext as well as ciphertext-only attacks. In the ciphertext-only situation the correlation probability also depends on the redundancy of the plaintext. Note that in the ASCII code the two most significant bits remain constant when restricted to lowercase letters. A promising approach would consist in decimating the cipher stream by the code length before applying the correlation attack to

the decimated LFSR sequence. If the decimation factor is a power of 2 the feedback polynomial remains unchanged. It follows that in the presence of correlation to an LFSR with few taps the use of 8 bit ASCII code is more problematic than the use of 7 bit ASCII code.

Our analysis can be extended to key stream generators having correlation to components whose output satisfies even nonlinear relations. The important point is not the linearity but the fact that only a few digits are involved in these relations. The statistical model as developed in Section 2 can be generalized to nonlinear relations. The only but essential advantage of linearity is the possibility of producing many relations holding for the same digit (by shifting and iterated squaring).

Our investigations have left an open problem. Besides the first iteration in Algorithm B we could not explain the surprising success of this attack. In mathematical terms the assignment of new probabilities defines a map P^*: $[0, 1]^N \to [0, 1]^N$, given by formulas (2.10) and (4.7) for the conditional probability. Steps 4 and 5 in Algorithm B consist of iterations of P^*. Under suitable conditions as indicated in Section 4, our experiments have shown that for a given initial vector $\mathbf{p} = (p, p, \ldots, p)$, $p > 0.5$, the iterated image of $\mathbf{p}$ converges, apart from a few components, to a fixed point which is a vertex of the N-cube. It remains to explain this behavior. For such an explanation step 6 has to be taken into account, where the digits are complemented at those positions where the corresponding component of the fixed point is zero. Moreover, it is unclear why Algorithm B can lead to a solution after several rounds.

The solution of the correlation problem may be viewed as the decoding of LFSR codes, and thus can be studied in terms of coding theory. In fact, James L. Massey has pointed out to us that iterative methods similar to our Algorithm B have been applied in decoding. In [2] Gallager has developed a decoding scheme, where the decoder computes all the parity checks and then changes any digit that is contained in more than some fixed number of unsatisfied parity-check equations. Using these new values the parity checks are recomputed, and the process is repeated. Our method contrasts to this approach in that we iterate the process of assigning conditional probabilities to every digit rather than changing digits according to the number of parity-check equations satisfied.

Acknowledgment

We wish to thank James L. Massey from the Swiss Federal Institute of Technology and Paul Schoebi from GRETAG for their interest and criticism, as well as Ernest F. Brickell for his suggestions to improve the final version of the paper.

References

[1] W. Blaser and P. Heinzmann, New cryptographic device with high security using public key distribution, *IEEE Student Papers*, 1982, pp. 145–153.

[2] R. G. Gallager, *Low-Density Parity-Check Codes*, MIT Press, Cambridge, MA, 1963.

[3] R. A. Rueppel, *Analysis and Design of Stream Ciphers*, Springer-Verlag, New York, 1986.

[4] T. Siegenthaler, Correlation-immunity of nonlinear combining functions for cryptographic applications, *IEEE Trans. Inform. Theory*, **30**, 776–780, 1984.
[5] T. Siegenthaler, Decrypting a class of stream ciphers using ciphertext only, *IEEE Trans. Comput.*, **34**, 81–85, 1985.
[6] J. H. van Lint, *Introduction to Coding Theory*, Graduate Texts in Mathematics, Vol. 86, Springer-Verlag, Berlin, 1982
[7] G. Z. Xiao and J. L. Massey, A spectral characterizaton of correlation-immune combining functions, *IEEE Trans. Inform. Theory*, **34**, 569–571, 1988.