Reduced Complexity Correlation Attacks on Two Clock-Controlled Generators

Thomas Johansson*

Dept. of Information Technology
Lund University, P.O. Box 118, 221 00 Lund, Sweden

Abstract. The *Shrinking Generator* and the *Alternating Step Generator* are two of the most well known clock-controlled stream ciphers. We consider correlation attacks on these two generators, based on an identified relation to the decoding problem for the deletion channel and the insertion channel, respectively. Several ways of reducing the decoding complexity are proposed and investigated, resulting in "divide-and-conquer" attacks on the two generators having considerably lower complexity than previously known attacks.

1 Introduction

A binary additive stream cipher is a synchronous stream cipher in which the keystream, the plaintext and the ciphertext are sequences of binary digits. The output of the keystream generator, $z_1, z_2, \ldots$ is added bitwise to the plaintext sequence $m_1, m_2, \ldots$, producing the ciphertext $c_1, c_2, \ldots$. Each secret key k as input to the keystream generator corresponds to an output sequence. Since the secret key k is shared between the transmitter and the receiver, the receiver can decrypt by adding the output of the keystream generator to the ciphertext, obtaining the message sequence.

The goal in stream cipher design is to efficiently produce random-looking sequences that in some sense are "indistinguishable" from truly random sequences. From a cryptanalysis point of view, a good stream cipher should be resistant against a *known-plaintext attack*. In a known-plaintext attack the cryptanalyst is given a plaintext and the corresponding ciphertext, and the task is to determine a key k. For a synchronous stream cipher, this is equivalent to the problem of finding the key k that produced a given keystream $z_1, z_2, \ldots, z_N$.

In stream cipher design, one usually use linear feedback shift registers, LFSRs, as building blocks in different ways, and the secret key is often used as the initial state of the LFSRs. A general methodology for producing random-like sequences from LFSRs that recently has been popular is *using the output of one or more LFSRs to control the clock of other LFSRs*. The purpose is to destroy the linearity of the LFSR sequences and hence provide the resulting sequence with a large linear complexity.

* Supported by the Foundation for Strategic Research - PCC under Grant 97-130.

K. Ohta and D. Pei (Eds.): ASIACRYPT'98, LNCS 1514, pp. 342–356, 1998.

The most important general attacks on LFSR-based stream ciphers are *correlation attacks*. Basically, if one can in some way detect a correlation between the known output sequence and the output of one individual LFSR, this can be used in a "divide-and-conquer" attack on the individual LFSR [13,11,5,6].

Two of the most well known clock-controlled stream ciphers are the Shrinking generator and the Alternating step generator. In this paper we consider correlation attacks on these two generators. Some basic attacks have been considered when the generators were introduced [1] and [8], and further studies in [7] and [4]. For an overview, see [12].

Our considerations are based on an identified relation to the decoding problem on the deletion channel and the insertion channel, respectively. Several ways of reducing the decoding complexity are proposed and investigated, resulting in "divide-and-conquer" attacks on the two generators mentioned above having considerably lower complexity than previously known attacks. For example, for the Shrinking generator with shift register length 61 as suggested in [10], but with known feedback polynomial, the complexity of breaking this generator is reduced from around 2^{80} [1] to $2^{40} - 2^{50}$ depending on the length of the received sequence.

In Section 2 we describe the Shrinking generator and the Alternating step generator, respectively. We also show the relation to the decoding problem for the deletion/insertion channel. In Section 3 we consider a procedure for MAP decoding on the deletion channel. In Section 4 we propose a suboptimal MAP decoding procedure with reduced complexity and then demonstrate how certain "weak" subsequences that appear in the output sequence can be used to further reduce the complexity of a "divide-and-conquer" attack on the Shrinking generator. In Section 5 and 6 the same ideas are used on the Alternating step generator and the insertion channel, essentially showing the same type of complexity reduction.

2 Preliminaries

The Shrinking Generator, or SG for short, uses two sources of pseudorandom bits to create a third source of pseudorandom bits, having better cryptographic quality than the original sources. The output sequence is a subsequence of the first source, which is selected according to the values of the second source. The two original sources are in the proposal [1] chosen to be two maximal length linear feedback shift registers (LFSR).

The output sequence is more precisely defined as follows. Let $\mathbf{a} = a_1, a_2, \ldots$ denote the output of the first LFSR, denoted LFSR_A, and let $\mathbf{s} = s_1, s_2, \ldots$ denote the output of the second LFSR, denoted LFSR_S. The two LFSRs have length L_A and L_S respectively. The output sequence of the generator, denoted $\mathbf{z} = z_1, z_2, \ldots$, is the sequence obtained from $\mathbf{a} = a_1, a_2, \ldots$ by removing all a_i's for which $s_i = 0$. This is depicted in Figure 1.

The Alternating Step Generator, or ASG for short, is closely related to the stop-and-go generator and was proposed by Günther [8] in 1987. See [8] for a

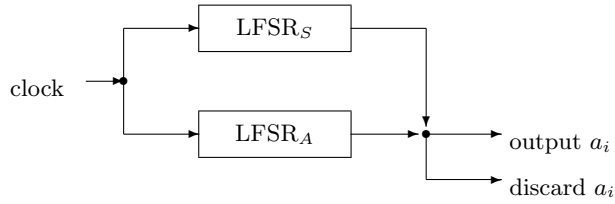

Fig. 1. The Shrinking Generator

further description of the ASG. Let us describe a modified version of the ASG, which we call ASG'.

Description of ASG': Again, we have three LFSRs, where LFSR$_S$ controls the clock of the two other LFSRs. Let LFSR$_S$ generate the sequence $\mathbf{s} = s_1, s_2, \ldots$. If $s_i = 1$ then the output symbol z_i is the output symbol from LFSR$_A$, and LFSR$_A$ is clocked. Otherwise, if $s_i = 0$ then the output symbol z_i is the output symbol from LFSR$_B$, and LFSR$_B$ is clocked. The ASG' is shown in Figure 2. It is not hard to show that ASG and ASG' are equivalent and hence we only

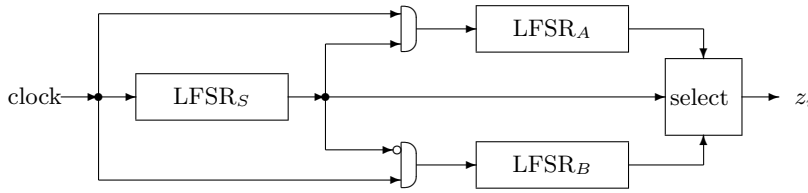

Fig. 2. The modified Alternating Step Generator, ASG'.

consider the ASG' in the sequel.

In the case of the SG, it was observed by Golic and O'Connor [4] that the sequence $\mathbf{a}$ can be recovered from the output sequence $\mathbf{z}$ if we can solve the corresponding *decoding problem on the deletion channel*. The deletion/insertion channel is a communication channel where the input symbols are deleted with a probability p and between any two undeleted input symbols i random symbols are inserted with distribution $P(i \text{ insertions}) = q^i(1-q)$, $i \geq 0$. If there are no insertions we call the channel the deletion channel, and if there are no deletions we call it the insertion channel.

For the SG we regard the sequence $\mathbf{s}$ from LFSR$_S$ as random and try to decode the output $\mathbf{z}$ to the correct sequence $\mathbf{a}$. It can be easily verified that if we assume that the sequence $\mathbf{a} = a_1, a_2, \ldots$ is the input to the deletion channel and the sequence $\mathbf{z} = z_1, z_2, \ldots$ is the output, the requirements for the deletion channel is fulfilled and the parameter p is $p = 1/2$. Since there are only 2^{L_A} possible input sequences an output sequence is uniquely decodable if it is long

enough and if the channel has a positive channel capacity [3]. With a fixed set of possible initial states we decode by simply checking each possible sequence with a MAP decoding algorithm, to be described in Section 5.

Having modified the ASG to ASG' we can then see that if we assume that the sequence $\mathbf{a} = a_1, a_2, \ldots$ is the input to the insertion channel and the sequence $\mathbf{z} = z_1, z_2, \ldots$ is the output, the requirements for the insertion channel is fulfilled and the parameter q is $q = 1/2$.

3 MAP Decoding on the Deletion Channel

By definition, a MAP decoding algorithm finds an input sequence $\mathbf{a}$ that for given $\mathbf{z}$ maximizes $P(\mathbf{a}$ transmitted$|\mathbf{z}$ received$)$, whereas a ML decoding algorithm finds a sequence $\mathbf{a}$ maximizing $P(\mathbf{z}$ received$|\mathbf{a}$ transmitted$)$. The derivation to be given is related to [9,4].

Assume that $a_1, \ldots, a_{L_A}$ is the given initial state of LFSR_A at time zero. Each initial state gives rise to a corresponding infinite sequence $\mathbf{a} = a_1, a_2, \ldots$. Denote by $\mathcal{A}$ the set of possible sequences. Assume also that the output sequence $\mathbf{z}$ is an infinite sequence $\mathbf{z} = z_1, z_2, \ldots$ obtained by transmitting some sequence $\mathbf{a}$ over the deletion channel, i.e. the sequence $\mathbf{a} = a_1, a_2 \ldots$ gives the output $\mathbf{z} = z_1, z_2 \ldots$. Let $\mathbf{A} = A_1, A_2, \ldots$ and $\mathbf{Z} = Z_1, Z_2, \ldots$ be the corresponding random variables. Continuing, we consider input sequences of fixed length t. Thus let $\mathbf{a}^t$ denote the sequence $\mathbf{a}^t = a_1, a_2, \ldots, a_t$, and let $\mathbf{A}^t = A_1, A_2, \ldots, A_t$ be the corresponding random variable. For a fixed length t the MAP decoding procedure calculates

$$P(\mathbf{A}^t = \mathbf{a}^t | \mathbf{Z} = \mathbf{z}), \tag{1}$$

for all sequences in $\mathcal{A}$ and selects a sequence $\mathbf{a} \in \mathcal{A}$ maximizing (1).

The length of the output sequence after t input symbols can be any value in $[0, t]$. Hence, introduce the random variables $\phi_t, t \geq 0$ as the number of output symbols after t input symbols. We can then write the above equation as

$$P(\mathbf{A}^t = \mathbf{a}^t | \mathbf{Z} = \mathbf{z}) = \sum_{i=0}^{t} P(\mathbf{A}^t = \mathbf{a}^t, \phi_t = i | \mathbf{Z} = \mathbf{z}). \tag{2}$$

The calculation of $P(\mathbf{A}^t = \mathbf{a}^t, \phi_t = i | \mathbf{Z} = \mathbf{z})$ can then be done iteratively by observing that

$$\begin{aligned} &P(\mathbf{A}^t = \mathbf{a}^t, \phi_t = i | \mathbf{Z} = \mathbf{z}) = \\ &\quad P(\mathbf{A}^{t-1} = \mathbf{a}^{t-1}, \phi_{t-1} = i | \mathbf{Z} = \mathbf{z}) P(A_t = a_t, \phi_t = i | \phi_{t-1} = i, \mathbf{Z} = \mathbf{z}) \\ &\quad + P(\mathbf{A}^{t-1} = \mathbf{a}^{t-1}, \phi_{t-1} = i - 1 | \mathbf{Z} = \mathbf{z}) P(A_t = a_t, \phi_t = i | \phi_{t-1} = i - 1, \mathbf{Z} = \mathbf{z}). \end{aligned} \tag{3}$$

We further observe that

$$P(A_t = a_t, \phi_t = i | \phi_{t-1} = i, \mathbf{Z} = \mathbf{z}) = \frac{1}{4}, \tag{4}$$

since a deletion occurs with probability 1/2 and then $A_t = a_t$ also with probability 1/2. Furthermore

$$P(A_t = a_t, \phi_t = i | \phi_{t-1} = i-1, \mathbf{Z} = \mathbf{z}) = \begin{cases} \frac{1}{2} \text{ if } a_t = z_i \\ 0 \text{ otherwise} \end{cases}, \tag{5}$$

because in this case there should be no deletion, which occur with probability 1/2. Then $A_t = z_i$ and thus $A_t = a_t$ has probability 1 if $a_t = z_i$ and 0 otherwise.

With given sequences $\mathbf{a}, \mathbf{z}$, each $\mathbf{A}^t = \mathbf{a}^t, \phi_t = i$, $0 \leq t\ T$, $0 \leq i \leq T$ can be considered as a node, denoted (t, i). Then the iterative calculation gives rise to a trellis, where $P(\mathbf{A}^t = \mathbf{a}^t, \phi_t = i | \mathbf{Z} = \mathbf{z})$ is the *metric* associated with each node. For simplicity, denote $P(\mathbf{A}^t = \mathbf{a}^t, \phi_t = i | \mathbf{Z} = \mathbf{z})$ simply by $\mu(t, i)$. The metric of new nodes will be updated according to whether $a_t = z_i$ or not. The metric update is obtained by combining (3), (4) and (5) as

$$\mu(t, i) = \mu(t-1, i)\frac{1}{4} + \mu(t-1, i-1)\frac{1}{2}\delta(a_t, z_i), \tag{6}$$

where

$$\delta(x, y) = \begin{cases} 1 \text{ if } x = y \\ 0 \text{ otherwise} \end{cases}.$$

As previously shown, we have

$$P(\mathbf{A}^t = \mathbf{a}^t | \mathbf{Z} = \mathbf{z}) = \sum_{i=0}^{t} \mu(t, i).$$

Let $N(t, i)$ be the number of different paths from node $(0, 0)$ to node (t, i). Then $\mu(t, i) = N(t, i)/2^{2t-i}$. This implies that we only have to consider the number of paths to each node (t, i) and that we can choose $N(t, i)$ as the metric to calculate. The advantage is that $N(t, i)$ is always an integer. The metric update using $N(t, i)$ is

$$N(t, i) = N(t-1, i) + N(t-1, i-1)\delta(a_t, z_i), \tag{7}$$

with initial value $N(0, 0) = 1$. We illustrate the procedure of creating the trellis and calculating the $N(t, i)$ metric by a small example given in Figure 3.

As stated for the probabilistic attack described in [4] the length of the sequence on which the decoding is performed need to be at least $3L_A$ for unique decoding, and we here choose to use the length $4L_A$. A straightforward implementation of the MAP decoding procedure is quadratic in the length. Hence the obtained complexity will be roughly $(4L_A)^2$ simple operations (about $(4L_A)^2/4$ nodes in the trellis each requiring one calculation of $\mu(t, i)$).

4 Reduced Complexity Decoding – Deletion Channel

We reduce the decoding complexity using two different approaches. Firstly, we propose and examine a suboptimal decoding algorithm, i.e., an algorithm with

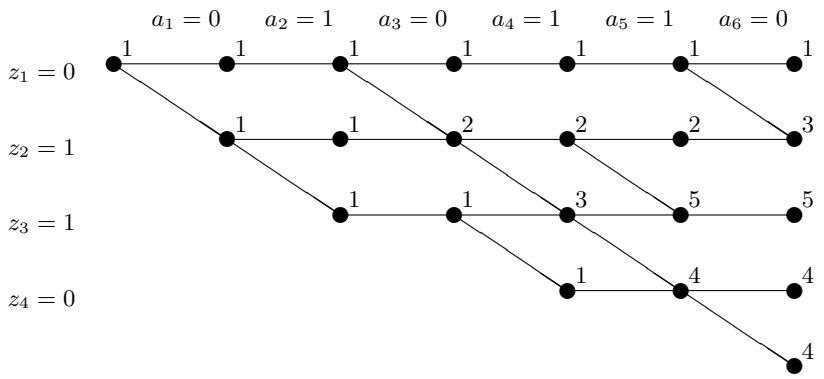

Fig. 3. A trellis with metric $N(t, i)$ for $\mathbf{a} = 0, 1, 0, 1, 1, 0, \ldots$ and $\mathbf{z} = 0, 1, 1, 0, \ldots$.

reduced complexity that has an almost optimal behavior. We use a stopping rule for the decision together with a list decoding approach (we keep only a fixed number of nodes at each time instant). Properties of this suboptimal decoding procedure is considered in the appendix. Let $C_{MAP'}$ be the expected complexity of testing one sequence with the above suboptimal algorithm. The algorithm implies a divide-and-conquer attack on LFSR_A by exhaustively testing all initial states. The complexity of such an attack is then approximately $2^{L_A} \cdot C_{MAP'}$.

Our second objective is to demonstrate that certain subsequences of the output sequence are weak in the sense that when they occur, they can be used to find the initial state of LFSR_A with lower complexity than exhaustively testing as mentioned above.

Assume that the output sequence $Z = z_1, z_2, \ldots$ contains a subsequence $z_T, z_{T+1}, \ldots, z_{T+M}$ such that either

$$(z_T, z_{T+1}, \ldots, z_{T+M}) = (0, 0, \ldots, 0, 1, 0, \ldots, 0) \tag{8}$$

or

$$(z_T, z_{T+1}, \ldots, z_{T+M}) = (1, 1, \ldots, 1, 0, 1, \ldots, 1). \tag{9}$$

The subsequence $z_T, z_{T+1}, \ldots, z_{T+M}$ is of length $M + 1$. W.l.o.g we can assume that (8) holds. Define the time t to be zero *exactly* where the occurrence of the single 1 is in the subsequence. In our notation, this means that $a_0 = 1$ and $s_0 = 1$. Let us now calculate $P(a_1 = 0)$ as follows,

$$P(a_1 = 0) = P(a_1 = 0|s_1 = 0)P(s_1 = 0) + P(a_1 = 0|s_1 = 1)P(s_1 = 1) \tag{10}$$

$$= \frac{1}{2} \cdot \frac{1}{2} + 1 \cdot \frac{1}{2} \tag{11}$$

$$= \frac{3}{4}, \tag{12}$$

since if $s_1 = 1$ then $a_1 = z_1 = 0$, and if $s_1 = 0$ then a_1 take any value in $\{0, 1\}$ with approximately equal probability.

The same arguments as above can be applied to $P(a_2 = 0)$ etc, as well as to $P(a_{-1} = 0)$ etc, and it is clear that

$$P(a_i = 0) = \frac{3}{4}, \quad -M_1 \leq i \leq M_2, \quad i \neq 0.$$

Furthermore, the deletion rate is 1/2. Hence, assuming M_1 deletions appearing in $a_{-2M_1}, \ldots, a_{-1}$, and M_2 deletions in $a_1, \ldots, a_{2M_2}$, we would end up with

$$P(a_i = 0) = \begin{cases} \frac{3}{4}, & 2M_1 \leq i \leq 2M_2, i \neq 0 \\ 0, & i = 0. \end{cases} \tag{13}$$

If the number of deletions is not exactly M_1 and M_2 respectively, they are at least close to these values and the distribution is close to the above.

With such a strong correlation identified, we can use it to reduce the complexity of the exhaustive search. We simply define the initial state to include the positions $a_{-2M_1}, \ldots, a_{2M_2}$ and search according to the above distribution. This idea can then be extended in different ways. We here examine three different approaches.

A. **Direct exhaustive search:** Using the proposed decoding procedure in the appendix we exhaustively search all 2^{L_A} initial states. The complexity of finding the correct initial state is on average $C_{MAP'} \cdot 2^{L_A - 1}$, and the length of $\mathbf{z}$ can be very small.

B. **Search using one weak subsequence:** We identify one weak subsequence of the form (8) or (9) of length $M+1$. Assume $2M \leq L_A - 1$. Define the initial state to include the $2M + 1$ index positions for which (13) hold, possibly together with some additional index positions. Search through all initial states having at most $M/2$ 1's in the corresponding $2M$ index positions and $a_0 = 1$, using the proposed decoding procedure. If the correct initial state is not found, an error is declared. The complexity of finding the correct initial state is on average approximately

$$C_{MAP'} \cdot 2^{L_A - 2M - 1} \binom{2M}{M/2},$$

and the required expected length of z for the weak subsequence to occur is approximately $2^M/M$.

C. **Search using several weak subsequences:** We identify W weak subsequences of the form (8) or (9) all of length $M + 1$, where $M \leq L_A - 1$. Then define the initial state to include the $2M + 1$ index positions for which (13) hold, possibly together with some additional index positions. Search through all initial states having at most w ones in the corresponding $2M$ index positions and $a_0 = 1$, using the proposed decoding procedure. If the correct initial state is not found, take a new weak subsequence and do the same again. If the correct initial state is not found after all weak subsequences have been used, an error is declared.

The complexity is approximately $C_{MAP'} \cdot W 2^{L_A-2M-1} \binom{2M}{w}$, and the expected observed length of $\mathbf{z}$ for W weak subsequences to occur is approximately $W2^M/M$. For the probability of finding the correct sequence to be large, W must be chosen such that

$$1/W \approx \sum_{i=0}^{w} (3/4)^{2M-w}(1/4)^w(1/2)^{L_A-2M-1}.$$

In order to show different possibilities and choices of parameters we consider an example where LFSR_A of the SG has $L_A = 61$ with known feedback polynomial. Our attack then applies to any LFSR_S having arbitrary degree L_S and possibly unknown feedback polynomial. In a comparison we choose $L_S \approx L_A = 61$. (In [10] it was suggested to choose length $61 - 64$ for both LFSRs and using secret feedback polynomials). A very rough estimate of the complexity in simple instructions for recovering the initial state of LFSR_A for different lengths of $\mathbf{z}$ and different methods are given in Table 1.

	Length of $\mathbf{z}$		
	2^{20}	2^{30}	2^{40}
Exhaustive search on LFSR_S [12]	2^{80}	2^{80}	2^{80}
Exhaustive search on LFSR_A [4]	2^{77}	2^{77}	2^{77}
A.	2^{71}	2^{71}	2^{71}
B.	2^{58}	2^{56}	2^{56}
C. with $2M = L_A - 1$		2^{50}	2^{40}

Table 1. Rough estimate of complexity for different attacks on the SG with $L_A = 61$.

4.1 Comments on the Values of Table 1:

As described in [12], the divide-and-conquer attack on LFSR_S requires approximately $2^{L_S} L_A^3$ operations which for $L_S \approx L_A = 61$ is around 2^{80} independent of output length. Furthermore, using the probabilistic attack described in [4] in an exhaustive search requires approximately $2^{L_A} \cdot (4L_A)^2$ operations, since the length of the sequence on which the decoding is performed need to be at least $3L_A$ for unique decoding (here chosen to be $4L_A$) and the decoding complexity is quadratic in the length.

For method A., the complexity is $2^{L_A} \cdot C_{MAP'}$, where the parameters of the suboptimal decoding algorithm is chosen such that $C_{MAP'} = 2^{10}$, giving an error probability of 0.34 as shown in the appendix. For method B., and output length 2^{20}, M is chosen to be $M = 25$. We then search through all sequences with at most 13 ones in $2M = 50$ index positions, one position fixed to 1, and the

remaining 10 positions arbitrarily. This gives complexity

$$\sum_{i=0}^{13} \binom{50}{i} \cdot 2^{10} \cdot C_{MAP'} \approx 2^{58}.$$

For output length 2^{30} or 2^{40}, we have $M = 30$ and searching through all sequences with at most 15 ones in $2M = 60$ index positions, one position fixed to 1, gives complexity

$$\sum_{i=0}^{15} \binom{60}{i} \cdot C_{MAP'} \approx 2^{56}.$$

Finally, for method C., we only consider the case when $M = 30$, which rules out the length 2^{20}. For length 2^{30} we expect to find about $2 \cdot 2^{30}/(2^{31}/31) = 31$ weak subsequences of the form (8) or (9). For each of these subsequences, we search through all sequences with at most 10 ones in $2M = 60$ index positions (one position fixed to 1). The probability of the event that the correct sequence has 10 or less ones in the $2M = 60$ index positions is approximately

$$\sum_{i=0}^{10} (3/4)^{60-i} (1/4)^i \binom{60}{i} \approx 0.08.$$

Hence the probability that at least one such event occurring among the 31 trials is large. The complexity of this procedure is roughly

$$\frac{1}{0.08} \cdot \sum_{i=0}^{10} \binom{60}{i} \cdot C_{MAP'} \approx 2^{50}.$$

Finally, for output length 2^{40} we expect about 2^{15} weak subsequences. The probability that the correct sequence has 3 or less ones in the $2M = 60$ index positions is

$$\sum_{i=0}^{3} (3/4)^{60-i} (1/4)^i \binom{60}{i} \approx 0.000047.$$

Searching through all sequences with at most 3 ones in the $2M = 60$ index positions gives complexity

$$\frac{1}{0.000047} \cdot \sum_{i=0}^{3} \binom{60}{i} \cdot C_{MAP'} \approx 2^{40}.$$

Note that in method A.–C. we only succeed with a certain probability when the selected $\mathbf{a}$ is correct, since the suboptimal MAP decoding fails with probability 0.34 etc.. If we want a very high probability of success, we can perform the whole process several times, each time on a new part of the output sequence. This slightly increases the complexity and the required length of the output sequence, compared to the above given numerical values.

4.2 Recovering the Initial State of LFSR$_S$

After having recovered the initial state of LFSR$_A$, we need to recover initial state of LFSR$_S$. This problem has not been addressed before. Due to the MAP decoding algorithm of Section 3, our candidate for initial state of LFSR$_A$ is correct with arbitrarily large probability (just run the algorithm long enough). We now proceed as follows. Again, run the MAP decoding algorithm and create the trellis. We have

$$P(s_t = 1|\mathbf{A} = \mathbf{a}, \mathbf{Z} = \mathbf{z}) = \sum_i P(\phi_t = i|\mathbf{A} = \mathbf{a}, \mathbf{Z} = \mathbf{z}) P(s_t = 1|\phi_t = i, \mathbf{A} = \mathbf{a}, \mathbf{Z} = \mathbf{z})$$
$$= \sum_i P(\phi_t = i|\mathbf{A} = \mathbf{a}, \mathbf{Z} = \mathbf{z}) P(s_t = 1|\phi_t = i, A_t = a_t, Z_i = z_i)$$

and

$$P(s_t = 1|\phi_t = i, A_t = a_t, Z_i = z_i) = \begin{cases} 0, & \text{if } a_t \neq z_i, \\ 2/3, & \text{if } a_t = z_i. \end{cases}$$

Furthermore,

$$P(\phi_t = i|\mathbf{A} = \mathbf{a}, \mathbf{Z} = \mathbf{z}) = \frac{\mu(t,i)}{\sum_j \mu(t,j)},$$

and hence we get

$$P(s_t = 1|\mathbf{A} = \mathbf{a}, \mathbf{Z} = \mathbf{z}) = \sum_i \frac{\mu(t,i)}{\sum_j \mu(t,j)} \cdot \frac{2}{3} \delta(a_t, z_i).$$

This procedure creates an a posteriori probability for each symbol s_t. Restoring the $\mathbf{s}$ sequence is now exactly the problem of decoding a received word to its nearest codeword on a noisy channel. One advantage is that the received word is very long and hence different ways of doing a fast decoding can be applied. One possibility is to use an iterative decoding process as suggested in [11] for fast correlation attacks. Another simpler method is to search for positions where the a posteriori probability $P(s_t = 1|\mathbf{A} = \mathbf{a}, \mathbf{Z} = \mathbf{z})$ is very small. This means that these positions are very likely to have $s_t = 0$. After finding L_S such positions one can perform a search over sequences having a low weight on these positions. The complexity of recovering the initial state of LFSR$_S$ using this approach is very low, and more details will be given in the full paper.

5 MAP Decoding on the Insertion Channel

In order to consider the ASG, we now consider MAP decoding on the insertion channel. The procedure is almost identical to the decoding procedure for the deletion channel. From Bayes rule,

$$P(\mathbf{A} = \mathbf{a}|\mathbf{Z}^t = \mathbf{z}^t) = P(\mathbf{Z}^t = \mathbf{z}^t|\mathbf{A} = \mathbf{a}) \cdot \frac{P(\mathbf{A} = \mathbf{a})}{P(\mathbf{Z} = \mathbf{z})}.$$

Since $P(\mathbf{A} = \mathbf{a})$ and $P(\mathbf{Z} = \mathbf{z})$ are both uniformly distributed it is clear that MAP and ML decoding is equivalent, i.e.,

$$\max_{\mathbf{a}} P(\mathbf{A} = \mathbf{a}|\mathbf{Z}^t = \mathbf{z}^t) = \max_{\mathbf{a}} P(\mathbf{Z}^t = \mathbf{z}^t|\mathbf{A} = \mathbf{a}).$$

It will now be apparent that the easiest way to describe a decoding process is in the form of ML decoding. Define the random variables $\phi_t, \quad t \geq 0$ to be the number of symbols from $\mathbf{a}$ that has appeared in $\mathbf{z}$ after observing t symbols from $\mathbf{z}$ (i.e. after observing $\mathbf{z}^t$). Clearly,

$$P(\mathbf{Z}^t = \mathbf{z}^t|\mathbf{A} = \mathbf{a}) = \sum_{i=0}^{t} P(\mathbf{Z}^t = \mathbf{z}^t, \phi_t = i|\mathbf{A} = \mathbf{a}),$$

and

$$\begin{aligned} &P(\mathbf{Z}^t = \mathbf{z}^t, \phi_t = i|\mathbf{A} = \mathbf{a}) = \\ &\quad P(\mathbf{Z}^{t-1} = \mathbf{z}^{t-1}, \phi_{t-1} = i|\mathbf{A} = \mathbf{a})P(Z_t = z_t, \phi_t = i|\mathbf{z}^{t-1}, \phi_{t-1} = i, \mathbf{A} = \mathbf{a}) \\ &\quad +P(\mathbf{Z}^{t-1} = \mathbf{z}^{t-1}, \phi_{t-1} = i-1|\mathbf{A} = \mathbf{a})P(Z_t = z_t, \phi_t = i|\mathbf{z}^{t-1}, \phi_{t-1} = i-1, \mathbf{A} = \mathbf{a}). \end{aligned}$$

Furthermore,

$$P(Z_t = z_t, \phi_t = i|\phi_{t-1} = i, \mathbf{A} = \mathbf{a}) = \frac{1}{4} \tag{14}$$

since an insertion occurs with probability 1/2 and then $Z_t = z_t$ also with probability 1/2. Finally

$$P(Z_t = z_t, \phi_t = i|\phi_{t-1} = i-1, \mathbf{A} = \mathbf{a}) = \begin{cases} \frac{1}{2} \text{ if } z_t = a_{\phi_{t-1}} \\ 0 \text{ otherwise} \end{cases}. \tag{15}$$

We can now see that the ML decoding procedure in this case is *exactly* the MAP decoding procedure for the deletion channel if the sequences $\mathbf{a}$ and $\mathbf{z}$ are switched. Hence all the results from Section 3 are valid also for the insertion channel if $\mathbf{a}$ and $\mathbf{z}$ are switched. With this conclusion we leave the details out.

6 Reduced Complexity Decoding – Insertion Channel

The ML decoding algorithm demonstrated in the previous section implies a divide-and-conquer attack on the ASG' by exhaustively testing all initial states of LFSR_A. The complexity of such an attack is then approximately $2^{L_A} \cdot C_{MAP'}$. We now demonstrate that again certain subsequences of the output sequence of the ASG' are weak in the sense that when they occur, they can be used to find the initial state of LFSR_A with lower complexity than exhaustively testing as mentioned above.

The basic observation is the following. Assume that the output sequence $\mathbf{z}$ contains a subsequence $z_T, z_{T+1}, \ldots, z_{T+M}$ such that either

$$(z_T, z_{T+1}, \ldots, z_{T+M-1}) = (0, 0, \ldots, 0) \tag{16}$$

or

$$(z_T, z_{T+1}, \ldots, z_{T+M-1}) = (1, 1, \ldots, 1). \tag{17}$$

Redefine the time t to be zero at time T and w.l.o.g assume (16). This means that now $(z_0, z_1, \ldots, z_{M-1}) = (0, 0, \ldots, 0)$. Then assuming that at least $M/2$ of the symbols in $\mathbf{z}^M$ (which has probability $> 1/2$) came from LFSR_A, we have $a_0 = 0, a_1 = 0, \ldots, a_{M/2} = 0$. Hence one can perform an exhaustive search over all possible initial states of LFSR_A with the first $M/2$ index positions set to zero. This will reduce the complexity with a factor $2^{M/2}$ compared with straightforward exhaustive search.

Having described the basic idea, we can now improve the performance in several ways. Instead of subsequences of the form (16) or (17), we consider any sequences of length $\geq M$ containing at most w ones (or at most w zeros). Each of the w ones comes from LFSR_A with probability $1/2$ and hence with probability 2^{-w} none of the ones comes from LFSR_A. If this occur, the first $(M - w)/2$ symbols of $\mathbf{a}$ are all zero with probability $> 1/2$. Hence a possible procedure is as follows.

1. Search for a length M subsequence of $\mathbf{z}$ containing at most w ones (or zeros).
2. Let t be zero at the beginning of the subsequence and assume that $a_0 = 0, \ldots, a_{(M-p)/2} = 0$. Then perform an exhaustive search over the remaining index positions $a_{(M-w)/2}, \ldots, a_{L_A-1}$.
3. Go to 1.

The conclusion is that by the basic observation the search is reduced by roughly a factor $\sqrt{L}$ where L is the length of the observed sequence, and by the above improvement the reduction factor can be made a bit larger.

Finally, the concept of weak subsequences are always present for the two considered generators. It is always possible that a "very weak" subsequence appear, implying a successful attack with very low complexity, even though the probability of such a sequence is very low. For example, an subsequence of $2L_A$ consecutive zeros or ones from the ASG' implies a successful attack with complexity almost zero.

References

1. D. Coppersmith, H. Krawczyk, and Y. Mansour, "The shrinking generator, *Lecture Notes in Computer Science* 773 (CRYPTO'93), pp. 22–39, 1994.
2. T. Cover, A. Thomas, *Elements of Information Theory*, Wiley and Sons, 1991.
3. A.S. Dolgapolov, "Capacity bounds for a channel with synchronization errors", *Problemy Peredachi Informatsii*, 26, pp. 27–37, 1990.
4. J. Golic', and L. O'Connor, "Embedding and probabilistic correlation attacks on clock-controlled shift registers", *Lecture Notes in Computer Science* 950 (EUROCRYPT'94), pp. 230–243, 1995.
5. J. Golic', M. Mihaljevic', "A generalized correlation attack on a class of stream ciphers based on the Levenstein distance", *Journal of Cryptology*, 3 (1991), pp. 201–212.
6. J. Golic', "Towards fast correlation attacks on irregularly clocked shift registers", *Lecture Notes in Computer Science* 921 (EUROCRYPT'95), pp. 248–262, 1995.

7. J. Golic', "Intrinsic statistical weakness of keystream generators", *Lecture Notes in Computer Science* 917 (ASIACRYPT'94), pp. 91–103, 1995.
8. C.G. Günther, "Alternating step generators controlled by de Bruijn sequences", *Lecture Notes in Computer Science* 304 (EUROCRYPT'87), pp. 5–14, 1988.
9. P. Hall, G. Dowling, "Approximate string matching", *Computing Surveys*, 12, pp. 381–402, 1980.
10. H. Krawczyk, "The shrinking generator: Some practical considerations, *Lecture Notes in Computer Science* 809 (FSE), pp. 45–46, 1994.
11. W. Meier, and O. Staffelbach, "Fast correlation attacks on certain stream ciphers", *Journal of Cryptology*, 1, pp. 159–176, 1989.
12. A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
13. T. Siegenthaler, "Correlation-immunity of nonlinear combining functions for cryptographic applications", *IEEE Trans. on Info. Theory*, 30 (1984), pp. 776–780.

Appendix: A Suboptimal MAP Decoding Algorithm

Introduce the random variables $X_t = P(\mathbf{A}^t = \mathbf{a}^t | \mathbf{Z} = \mathbf{z})$, i.e., $X_t = \sum_{i=0}^{t} \mu(t,i)$. Using the recursion (3) we have

$$X_t = \sum_{i=0}^{t} (\frac{1}{4}\mu(t-1,i) + \frac{1}{2}\mu(t-1,i-1)\delta(a_t,z_i)),$$

which simplifies to

$$X_t = \frac{1}{4}X_{t-1} + \frac{1}{2}X_{t-1}\frac{\sum_{i:a_t=z_i}\mu(t-1,i-1)}{X_{t-1}}.$$

Introduce the random variables $S_t = 2\frac{\sum_{i:a_t=z_i}\mu(t-1,i-1)}{X_{t-1}}$. Then

$$X_t = \frac{1}{4}X_{t-1} + \frac{1}{4}X_{t-1}S_t = \frac{1}{2}X_{t-1}\frac{1+S_t}{2}$$

and after taking logarithms one gets

$$\log X_t = -1 + \log X_{t-1} + \log\frac{1+S_t}{2} = -t + \sum_{i=1}^{t}\log\frac{1+S_t}{2}$$

Clearly, for $\mathbf{a}$ being a random sequence $P(a_t = z_i) = 1/2$ for all t, i and hence $E(S_t) = 1$ implying $E(\frac{1+S_t}{2}) = 1$. Now, the Jensen's inequality [2] states that $E(\log\frac{1+S_t}{2}) \leq \log E(\frac{1+S_t}{2})$ with equality only if S_t has a deterministic distribution. But S_t has *not* a deterministic distribution and hence $E(\log\frac{1+S_t}{2}) < \log 1 = 0$. Hence

$$E(\log X_t) = -t + \sum_{i=1}^{t} E(Y_i), \tag{18}$$

where each $E(Y_i) < 0$.

Next, consider **a** being the correct sequence. Then if P_t is the number of output symbols after t input symbols we have

$$P(a_t = z_i) = \begin{cases} 1/2, \text{ if } i \neq P_t \\ 3/4, \text{ if } i = P_t \end{cases}$$

and hence $E(S_t) = 1 + \frac{1}{2}E(\mu(t-1, P_t))$. Without being able to formally prove the fact, simulations show that $E(\log \frac{1+S_t}{2}) > 0$ for this case. Thus we have the same expression as in (18), but now with $E(Y_i) > 0$.

These facts have been investigated by simulations. The value of $\log X_t - t$ expressed in the form $\log X_t - t = C \cdot t$ have been investigated for different t, covering both cases (random sequence, correct sequence). The result is tabulated in Table 2.

$\log X_t - t$		
t = Length of $\mathbf{a}^t$	CORRECT **a**	RANDOM **a**
10	$0.090775 \cdot t$	$-0.096476 \cdot t$
20	$0.072374 \cdot t$	$-0.097060 \cdot t$
30	$0.064135 \cdot t$	$-0.083986 \cdot t$
40	$0.059176 \cdot t$	$-0.069533 \cdot t$
60	$0.053739 \cdot t$	$-0.057280 \cdot t$
80	$0.050644 \cdot t$	$-0.051867 \cdot t$
100	$0.048625 \cdot t$	$-0.049880 \cdot t$
200	$0.044149 \cdot t$	$-0.051356 \cdot t$
500	$0.041362 \cdot t$	$-0.042727 \cdot t$
1000	$0.040404 \cdot t$	$-0.036148 \cdot t$
5000		$-0.033180 \cdot t$

Table 2. Tabulation of $\log X_t - t$ for **a** being a correct/random sequence.

Clearly, $\{X_t, t \geq 1\}$ is a stochastic process, for which a stopping rule can be introduced. For implementation purposes, we simplify this to apply only on certain index positions, e.g., $D \cdot n$ for $n = 1, 2, \ldots$ and some integer D. Such a stopping rule will introduce a small probability of error in our hypothesis testing problem, i.e., $P(\text{“not correct”}|\mathbf{a} \text{ “correct”}) = \epsilon > 0$. On the other hand, this will significantly reduce the computational complexity since in an exhaustive search the algorithm will terminate very quickly for most random sequences **a**. One also has to select a decision region for stopping/not stopping. Looking at Table 2 a suitable choice might be to stop if $\log X_t - t < 0$.

Secondly, a closer look at $\mu(t, i)$ for given t and a correct sequence **a** shows that the probability mass of $\mu(t, i)$ is concentrated to a few nodes (t, i) on each level. Therefore, one can consider a suboptimal decoding algorithm that only stores the L most probable nodes on each level. Furthermore, if I_{max} is the largest i such that $N(t, i) \neq 0$, then the nodes $(t, i), i = I_{max-L+1}, \ldots, I_{max}$ is a good approximation of the L most probable nodes, which we use.

These considerations give the following proposed algorithm.

1. Initialize $N(0,0) = 1$, $I_{max} = 1$.
2. If $a_t = z_{I_{max}}$ then $I_{max} = I_{max} + 1$. Update
$$N(t,i) = N(t-1,i) + N(t-1,i-1)\delta(a_t, z_i),$$
for $i = I_{max} - L + 1, \ldots, I_{max}$.
3. If $t = 0 \pmod D$ do the following. Calculate $X_t = \sum_{i=I_{max}-L+1}^{I_{max}} N(t,i)/2^{2t-i}$. If $\log X_t - t < 0$ output "wrong sequence" and stop. If $t \geq T_{max}$ output "correct sequence" and stop.
4. Increase t by 1 and go to 2.

The above algorithm is given to be easily understood. When implementing it, several steps above should be done differently.

The performance of the algorithm relies on the relation between two important parameters, the probability of declaring a wrong sequence when having the correct one and the average complexity of the decoding algorithm for a random sequence (until it stops and outputs "wrong"). To measure the average complexity we consider the average depth of the trellis before stopping, i.e., if T_{stop} is the value of $t = D \cdot n$ when the algorithm stops, i.e., when $\log X_{D \cdot n} - D \cdot n < 0$ for the first $n = 1, 2, \ldots$. This is suitable since for a fixed number L of remaining states in each level of the trellis, the decoding complexity is (essentially) a linear function of T_{stop} and hence the expected decoding complexity a linear function of $E(T_{stop})$. Some simulated values of the above parameters are given in Table 3. The final conclusion of this section is a choice of parameters for complexity cal-

D	P("wrong" output|**a** correct)	$E(T_{stop})$
10	0.34	25.4
20	0.24	34.8
30	0.21	50
50	0.16	70
100	0.10	117
200	0.01	209

Table 3. Performance of the proposed algorithm in terms of error probability and decoding complexity for different D when $L = 10$.

culations. Selecting $D = 10$ and $L = 10$ will give an error probability of 0.34 and expected trellis depth 25.4. In this case the expected number of nodes in the trellis is less than 200 (there are fewer than L nodes on each level in the beginning of the trellis). Each node requires a few instructions to be updated, resulting in a very rough estimate of $C_{MAP'} = 2^{10}$ simple operations as an average of the complexity of testing one sequence **a**.