Predicting the Shrinking Generator with Fixed Connections

Patrik Ekdahl[1], Willi Meier[2], and Thomas Johansson[1]

[1] Dept. of Information Technology
Lund University, P.O. Box 118
SE-221 00, Lund, Sweden
{patrik,thomas}@it.lth.se
[2] FH Aargau,
CH-5210 Windisch, Switzerland
meierw@fh-aargau.ch

Abstract. We propose a novel distinguishing attack on the shrinking generator with known feedback polynomial for the generating LFSR. The attack can e.g. reliably distinguish a shrinking generator with a weight 4 polynomial of degree as large as 10000, using 2^{32} output bits. As the feedback polynomial of an arbitrary LFSR is known to have a polynomial multiple of low weight, our distinguisher applies to arbitrary shrunken LFSR's of moderate length. The analysis can also be used to predict the distribution of blocks in the generated keystream.

1 Introduction

The shrinking generator (SG) is a well known pseudo random generator proposed in 1993 by Coppersmith, Krawczyk and Mansour [3]. It is intended for use in stream cipher applications and is of interest by its conceptual simplicity: It combines only two linear feedback shift registers (LFSR's) in a simple way. Thus far, it has resisted against efficient cryptanalysis.

In this paper, a practical distinguishing attack on the (SG) is proposed. The attack exploits a newly detected non-randomness in the distribution of output blocks in the generated keystream.

Recall that the (SG) [3] uses two independent binary LFSRs, say A and S, as basic components. The pseudo random bits are produced by shrinking the output sequence of the generating LFSR A under the control of the selecting LFSR S as follows: The output bit of LFSR A is taken if the current output bit of LFSR S is 1, otherwise it is discarded. It is recommended in [3] that besides the initial states also the feedback polynomials be defined by the secret key. We would like to stress that the present attack is not applicable to a (SG) with unknown feedback polynomial for the generating LFSR.

There have been known several approaches for attacking the (SG): A basic divide-and-conquer attack, which requires an exhaustive search through all possible initial states and feedback polynomials of S, [3]. On the other hand, a correlation attack targeting LFSR A is proposed in [6] and is experimentally analyzed in [16]. It requires exhaustive search through all initial states and all possible feedback polynomials of LFSR A. In [12], a reduced complexity correlation attack based on searching for specific subsequences of the output sequence has been given, where the complexity and required keystream length are exponential in the length of LFSR A.

It has been shown in [7, 8], that the output sequence has a detectable linear statistical weakness if the feedback polynomial of the generating LFSR A has very low weight and moderate degree, or is known to have polynomial multiples with this property.

For our attack we assume the feedback connection of the generating LFSR A to be known, whereas the selecting sequence can be any random sequence with independent and equally distributed bit probabilities.

To describe our approach, denote by *a-stream* the output sequence generated by LFSR A and by *z-stream* the output sequence generated by the (SG). The bits in the a-stream are denoted $a_n, n \geq 0$ and the bits in the z-stream are denoted $z_i, i \geq 0$. Rather than single bits, we consider bit strings (blocks) in the a-stream and compare them with suitable strings in the z-stream.

Consider a block of odd length, centered at the position of a_n in the a-stream. If this block is xor-ed with similar blocks (of equal length) centered at all other tap positions (including the feedback position) in the LFSR-recursion, the sum is (trivially) the all-zero block. For blocks of odd length, the majority bit is set 1 if the number of ones is larger than the number of zeroes, and 0 otherwise. Then a key observation is that the majority bits of such blocks fulfill the linear recursion of LFSR A more often than random.

Through the shrinking process, the exact positions of the undiscarded a-stream bits in the z-stream get lost. However, the deletion rate is 1/2, and we can guess averaged *shrunken tap positions* by halving the distances of the tap positions in the a-stream.

The main idea of our attack is to consider samples in the z-stream where the blocks near the shrunken tap positions are all imbalanced (a block is imbalanced if it has a different number of ones than zeroes). The bits in these blocks have with high probability been generated by bits in the neighborhood of the original tap positions. If a z-block (i.e., a block in the z-stream) with high imbalance is found, the probability that the corresponding a-block is imbalanced, is quite high. By estimating the imbalance in the blocks in the a-stream using the measured imbalance in the blocks in the z-stream, we can estimate the majority bit of the a-blocks.

If these estimated majority bits fulfill the linear recursion of LFSR A more often than random, we can distinguish the z-stream generated by the (SG) from a truly random sequence. This is indeed the case: Theoretical estimates as well as extensive experiments have shown that we are able to reliably distinguish the (SG) from random for low-weight recursions of LFSR A. E. g., for a weight 4 recursion of length 10000, the attack needs about 2^{32} output bits, and for a weight 3 recursion of length 40000, the attack needs about 2^{23} output bits to reliably distinguish the (SG). This compares nicely with the estimates given for

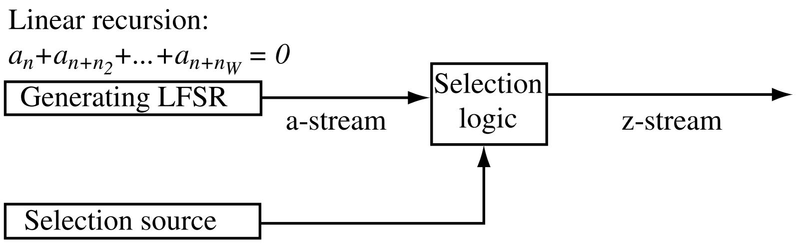

Fig. 1. Model of the shrinking generator used in the attack.

the distinguishers as described in [7, 8], both of which for these parameters needs about 2^{48} output bits, and 2^{39} output bits, respectively.

As the feedback polynomial of an arbitrary LFSR of length L is known to have a polynomial multiple of weight 4 and length about $2^{L/3}$, [1, 9, 13, 17] our distinguisher also applies to arbitrary shrunken LFSR-sequences of moderate length.

The paper is organized as follows. In Section 2 we present the attack. In Section 3 we analyze the proposed attack in terms of probability of success and required length of observed keystream. Simulation results are presented in Section 4, and some concluding remarks are given in Section 5. Finally, the technical details are presented in Appendix A.

2 Description of the attack

For the description of the attack, we assume a (SG) with a weight W feedback polynomial generating the a_n sequence. See Fig. 1. The known recurrence equation is given by

$$a_n + a_{n+n_2} + a_{n+n_3} + \ldots + a_{n+n_W} = 0, \quad n \geq 0. \tag{1}$$

We do not distinguish the feedback position from the other tap positions, so for a certain bit a_n, the tap positions are $n, n+n_2, n+n_3, \ldots, n+n_W$. Now, consider a bit string (block) surrounding a position a_n in the a-stream. If we xor this block with the blocks (of equal length) surrounding the other positions in the LFSR recurrence equation, $a_{n+n_2}, \ldots, a_{n+n_W}$, the sum (trivially) is the all-zero block. If we choose the considered block length to be odd, each of the W blocks must have a unique *majority bit.* The majority bit is 1 if the number of ones in the block is larger than the number of zeros, and vice versa. The main observation is, that the majority bits of such blocks fulfill the linear recurrence equation of the LFSR, with a probability larger than 1/2. First we formally introduce the intuitive notion of *imbalance.*

Definition 1. *The imbalance of a block B, $Imb(B)$, is defined as*

$$\textit{number of 1's in } B - \textit{number of 0's in } B$$

Furthermore, a block B is said to be imbalanced if $Imb(B) \neq 0$.

The general idea is to search for imbalanced blocks in the z-stream at positions $z_{i_1}, z_{i_1+n_2/2}, \ldots, z_{i_1+n_W/2}$. These positions are called the shrunken tap positions, where i_1 implicitly is the reference position. The bits in these blocks have with high probability been generated by bits in the neighborhood of the unshrunken tap positions, $a_{n_1}, a_{n_1+n_2}, \ldots, a_{n_1+n_W}$, where w.l.o.g. we assume $a_{n_1} = z_{i_1}$. If we find a block with high imbalance in the z-stream, then the probability that the corresponding block in the a-stream is imbalanced, is quite high. Then we can derive good approximations of the true majority bits in the a-blocks.

The attack has two phases. The first phase is a search for positions in the output sequence, where we find imbalanced blocks centered around every of the shrunken taps. Whenever we find such a position we say we have a **hit**, and invoke the second phase, which estimates the majority bits of the unshrunken segments. We then count the number of times the xor sum of the estimated majority bits equal zero, and compare with the truly random case.

2.1 First phase

Pick a block B_1 of odd length $BL_1 = E+1$ centered around a reference position z_{i_1}, where E is an *even* parameter to the attack. As we assume $z_{i_1} = a_{n_1}$, the bits in B_1 come from bits surrounding a_{n_1}.

The next unshrunken tap $a_{n_1+n_2}$ is with high probability mapped to an interval near $i_1+n_2/2$. This interval size grows proportionally to $\sqrt{n_2}$. The same holds for the other taps. Thus at tap position $n_j, j = 2, \ldots, W$, we measure on a block centered at position $i_1 + n_j/2$ of length $BL_j \approx BL_1 + \sqrt{n_j}/2$. The $\approx$ symbol denotes here: "take the closest odd integer", since we need an odd length for having a unique imbalance.

Next, we measure the imbalance in each block, $Imb(B_j), j = 1 \ldots W$. Whenever we have $|Imb(B_j)| > T, j = 1 \ldots W$, where T is an imbalance threshold, we have found a *hit*, and it is likely that the a-blocks surrounding the unshrunken taps, also are imbalanced. If we do not find imbalanced z-blocks at the chosen reference position we pick a new z_{i_1} and again measure the imbalance. However, if we have a *hit* in phase 1, we invoke the second phase.

2.2 Second phase

The first goal of the second phase is to try to estimate the bit probability of the a-blocks. First we introduce some notations. Let S_1 be the a-block of length L_a surrounding a_{n_1}. Similarly, we denote by $S_j, j = 2, \ldots, W$ the a-block of length L_a surrounding $a_{n_1+n_j}$. Denote by $p_j, j = 1, \ldots, W$ the estimated probability that a bit in S_j equals 1.

Since we assumed that the center bit $z_{i_1} = a_{n_1}$, we will use the first block as a reference. The first z-block is of length $E+1$. We denote this reference block length by $L_z = E+1$. For our considerations we can assume that the z-block B_1 of odd length, was produced by an odd length a-block. Thus we choose the a-block length as $L_a = 2L_z - 1 = 2E+1$.

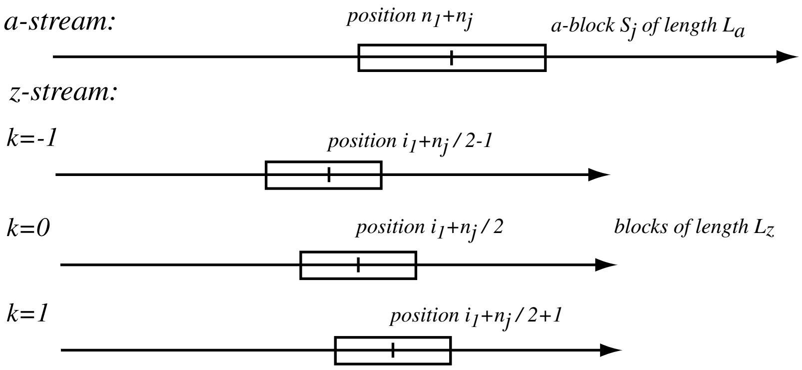

Fig. 2. Picture of which blocks are used in the weighting of estimating $p_j, j = 2, \ldots, W$. The interval size shown here is only 3 positions.

If we measure the imbalance $Imb(B_1)$ in B_1, and assume that the bits which where discarded are balanced, i.e., the $L_a - L_z = E$ bits not visible as output bits are equally distributed, we have the following estimate

$$p_1 = \frac{\text{number of ones in } S_1}{L_a} = \frac{(Imb(B_1) + L_z)/2 + E/2}{L_a} \tag{2}$$

$$= \frac{1}{2} + \frac{Imb(B_1)}{4E+2}. \tag{3}$$

For the other $W - 1$ taps we proceed in a slightly different way. Since we are not sure which z-block contains the most bits from S_j, we must consider several z-blocks centered near $z_{i_1+n_j/2}$. Thus, we pick an interval $\mathcal{I}_j$ and calculate a weighted average over $\mathcal{I}_j$, of the bit probability in S_j as

$$p_j = \frac{1}{2} + \sum_{k \in \mathcal{I}_j} \frac{imb_{j,k}}{4E+2} P(k \text{ is the best position for estimating } S_j) \quad j = 2 \ldots W, \tag{4}$$

where $imb_{j,k}$ is the imbalance in the z-block of length L_z surrounding $z_{i_1+n_j/2+k}$. The expression for $P(k \text{ is the best} \ldots)$ is discussed in Appendix A, but the aim is to try to derive a probability that the z-block at $i_1 + n_j/2 + k$ has the most bits from S_j, and we assume that the binomial distribution

$$P(k \text{ is the best position for estimating } S_j) = Bin[n_j, 0.5](\frac{n_j}{2} + k) \tag{5}$$

is adequate. Fig. 2 shows a picture of the blocks used in the weighting process for a small interval of $k \in [-1, 0, 1]$.

In (4), we have used similar calculations as in (2) and the additional assumption that the bit probability is 0.5 if the best position, k, is outside the interval $\mathcal{I}_j$.

The estimated bit probability $p_j, j = 1, \ldots, W$ also represent an estimate of the majority bit for S_j. For each hit we have in phase 1 of the attack, we can determine an estimate of the xor sum of the majority bits. The simplest decision rule is to first make a hard decision on the estimated majority bit $\hat{m}_j$ as

$$\hat{m}_j = \begin{cases} 1 & \text{if } p_j \geq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}, \quad j = 1, \ldots, W, \tag{6}$$

then check if the xor sum $\sum_j \hat{m}_j = 0$ holds. By dividing the number of times the xor sum equals zero with the number of hits we have, we can derive a measured final probability that the sum of the true majority bits equals zero.

We can also employ a soft decision rule. It is well known [11] that a soft decision rule is no worse than a hard decision rule, so employing a soft decision rule can only improve the attack. The hard decision rule is however simpler to analyze, and for that reason we have use a hard decision rule in the analysis and in our simulations.

If we indeed are attacking the (SG), we expect the total probability

$$P = 0.5 + \varepsilon_H, \tag{7}$$

where ε_H is a positive value depending on the number H of hits we get in phase 1. If we are attacking a true random sequence, we expect $P = 0.5$. The optimal test to distinguishing these cases [4] is a Maximum Likelihood (ML) test, where we decide using a threshold Γ such that if $P \geq \Gamma$ we decide the output sequence to be generated from the (SG), otherwise we decide it is a random source. We have summarized the proposed attack in Fig. 3.

3 Analysis of the proposed attack

We start by considering the probability that the majority bits of $S_j, j = 1, \ldots, W$ fulfill the recurrence equation. We will interchangeably be using imbalance and *Hamming weight* (Hw) as they measure similar quantities, where the Hamming weight of a block B is the number of ones in B. For odd block length L we have the conversion between imbalance *imb* and Hamming weight *hw* as:

$$imb = 2hw - L,$$
$$hw = \frac{imb + L}{2}.$$

3.1 The probability of the sum of the majority bits

Assume we look at the direct output of a LFSR with W taps (including the feedback position). Pick a reference position a_n in the a-stream, and consider the positions $a_{n+n_2}, \ldots, a_{n+n_W}$ that together with a_n sum to zero, according to the linear recurrence equation of the LFSR. At each position we take a centered block (or vector) S_j of odd length L_a. The positions and the length are assumed to be chosen such that the blocks are non-overlapping. We aim to calculate the

Input: even integer E, imbalance threshold T, the tap positions $n_j, j = 1, \ldots, W$, where $n_1 = 0$, and a sequence $Z_i, i = 1, \ldots, N$ of received bits.

1. Setup variables.

$$\begin{aligned} BL_1 &= E + 1, \\ BL_j &= \text{ closest odd integer } (BL_0 + \sqrt{n_j}/2), \quad j = 2, \ldots, W, \\ \mathcal{I}_j &= \text{ integers in } [-\sqrt{n_j}/3, \ldots, \sqrt{n_j}/3], \quad j = 2, \ldots, W \\ low &= E/2, high = N - (n_W/2 + \sqrt{n_W}/3 + E/2), \\ hits &= 0, good = 0. \end{aligned}$$

2. For $i_1 = low$ to $high$ do
   (a) For all $j = 1, \ldots, W$, let B_j be the z-block centered at $i_1 + n_j/2$ of length BL_j.
   (b) Let $imb_j = Imb(B_j), j = 1, \ldots, W$.
   (c) If $\forall j \; |imb_j| > T$ then
      i. Increase $hits$ and set

$$\begin{aligned} p_1 &= \frac{1}{2} + \frac{imb_1}{4E+2} \\ p_j &= \frac{1}{2} + \sum_{k \in \mathcal{I}_j} \frac{imb_{j,k}}{4E+2} \binom{n_j}{n_j/2 + k} (\frac{1}{2})^{n_j}, j = 2, \ldots, W, \end{aligned}$$

      where $imb_{j,k}$ is the measured imbalance in the z-block of length BL_1 centered at position $i_1 + n_j/2 + k$.
      ii. Make a hard decision $\forall j, \hat{m}_j = 1$ if $p_j \geq 0.5$ otherwise $\hat{m}_j = 0$.
      iii. if $\sum_j \hat{m}_j = 0$ increase $good$. ($\sum$ denotes here the xor sum.)
3. Set $\Gamma = 0.5 + \sqrt{hits}/(2hits)$.
4. If $good/hits > \Gamma$ return **Shrinking** else return **Random**.

Fig. 3. A summary of the proposed attack using a hard decision rule.

probability that the majority bits of the segments fulfill the recurrence equation of the LFSR. We assume the distribution of the possible vectors at tap $j, j = 1, \ldots, W-1$, only to be dependent on the Hw of the vector. The vector at the final tap is totally determined by the choices at the $W-1$ previous taps since the xor of the vectors must be the all-zero vector. Introduce the notation

$$V_j(\alpha) = P(H(S_j) = \alpha), \tag{8}$$

for the probability that we have a vector of length L_a and Hw α at tap $j, j = 1, \ldots, W-1$.

We can partition the possible vectors at the taps into sets of equal Hamming weight. The probability of each set is given by $V_j(\alpha)$. When we xor the first two vectors, we get a new probability distribution on the sets of different Hamming weight. In these sets, the majority bit of the vector sum may or may not agree with the sum of the majority bits of the xored vectors. So when we derive the

L_a	Simulation	Theory
5	0.56082 0.56050	0.56054
11	0.52254 0.52169	0.522076
17	0.51372 0.51298	0.51340
29	0.50764 0.50727	0.50748

Table 1. Simulation results versus theory for the probability that the majority bits sum to zero.

new distribution, we also keep track of when the sum of the constituent majority bits agree with the majority bit of the resulting vector. When we have xored the $W-1$ vectors, we know that the final vector *must* agree with the sum of the first $W-1$ vectors. Thus we can sum up the probabilities where the sum of the constituent majority bits agree with the majority bit of the resulting vector.

The technical part of the calculations can be found in Appendix A. We only conclude the validity of the approach by comparing the theoretical probability to a number of simulations for different block length L_a with $W = 4$. The comparison is shown in Table 1. The number of trials in the simulations were 10 000 000.

3.2 Skewing the distributions V_j

Since in phase 1 of the attack we impose the condition that the absolute value of the imbalance in the z-block should be greater than a certain threshold T, the distributions V_j for the a-blocks will not be binomial, as expected from a random source. As the probabilities $P(Imb(B_j) > T) = 0.5$ and $P(Imb(B_j) < -T) = 0.5$ are equally likely, we divide the distribution V_j into two parts.

$$V_j(\alpha) = \frac{V_j^+(\alpha) + V_j^-(\alpha)}{2}, \quad j = 1, \ldots, W \tag{9}$$

where

$$V_j^+(\alpha) = P(H(S_j) = \alpha | Imb(B_j) > T), \tag{10}$$

$$V_j^-(\alpha) = P(H(S_j) = \alpha | Imb(B_j) < -T), \tag{11}$$

and $H(S_j)$ is the Hw of the a-block S_j.

We can not hope to give an exact expression for (10) and (11) since there are too many dependencies between adjacent $imb_{j,k}$ in (4). We can however derive approximate expressions. The derivation is omitted due to space limitations, but can be found in an extended version of this paper available from the authors. Intuitively, and also from the derived approximations, we note that V_j^+ and V_j^- are "mirrored" distributions in the sense that $V_j^+(\alpha) = V_j^-(L_a - \alpha)$.

3.3 An approximation on the required number of hits

First, we consider the probability that the estimated majority bit $\hat{m}_j$ is correct. We will confine us to the case of a hard decision rule. Through simulations we have noted that these probabilities are not independent. The first $W-1$ majority bits are more often correctly predicted if all or most of the $W-1$ a-block are strongly imbalanced. But then also the W's block tends to have some imbalance and thus gives better prediction of the majority bit. In the analysis we will however assume that they are independent.

Assume without loss of generality that $m_j = 1$. Then we know that the vectors of S_j are drawn (on the average) from the distribution V_j^+. The probability P_{m_j} that m_j is correct is determined by the probability mass that gives a majority bit equal to 1. We have

$$P_{m_j} = \sum_{\alpha=(L_a+1)/2}^{L_a} V_j^+(\alpha), \quad j = 1, \ldots, W. \tag{12}$$

The same holds if we have $m_j = 0$ because of the mirrored properties of V_j^+ and V_j^-.

Let m_j be the true majority bit of the a-block S_j, and as before, let $\hat{m}_j$ be the hard decision estimate as derived in the attack. We have $P(m_j = \hat{m}_j) = P_{m_j}$. We can model this as if we have a noisy observation of the true majority bits, where the noise for each observation is equal to 0 with probability P_{m_j}. Furthermore, let P_n denote the probability that noise variables sum to zero. Using the independency assumption and the *piling-up-lemma* [14] we have

$$P_n = \frac{1}{2} + 2^{W-1} \prod_{j=1}^{W} (P_{m_j} - 0.5). \tag{13}$$

The correctness of the estimates $\hat{m}_j, j = 1, \ldots, W$ are assumed independent of the probability $P(\sum_j m_j = 0)$. Recalling (7) and introducing the notation P_M for the probability that the true majority bits sum to zero, we have an approximation of the total probability P, that the estimated majority bits sum to zero:

$$P = \frac{1}{2} + 2(P_M - 0.5)(P_n - 0.5) = \frac{1}{2} + \varepsilon_H. \tag{14}$$

Using similar arguments as in [2, 5], we can state an upper bound on the number of hits H we need in phase 1 to distinguish the (SG) from a truly random sequence as:

$$H \leq \frac{1}{\varepsilon_H^2}. \tag{15}$$

3.4 A lower bound on the expected number of hits

In phase 1 of the attack, we search for positions in the z-stream where we find W blocks $B_j, j = 1, \ldots, W$ of length $BL_j, j = 1, \ldots, W$, such that we simultaneously have $|Imb(B_j)| > T$. If we again make the simplifying assumption that the imbalance (or Hamming weight) of the blocks are independent, we can derive a lower bound on the expected number of hits in phase 1, given a received sequence of length N from the generator.

The probability P_{B_j} that we have $|Imb(B_j)| > T$ for a block of length BL_j where we assume the keystream bits from the generator to be independent and random is given by

$$P_{B_j} = \sum_{h=1+\frac{T+BL_j}{2}}^{BL_j} \binom{BL_j}{h} \left(\frac{1}{2}\right)^{BL_j} + \sum_{h=0}^{\frac{-T+BL_j}{2}-1} \binom{BL_j}{h} \left(\frac{1}{2}\right)^{BL_j}. \qquad (16)$$

The approximated joint probability of finding W blocks is given by $P_H = \prod_{j=1}^{W} P_{B_j}$ and thus we would expect

$$H \geq P_H N. \qquad (17)$$

3.5 Parameters trade-offs

If we choose a large value for the parameter E we will have larger blocks B_j and the probability for the bit a_{n_j} to be mapped inside B_j increases. On the other hand the a-blocks will also be larger, resulting in a smaller probability for the true majority bits summing to zero. If we only choose the lengths BL_j larger then the distribution for the smaller block which is scanned in (4) tends to be more binomial, which gives us less non-randomness to extract information from.

If we choose a high imbalance threshold T, we will get more skewed distributions V_j and we will need a fewer number of hits to distinguish the sequence. But on the other hand, the probability for getting a hit in phase 1 decreases, thus requiring a longer received sequence.

4 Simulation results

In this section we present some simulation results and do a comparison with the derived theoretical approximation. In the attack we have used the hard decision rule and a decision threshold $\Gamma = 0.5 + \sqrt{hits}/(2hits)$, which corresponds to about one standard deviation from the expected value if the sequence were truly random. We start by attacking a weight 4 LFSR given by the linear recurrence equation

$$a_n + a_{n+302} + a_{n+703} + a_{n+1000} = 0. \qquad (18)$$

Using a threshold $T = 3$ and block size parameter $E = 14$, the theoretical calculations (14) and (16) give $P_H = 0.02648$ and $\varepsilon_H = 0.00092$. Thus we would need about $H = 1/\varepsilon_H^2 \approx 2^{20}$ hits in phase 1 and $N = H/P_H \approx 2^{25}$ received output bits. Running the attack with these configurations 50 times (with random initial state), we can distinguish the (SG) in all 50 cases.

As the degree of the feedback polynomial increases, the attack naturally needs a larger received sequence, and it becomes impractical to simulate the attack.

Tap positions (excluding 0)	Theoretic parameters P_H	ε_H	H	N	N used in attack	Successes out of 50 runs
$302, 733, 1000$	0.02648	$2^{-10.1}$	$2^{20.2}$	$2^{25.4}$	2^{23}	43
					2^{24}	46
					2^{25}	50
$812, 1433, 2500$	0.03586	$2^{-11.5}$	$2^{23.0}$	$2^{27.8}$	2^{25}	39
					2^{26}	46
					2^{27}	50
$2333, 5847, 8000$	0.05542	$2^{-13.5}$	$2^{27.0}$	$2^{31.2}$	2^{28}	42
					2^{29}	48
					2^{30}	50
$3097, 6711, 10000$	0.05989	$2^{-13.9}$	$2^{27.7}$	$2^{31.8}$	2^{28}	45
					2^{29}	45
					2^{30}	46

Table 2. Theoretic results and simulation results of attacking the (SG) with some weight 4 feedback polynomials

Tap positions (excluding 0)	Theoretic parameters P_H	ε_H	H	N	N used in attack	Successes out of 50 runs
$17983, 40000$	0.1414	$2^{-10.2}$	$2^{20.3}$	$2^{23.1}$	2^{21}	36
					2^{22}	46
					2^{23}	50
$73, 131, 219, 300$	0.0068	$2^{-11.56}$	$2^{23.1}$	$2^{30.3}$	2^{29}	48
					2^{30}	50

Table 3. Theoretic results and simulation results of attacking the (SG) with weight 3 and weight 5 feedback polynomial

Also, the approximations made in the analysis tends to overestimate the required N, as experiments with shorter sequences have shown. We have summarized some results on attacking weight 4 feedback polynomials in Table 2. We also give two examples of attacking a (SG) with weight 3 and weight 5 feedback polynomial in Table 3.

The computational complexity of the attack is quite modest. If we use precomputed tables for (5) we see that we need to scan the input sequence once, and whenever we have a *hit* we calculate (4). The size of the interval $\mathcal{I}_j$ is proportional to $\sqrt{n_j}$ and hence we have the computational complexity $O(N\sqrt{n_W})$, where N is the number of received output bits and n_W is the last tap position or the degree of the feedback polynomial.

5 Conclusions

We have proposed a practical distinguishing attack on the shrinking generator with a low-weight feedback polynomial for the generating LFSR. The new idea is that the majority bits of blocks in the LFSR stream fulfill the linear recursion more often than random, and we have presented and analyzed a quite powerful method of estimating those bits, based on the received sequence.

We do not need to have the first segment as reference segment. We could also use a tap in the middle and calculate both forward and backward, thus increasing the probabilities of the estimates.

The proposed attack can also be used to predict the distribution of bits in the generated sequence. Assume we know that the generating source is the (SG). First we calculate the bit probability of the $W-1$ first segments. By the theoretical calculations of the probability that the majority bits sum to zero we can derive the unknown distribution of the last segment.

The authors also would like to thank Vishal Chandra, who helped simulating some early ideas for an attack, and the anonymous referees for giving valuable comments on how to improve the manuscript.

References

1. A. Canteaut, M. Trabbia, "Improved Fast Correlation Attacks Using Parity-Check Equations of Weight 4 and 5", *Advances in Cryptology – EUROCRYPT 2000*, LNCS vol 1807, Springer-Verlag, 2000, pp. 573-588.
2. D. Coppersmith, S. Halevi, C. Jutla, "Cryptanalysis of stream ciphers with linear masking", *Advances in Cryptology - CRYPTO 2002*, LNCS vol 2442, Springer, 2002, pp 515-532.
3. D. Coppersmith, H. Krawczyk, Y. Mansour, "The Shrinking Generator", *Advances in Cryptology - Crypto'93*, LNCS vol 773, Springer-Verlag, 1994, pp 22-39.
4. T. Cover, J. Thomas, *Elements of Information Theory*, Wiley series in telecommunications, Wiley-interscience, 1991.
5. P. Ekdahl, T. Johansson, "Distinguishing attacks on SOBER", *Fast Software Encryption (FSE) 2002*, LNCS vol 2365, Springer 2002, 210-224.
6. J. Dj. Golić, L. O'Connor, "Embedding and probabilistic correlation attacks on clock-controlled shift registers", *Advances in Cryptology – EUROCRYPT'94*, LNCS vol 950, Springer-Verlag, 1995, pp. 230-243.
7. J. Dj. Golić, "Towards Fast Correlation Attacks on Irregularly Clocked Shift Registers", *Advances in Cryptology – EUROCRYPT'95*, LNCS vol 921, Springer-Verlag, 1995, pp. 248-262.
8. J. Dj. Golić, "Linear Models for Keystream Generators", *IEEE Trans. on Computers*, vol 45, No 1 January, IEEE Press, 1996, pp. 41-49.
9. J. Dj. Golić, "Computation of low-weight parity-check polynomials", *Electronic Letters*, Vol. 32, No 21 October, 1996.
10. J. Dj. Golić, "Correlation analysis of the Shrinking Generator", *Advances in Cryptology – CRYPTO 2001*, LNCS vol 2139, Springer-Verlag, 2001, pp. 440-457.
11. R. Johannesson, K. Sh. Zigangirov, *Fundamentals of convolutional coding*, IEEE Series on digital and mobile communication, IEEE Press, 1999, p 5.

12. T. Johansson, "Reduced complexity correlation attacks on two clock-controlled generators", *Advances in Cryptology – ASIACRYPT'98*, LNCS vol 1514, Springer-Verlag, 1998, pp. 342-357.
13. T. Johansson, F. Jönsson, "Fast Correlation Attacks Through Reconstruction of Linear Polynomials", *Advances in Cryptology – CRYPTO 2000*, LNCS vol 1880, Springer-Verlag, 2000, pp. 300-315.
14. M. Matsui, "Linear cryptanalysis method for DES cipher, *Andvances in Cryptology – EUROCRYPT'93*, LNCS vol 765, Springer-Verlag, 1994, pp. 386-397.
15. A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
16. L. Simpson, J. Dj. Golić, E. Dawson, "A probabilistic correlation attack on the shrinking generator", *Information Security and Privacy'98 - Brisbane*, LNCS vol 1438, Springer-Verlag, 1998, pp. 147-158.
17. D. Wagner, "A Generalized Birthday Problem", *Advances in Cryptology – CRYPTO 2002*, LNCS vol 2442, Springer-Verlag, 2002, pp. 288-303.

Appendix A

In this Appendix we have gathered some of the more technical details in the proposed attack and the analysis of the attack.

Some details of phase 2

The interval size in (4) is a tunable parameter to the attack and we have chosen it to be the integers in the range $[-\sqrt{n_j}/3, \ldots, \sqrt{n_j}/3]$. Next we consider the probability that the z-block centered at $z_{i_1+n_j/2+k}, k \in \mathcal{I}_j$, is the best block to choose. An exact mathematical expression for this notion is hard to find because of the deletion process. To simplify the calculations we assume that this probability is given by the binomial distribution, $Bin[n_j, 0.5](n_j/2 + k)$. This assumption does not take into account that the bit $a_{n_1+n_j}$ might not be visible (printed) in the z-stream at all, since it might be deleted. However, we can disregard whether the bit is printed or not since we are only trying to estimate the surrounding of $a_{n_1+n_j}$. Thus, the assumed distribution is reasonably adequate, and we write

$$P(k \text{ is the best position for estimating } S_j) = P(k \text{ is best}) = \\ Bin[n_j, 0.5](\frac{n_j}{2} + k) = \binom{n_j}{\frac{n_j}{2} + k} \left(\frac{1}{2}\right)^{n_j} \qquad (19)$$

We note that the interval $\mathcal{I}_j$ is chosen such that $\sum_{k \in \mathcal{I}_j} P(k \text{ is best}) \approx 0.5$.

Probability that the sum of the majority bits equal zero

We will assume that the bits in the first $W-1$ vectors in the LFSR case can be approximated by the true random case. Given this assumption, the probability V_j is given by the binomial distribution

$$V_j(\alpha) = Bin[L_a, 0.5](\alpha), \quad i = 1 \ldots W-1. \qquad (20)$$

When xoring vectors together, we will denote the distribution of vectors (for the sum) by $Q_j(\alpha)$. We will add a super script to this Q later, but for now on, we say that $Q_1(\alpha) = V_1(\alpha), \alpha = 0 \dots L_a$, since we have not added any vectors but the first. $Q_2(\alpha)$ will be the probability of a vector of Hw α when we have xored two V distributions. We also introduce an operator to determine the majority bit. Let $Maj(\alpha)$ denote the majority bit of a vector of length L_a and Hw α. We have

$$Maj(\alpha) = \begin{cases} 1 \text{ if } \alpha > (L_a - 1)/2, \\ 0 \text{ otherwise.} \end{cases}$$

The next lemma states what Hw's to expect, and how many of them we get, when xoring two vectors together.

Lemma 1. *Let A be a fixed vector of length L_a and Hw α. If we xor all possible vectors (one at a time) of Hw β with A, then the possible Hw's of the xor sum are $\gamma = \alpha - \beta + 2\kappa$ where $max(0, \beta - \alpha) \leq \kappa \leq min(L_a - \alpha, \beta)$. The number of resulting vectors with Hw γ is given by*

$$\binom{\alpha}{\beta - \kappa}\binom{L_a - \alpha}{\kappa}. \tag{21}$$

Proof. Denote by **B** the set of all vectors with Hw equal to β. Assume κ of the 1's in **B** coincide with the 0's of A. This implies that $\beta - \kappa$ of the 1's in **B** coincide with the 1's of A. The number of such vectors in **B** is

$$\binom{\alpha}{\beta - \kappa}\binom{L_a - \alpha}{\kappa}.$$

For the choice of κ we must have $0 \leq \beta - \kappa \leq \alpha$ or equivalently $\alpha - \beta \leq \kappa \leq \beta$, since at least zero 1's and at most α of the 1's can coincide. Similarly, we must have $0 \leq \kappa \leq L_a - \alpha$, since the number of 0's in A is $L_a - \alpha$. Combining these restrictions we get $max(0, \alpha - \beta) \leq \kappa \leq min(L_a - \alpha, \beta)$. The resulting Hw of the xor sum is $\gamma = \alpha - (\beta - \kappa) + \kappa = \alpha - \beta + 2\kappa$, which proves the lemma.

Thus, if we pick *one* vector B of Hw β uniformly at random (over the set of vectors with Hw β) and xor with the fixed vector A, we have the probability

$$P_\gamma(\alpha, \beta, \kappa) = \frac{\binom{\alpha}{\beta - \kappa}\binom{L_a - \alpha}{\kappa}}{\binom{L_a}{\beta}}, \tag{22}$$

of obtaining a resulting vector of Hw $\gamma = \alpha - \beta + 2\kappa$, where $max(0, \beta - \alpha) \leq \alpha \leq min(L_a - \alpha, \beta)$.

When xoring two vectors, we must keep track if the xor of the majority bits agrees with the majority bit of the sum. For instance, we can get the vector 11110 by

Vector	H-weight	Maj. bit		Vector	H-weight	Maj. bit
11000	2	0	or by	11100	3	1
⊕00110	2	0		⊕00010	1	0
11110	4	1		11110	4	1

We can determine this condition by checking if $Maj(\alpha) \oplus Maj(\beta) = Maj(\gamma)$, and we will save the corresponding probability mass in different variables, $Q_j^0(\alpha)$ for when the sum of the constituent majority bits agrees with the majority bit of the sum, and $Q_j^1(\alpha)$ for when it does not. One can think of the super script as a parity bit for making the sum (of the constituent majority bits and the resulting majority bit) equal to zero.

Recalling (22) and the fact that we defined $Q_1^0(\alpha) = V_1(\alpha), \alpha = 0, \ldots, L_a$ we have for the sum of $j = 2 \ldots W-1$ vectors:
If $Maj(\alpha) \oplus Maj(\beta) = Maj(\gamma)$,

$$Q_j^0(\gamma) = \sum_{Cond(\alpha,\beta,\kappa)} P_\gamma(\alpha,\beta,\kappa) Q_{j-1}^0(\alpha) V_j(\beta), \tag{23}$$

$$Q_j^1(\gamma) = \sum_{Cond(\alpha,\beta,\kappa)} P_\gamma(\alpha,\beta,\kappa) Q_{j-1}^1(\alpha) V_j(\beta), \tag{24}$$

and if $Maj(\alpha) \oplus Maj(\beta) \neq Maj(\gamma)$,

$$Q_j^0(\gamma) = \sum_{Cond(\alpha,\beta,\kappa)} P_\gamma(\alpha,\beta,\kappa) Q_{j-1}^1(\alpha) V_j(\beta) \tag{25}$$

$$Q_j^1(\gamma) = \sum_{Cond(\alpha,\beta,\kappa)} P_\gamma(\alpha,\beta,\kappa) Q_{j-1}^0(\alpha) V_j(\beta), \tag{26}$$

where $Cond(\alpha,\beta,\kappa)$ determines the summation conditions according to

$$Cond(\alpha,\beta,\kappa) = \begin{cases} \forall \alpha,\beta,\kappa : \\ \alpha - \beta + 2\kappa = \gamma, \\ max(0, \beta - \alpha) \leq \kappa \leq min(L_a - \alpha, \beta). \end{cases} \tag{27}$$

In Q_{W-1}, we have the distribution of all but the last tap position vector. If we know that the sequence comes from a LFSR, we know that the last vector must force the total vector sum to the all-zero vector. Thus we get the probability that the xor of the majority bits (m_j) fulfills the recurrence equation as

$$P_M = P(\sum_{j=1}^{W} m_j = 0) = \sum_{\alpha=0}^{L_a} Q_{W-1}^0(\alpha). \tag{28}$$

These are the probabilities of the vectors that *will* have 0 as "majority parity bit", thus fulfilling the recurrence equation.