A New Distinguisher for Clock Controlled Stream Ciphers

Håkan Englund and Thomas Johansson

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

Abstract. In this paper we present a distinguisher targeting towards irregularly clocked filter generators. The attack is applied on the irregularly clocked stream cipher called LILI-II. LILI-II is the successor of the cipher LILI-128 and its design was published in [1]. There have been no known attacks better than exhaustive key search on LILI-II. Our attack is the first of this kind that distinguishes the cipher output from a random source using 2^{103} bits of keystream using computational complexity of approximately 2^{103} operations.

1 Introduction

Stream ciphers are a part of the symmetric family of encryption schemes. Stream ciphers are divided into two classes, synchronous and self-synchronous. In this paper we will consider a special class of the synchronous stream ciphers, namely irregularly clocked binary stream ciphers. The considered class of irregularly clocked stream ciphers include a filter generator from which the output is decimated in some way. A filter generator consists of a linear part and a boolean function (typically a nonlinear boolean function). To create the keystream some positions are taken from the internal state of the linear part and fed into the boolean function. The output of the boolean function is then combined with the message by an output function, typically the XOR operation.

Although there exist standardized block ciphers like AES [2], many people believe that the use of stream ciphers can offer advantages in some cases, e.g., in situations when low power consumption is required, low hardware complexity or when we need extreme software efficiency. To reinforce the trust in stream ciphers it is imperative that the security of stream ciphers are carefully studied.

Several different kinds of attacks can be considered on stream ciphers. We usually consider the plaintext to be known, i.e. the keystream is known and we try to recover the key. In 1984 Siegenthaler [3] introduced the idea of exploiting the correlations in the keystream. As a consequence of this attack, nonlinear functions must have high nonlinearity.

This attack was later followed by the fast correlation attack by Meier and Staffelbach [4]. In a fast correlation attack one first tries to find a low weight parity check polynomial of the LFSR and then applies some iterative decoding procedure. Many improvements have been introduced on this topic, see [5–10].

Algebraic attacks have received much interest lately. These attacks try to reduce the key recovery problem to the problem of solving a large system of algebraic equations [11, 12].

In this paper we will consider a distinguishing attack. A distinguishing attack is a known keystream attack, i.e., we have access to some amount of the keystream and from this data we try to decide whether this data origins from the cipher we consider, or if the data appears to be random data, see e.g., [13–17].

One of the submissions to the NESSIE project [18] was the irregularly clocked stream cipher LILI-128 [19]. Several attacks such as [20, 7, 11, 12, 17] on LILI-128 motivated a larger internal state. The improved design that became the successor of LILI-128 is called LILI-II [1], and it was first published in ACISP 2002. LILI-II was designed by Clark, Dawson, Fuller, Golić, Lee, Millan, Moon and Simpson, and uses a 128 bit key which is expanded and used with a much larger internal state, namely 255 bits instead of 128 in LILI-128.

So far no attacks on LILI-II have been published. In this paper we present a distinguishing attack on LILI-II. The attack uses a low weight multiple of one of the linear feedback shift registers (LFSR), i.e., it belongs to the class of linear distinguishers, see [21, 22]. It collects statistics from sliding windows around the positions of the keystream, where the members of this recursion are likely to appear. The strength of the attack is the updating procedure used when moving the windows, this procedure allows us to receive many new samples with very few operations. To distinguish the cipher from a random source we need 2^{103} bits of keystream and the complexity is around 2^{103} operations. This is the first attack on LILI-II faster than exhaustive key search.

The paper is organized as follows, in Section 2 we explain some theory needed for the attack. Section 3 describes the idea and the different steps in the attack. In Section 4 we describe the stream cipher LILI-II, and how the attack can be applied to this cipher step by step. To verify the correctness of the attack some simulations are presented in Section 5. The results of the attack is assembled in Section 6, and finally we conclude the paper in Section 7.

2 Preliminaries

2.1 Irregularly Clocked Filter Generators

Our attack considers irregularly clocked stream ciphers where the output is taken from some LFSR sequence in some arbitrary way. Note that many well known designs are of this form, e.g., the shrinking generator, self-shrinking generator, alternating step, LILI-128, LILI-II etc. The case we consider in this paper is illustrated in Figure 1.

The keystream generator is divided into two parts, a clock control part and a data generation part. The clock control part produces symbols, denoted by c_t at time instant t in Figure 1, in some arbitrary way. This sequence determines how many times we should clock the LFSR in the data generation part, before we produce a new output symbol.

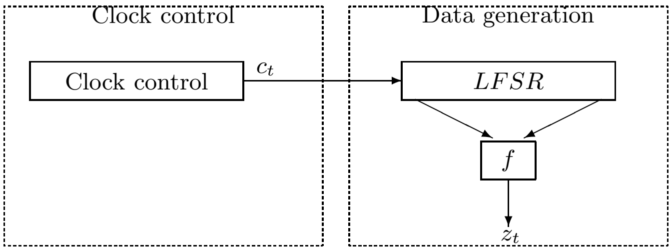

Fig. 1. A general model of an irregularly clocked filter generator.

The data generation part is a filter generator, i.e., an LFSR producing a linear sequence, denoted by $s_t, s_{t+1}, \ldots$, from this LFSR some symbols are taken from the internal state and are used as input into a boolean function, denoted f in Figure 1.

2.2 Finding a Low Weight Multiple

In our attack we need a low weight recursion of weight w for the LFSR sequence s, i.e, a relation that sums to zero for all time instances t.

$$s_t + s_{t+\tau_1} + \ldots + s_{t+\tau_{w-1}} = 0 \mod 2. \tag{1}$$

One technique to find such relations is to find multiples of the original feedback polynomial. Several methods to find such multiples of low weight has been proposed and they focus on optimizing different aspects, e.g., finding multiple with as low degree as possible, or accepting a higher degree but reducing the complexity to find the multiple. In our attack the degree of the multiple is of high concern.

Assume that we have a feedback polynomial $g(x)$ of degree r and search for a multiple of weight w, according to [23] the critical degree when these multiples start to appear is $(w-1)!^{1/(w-1)}2^{r/(w-1)}$. Golić [23] also describes an algorithm that focuses on finding multiples of the critical degree. The first step is to calculate the residues $x^i \mod g(x)$, then one computes the residues $x^{i_1} + \ldots x^{i_k} \mod g(x)$ for all $\binom{n}{k}$ combinations $1 \leq i_1 \leq \ldots \leq i_k \leq n$, with n being the maximum degree of the multiples. The last step is to use fast sorting to find all of the zero and one matches of the residues from the second step. The complexity of this algorithm is approximately $O(S \log S)$ with $S = \frac{(2k)!^{1/2}}{k!}2^{r/2}$ for odd multiples of weight $w = 2k+1$, and $S = \frac{(2k-1)!^{k/(2k-1)}}{k!}2^{rk/(2k-1)}$ for even multiples of weight $w = 2k$.

Wagner [24] presented a generalization of the birthday problem, i.e., given k lists of r-bit values, find a way to choose one element from each list, so that these k values XOR to zero. This algorithm finds a multiple of weight $w = k+1$ using lower computational complexity, $k \cdot 2^{r/(1+\lfloor \log k \rfloor)}$, than the method described above, on the expense of the multiples degree, which is $2^{r/(1+\lfloor \log k \rfloor)}$. Since the number of samples is of high concern to us we have chosen to work with the method described in [23]. From now on we assume that the LFSR sequence is described by a low weight recursion.

3 Description of the Attack

We consider a stream cipher as given in Figure 1, where $s_0, s_1, s_2, \ldots$ denotes the sequence from the LFSR in the data generation part, and $z_0, z_1, z_2, \ldots$ denotes the keystream sequence. The clock control mechanism c_t determines for each t how many times the LFSR is clocked before z_t is produced. After observing $z_0, z_1, \ldots, z_T$ the LFSR has been clocked $\sum_{t=0}^{T} c_t$ times. Since we are attacking irregularly clocked ciphers we will not know exactly where symbols from the LFSR sequence will be located in the output keystream, not even if they appear at all.

We will fix one position in the recurrence relation, and around the estimated location of the other symbols we will use sliding windows. When using windows of adequate size we have a high probability that all the symbols in the relation (if not removed by the irregular decimation) are included. We will then calculate how many symbols from the different windows sum to zero. Only one of these combinations contribute with a bias, the others will appear as random samples. In the following subsections we will describe the different steps we use in our attack.

The way we build the distinguisher is influenced by previous work on distinguishers, see for example [14, 21, 22, 25, 13].

3.1 Finding a Low Weight Multiple

The success of our attack depends on the use of low weight recurrence relations, hence the first step is to find a low weight multiple of the LFSR in the data generation part. In the attack we use a multiple of weight three, it is also possible to mount the attack with multiples of higher weight. Using a multiple of higher weight lowers the degree of the multiple, but it also lowers the probability that all symbols in a recurrence are included after the decimation, and in general also lowers the correlation property of the boolean function. So from now on we assume that we use a weight three recurrence relation. We will use the methods described in Section 2.2 to find the multiple.

3.2 Calculating the Correlation Property of f for a Weight Three Recursion

Assume that we have a weight three relation for the LFSR sequence according to

$$s_t + s_{t+\tau_1} + s_{t+\tau_2} = 0 \mod 2.$$

Let $S_t = (s_{t+i_1}, s_{t+i_2}, \ldots, s_{t+i_d})$ denote the input bits to f at time t taken from positions $i_1, i_2, \ldots, i_d$. The correlation property for weight three recurrence relation of the nonlinear boolean function f, denoted ε_f, is defined to be

$$\varepsilon_f = \left|\frac{1}{2} - \Pr\Big(f(S_t) \oplus f(S_{t+\tau_1}) \oplus f(S_{t+\tau_2}) \; = 0 \mid s_t \oplus s_{t+\tau_1} \oplus s_{t+\tau_2} = 0, \ \forall t \Big)\right|.$$

If the LFSR would be regularly clocked ($c_t = 1, \ \forall t$), the probability above is equivalent to

$$|\frac{1}{2} - \Pr(z_t + z_{t+\tau_1} + z_{t+\tau_2} = 0)|.$$

The correlation property can be calculated by simply trying all possible input combinations into the function. Since we use a weight three recursion some combinations will not be possible and the distribution will be biased (bias>0), see [17]. The correlation property of boolean functions has also been discussed in [20].

3.3 The Positions of the Windows

Consider again the weight three relation, but now with irregular clocking. We denote the expected value of the clocking sequence c_t by $E(C)$. The size of the windows depends on the distance from the fixed position, hence we will fix the center position in the recurrence and use windows around the two other positions. We rewrite the recurrence as

$$s_{t-\tau_1} + s_t + s_{t+\tau_2-\tau_1} = 0 \mod 2.$$

The expected distance between the output from f, corresponding to input $S_{t-\tau_1}$ and S_t, is $\tau_1/E(C)$ since the sequence is decimated, similarly the distance between S_t and $S_{t+\tau_2-\tau_1}$ is $\frac{\tau_2-\tau_1}{E(C)}$. Figure 2 illustrates how we position the windows of size r in the case of a weight three recursion.

$$\boxed{z_{t-\frac{\tau_1}{E(C)}-\frac{r_1}{2}} \cdots z_{t-\frac{\tau_1}{E(C)}+\frac{r_1}{2}}} \quad \cdots \quad \boxed{z_t} \quad \cdots \quad \boxed{z_{t+\frac{\tau_2-\tau_1}{E(C)}-\frac{r_2}{2}} \cdots z_{t+\frac{\tau_2-\tau_1}{E(C)}+\frac{r_2}{2}}}$$

Fig. 2. Illustration of the window positions in the case with a weight three recurrence relation.

3.4 Determining the Size of the Windows

The output sequence from the clock-control part, denoted by c_t in Figure 1, is assumed to have a fixed distribution independent of t. By using the central limit theorem we know that the sum of a large number of random variables approaches the normal distribution. So $Y_n = C_1 + C_2 + \ldots + C_n \in \mathrm{N}(n \cdot E(C), \sigma_c\sqrt{n})$, where n denotes the number of observed symbols, $E(C)$ the expected value of the clocking sequence and σ_c the standard deviation of the clocking sequence.

If the windows are sufficiently large, the correct position of the symbol will be located inside the window with a high probability,

$$\begin{aligned} P(nE(C) - \sigma_c\sqrt{n} < Y_n < nE(C) + \sigma_c\sqrt{n}) &= 0.682, \\ P(nE(C) - 2\sigma_c\sqrt{n} < Y_n < nE(C) + 2\sigma_c\sqrt{n}) &= 0.954. \end{aligned}$$

Thus we choose a window size of four standard deviations.

3.5 Estimate the Number of Bits We Need to Observe

The main idea of the distinguishing attack is to create samples of the form

$$z_{w_1} + z_t + z_{w_2},$$

where w_1 is any position in the first window and w_2 is any position in the second window. We will run through all such possible combinations. As will be demonstrated, each sample is drawn according to a biased distribution.

To determine how many bits we need to observe to reliably distinguish the cipher from a random source, we need to make an estimate of the bias.

First we consider the case of a regularly clocked cipher. We denote the window sizes by $r_1,\ r_2$. In the following estimations we have to remember that we are calculating samples, and that for every time instant we get $r_1 \cdot r_2$ new samples. For each time instant one relation contributes with the bias ε_f, the other $r_1 \cdot r_2 - 1$ relations are random. The bias can roughly be calculated as

$$\varepsilon_f \cdot \frac{1}{r_1} \cdot \frac{1}{r_2},$$

assuming that $s_{t-\tau_1}$ and $s_{t+\tau_2-\tau_1}$ always appear inside the windows. When we have irregular clocking the output from the LFSR is decimated, i.e., some terms will not contribute to the output sequence. The probability that the end two terms of a weight three recurrence relation is included in the keystream and in the windows is denoted by p_{dec}. In the approximation we neglect the probability that the result in some cases deviates more than two standard deviations from the expected position, i.e., the component lies outside the window. This gives an estimate of the full bias showed in (2), the approximation has been compared with simulation results and works well, see Section 5.

$$\varepsilon_{full} = \varepsilon_f \cdot \frac{1}{r_1} \cdot \frac{1}{r_2} \cdot p_{dec}. \tag{2}$$

In the approximation we have estimated that the probability for the position of the taps inside the windows is uniform, the purpose is to make the updating procedure when moving the windows as efficient as possible, this is in fact the strength of the attack. A better approximation would be to weight the positions inside the window according to the normal distribution. This might decrease the number of needed symbols but would make an efficient updating procedure much more difficult.

We can now estimate how many keystream bits we need to observe, in order to make a correct decision. In [25] the *statistical distance* is used.

Definition 1 *The statistical distance, denoted* ε*, between two distributions* P_0, P_1 *defined over a finite alphabet* $\mathcal{X}$*, is defined as*

$$\varepsilon = |P_0 - P_1| = \frac{1}{2} \sum_{x \in \mathcal{X}} |P_0(x) - P_1(x)|, \tag{3}$$

where x *is an element of* $\mathcal{X}$*.*

If the distributions are smooth, the number of variables N we need to observe is $N \approx 1/\varepsilon^2$, see [25]. Note that the error probabilities are decreasing exponentially with N. Thus the number of samples we need to observe can be estimated as

$$\frac{r_1^2 \cdot r_2^2}{\varepsilon_f^2 \cdot p_{dec}^2}. \tag{4}$$

At each time instant we receive $r_1 \cdot r_2$ new samples, and hence the total number of bits we need for the distinguisher can be estimated by (5).

$$N \approx \frac{r_1 \cdot r_2}{\varepsilon_f^2 \cdot p_{dec}^2}. \tag{5}$$

The above Equations (2-5) assume independent samples, this is not true in our case, but the equation is still good approximation on the number of samples needed in the attack. A similar expression can be derived in a another independent way, in Appendix A it is stated that the standard deviation for the total sum of samples is $\sqrt{N\frac{r_1 r_2}{4}}$. The bias of the samples is denoted by ε_{tot}, for a successful attack $N\varepsilon_{tot} > 2\sqrt{N\frac{r_1 r_2}{4}}$ should hold, solving this equation for N gives $N > \frac{r_1 r_2}{\varepsilon_{tot}^2}$.

3.6 Complexity of Calculating the Samples

The strength of the distinguisher is that the calculation of the number of ones and zeros in the windows can be performed very efficiently, when we move the first position from z_t to z_{t+1} we also move the windows one step to the right.

We denote the number of zeros in window one at time instant t by X_t, similarly we denote the number of zeros in windows two by Y_t, the number of samples that fulfill $z_{w_1} + z_t + z_{w_2} = 0$ is denoted W_t, where w_1, w_2 are some positions in window one respectively window two . Hence when moving the windows we get the new number of zeros X_{t+1} and Y_{t+1} by subtracting the first bit in the old window and adding the new bit included in the window, e.g., for window one,

$$X_{t+1} = X_t - z_{t-\frac{\tau_1}{E(C)}-\frac{r_1}{2}} + z_{t-\frac{\tau_1}{E(C)}+\frac{r_1}{2}+1},$$

and similarly for window two. From the X_{t+1} and Y_{t+1} we can, with few basic computations calculate W_{t+1}.

We define one operation as the computations required to calculate W_{t+1} from X_t and Y_t.

Theorem 1. *The proposed distinguisher requires* $N = \frac{r_1 \cdot r_2}{\varepsilon_f^2 \cdot p_{dec}^2}$ *bits of keystream and uses a computational complexity of approximately* N *operations.*

Although the number of zeros in the windows X_{t+1}, Y_{t+1} are dependent of the previous number of zeros in the window X_t, Y_t, the covariance between the number of samples received at time instant t and $t+1$ is zero, $\mathrm{Cov}(W_{t+1}, W_t) = 0$, see Appendix A.

3.7 Hypothesis Testing

The last step in the attack is to determine whether the collected data really is biased. A rough method for the hypothesis test is to check whether the result deviates more than two standard deviations from the expected result in the case when the bits are truly random. The standard deviation for a sum of these samples, can be estimated by $\sigma = \sqrt{N \frac{r_1 r_2}{4}}$, see Appendix A, where r_1 and r_2 are the sizes of the windows and N is the number of bits of keystream we observe.

3.8 Summary of the Attack

In Figure 3 we summarize the attack, where X_t denotes the number of zeros in window one, Y_t the number of zeros in window two, W denotes the total sum of the samples and $r_1,\ r_2$ the sizes of window one respectively window two.

1. Find a weight three multiple of the$LFSR$.
2. Calculate the bias ε_f.
3. Determine the positions of the windows.
4. Calculate the sizes $r_1,\ r_2$ of the windows.
5. Estimate the number of bits N we need to observe.
6. for t from 0 to N
   - if $z_t = 0$
     - $W += X_t \cdot Y_t + (r_1 - X_t)(r_2 - Y_t)$
   - else if $z_t = 1$
     - $W += X_t(r_2 - Y_t) + (r_1 - X_t)Y_t$
   - end if
   - Move window and update $X_t,\ Y_t$

   end for
7. **if** $|W - N \cdot \frac{r_1 \cdot r_2}{2}| > \sqrt{N \cdot r_1 \cdot r_2}$

   output "cipher" otherwise "random".

Fig. 3. Summary of the proposed distinguishing attack.

4 LILI-II

4.1 Description of LILI-II

LILI-II [1] is the successor of the NESSIE candidate stream cipher LILI-128 [19]. Attacks such as [20, 7, 11, 12, 17] on LILI-128 motivated a larger internal state, which is the biggest difference between the two ciphers, LILI-II also use a nonlinear boolean function f_d with 12 input bits instead of 10 as in LILI-128.

Both the members of the LILI family are binary stream cipher that use irregular clocking. They consists of an $LFSR_c$, that via a nonlinear function clocks a second LFSR, called $LFSR_d$, irregularly. The structured can be viewed in Figure 4. LILI-II use a key length of 128 bits, the key is expanded and

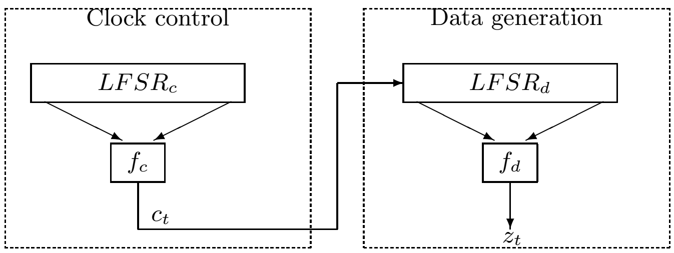

Fig. 4. Overview of LILI keystream generator.

used to initialize the two LFSRs. The first shift register, $LFSR_c$ is a primitive polynomial of length 128, and hence has a period of $2^{128} - 1$. The feedback polynomial for $LFSR_c$ is given by

$$\begin{aligned}
&x^{128} + x^{126} + x^{125} + x^{124} + x^{123} + x^{122} + x^{119} + x^{117} + x^{115} + x^{111} + x^{108} + \\
&x^{106} + x^{105} + x^{104} + x^{103} + x^{102} + x^{96} + x^{94} + x^{90} + x^{87} + x^{82} + x^{81} + \\
&x^{80} + x^{79} + x^{77} + x^{74} + x^{73} + x^{72} + x^{71} + x^{70} + x^{67} + x^{66} + x^{65} + \\
&x^{61} + x^{60} + x^{58} + x^{57} + x^{56} + x^{55} + x^{53} + x^{52} + x^{51} + x^{50} + x^{49} + \\
&x^{47} + x^{44} + x^{43} + x^{40} + x^{39} + x^{36} + x^{35} + x^{30} + x^{29} + x^{25} + x^{23} + \\
&x^{18} + x^{17} + x^{16} + x^{15} + x^{14} + x^{11} + x^{9} + x^{8} + x^{7} + x^{6} + x^{1} + 1.
\end{aligned}$$

The Boolean function f_c takes two input bits from $LFSR_c$, it is chosen as

$$f_c(x_0, x_{126}) = 2 \cdot x_0 + x_{126} + 1\ . \qquad (6)$$

The output of this function is used to clock $LFSR_d$ irregularly. The output sequence from f_c is denoted c_t and $c_t \in \{1, 2, 3, 4\}$, i.e., $LFSR_d$ is clocked at least once and at most four times between consecutive outputs. On average, $LFSR_d$ is clocked $\bar{c} = 2.5$ times.

$LFSR_d$ is chosen to have a primitive polynomial of length 127 which produces a maximal-length sequence with a period of $P_d = 2^{127} - 1$. The original polynomial was found not to be primitive, see [26], and has hence been changed into

$$x^{127} + x^{121} + x^{120} + x^{114} + x^{107} + x^{106} + x^{103} + x^{101} + x^{97} + x^{96} + x^{94} + x^{92} + x^{89} + x^{87} + x^{84} + x^{83} + x^{81} + x^{76} + x^{75} + x^{74} + x^{72} + x^{69} + x^{68} + x^{65} + x^{64} + x^{62} + x^{59} + x^{57} + x^{56} + x^{54} + x^{52} + x^{50} + x^{48} + x^{46} + x^{45} + x^{43} + x^{40} + x^{39} + x^{37} + x^{36} + x^{35} + x^{30} + x^{29} + x^{28} + x^{27} + x^{25} + x^{23} + x^{22} + x^{21} + x^{20} + x^{19} + x^{18} + x^{14} + x^{10} + x^{8} + x^{7} + x^{6} + x^{4} + x^{3} + x^{2} + 1$$

Twelve bits are taken from $LFSR_d$ as input to the function f_d, these bits are taken from the positions (0,1,2,7,12,20,30,44,65,80,96,122) of the LFSR. The function f_d is given as a truth table, note that also the boolean function described in the original proposal was weak and has been replaced, see [26].

4.2 Attack Applied on LILI-II

Low Weight Multiple: According to [23] weight three multiples will start to appear at the degree 2^{64}, since the original shift register has degree 127. The complexity to find the multiple is $O(2^{70})$.

If we instead would mount the attack with a weight four multiple the expected degree of the polynomial would be $2^{43.19}$, the complexity to find a weight four multiple is $O(2^{91.81})$.

Correlation property of f_d: In Table 1 some examples are presented from two clock controlled ciphers, these results are based on a weight three and a weight four recursion. In the case of LILI-128 and LILI-II the correlation property

Generator	Number of input bits	Bias	
		$w = 3$	$w = 4$
LILI-128	10	$2^{-9.00}$	$2^{-9.07}$
LILI-II	12	$2^{-13.22}$	$2^{-12.36}$

Table 1. The correlation property of boolean functions of some clock controlled generators, using weight three and weight four recursions.

of f_d are approximately the same for a weight three relation as for a weight four relation. When using multiples of higher weight than four the correlation property of the functions decreases significantly.

Position of the Windows: When trying to find a multiple of weight three for $LFSR_d$ in LILI-II, we expect the degree of the recurrence to be 2^{64}, i.e., $\tau_1 \approx 2^{63}$ and $\tau_2 \approx 2^{64}$, and hence $\tau_2 - \tau_1 \approx 2^{63}$. The output sequence from the clock-control part denoted by c_t in Figure 4 takes the values $c_t \in \{1, 2, 3, 4\}$ with equal probability, i.e., a geometric distribution. Thus in the case of LILI-II we know that $E(C) = 2.5$ and $\sigma_c = \sqrt{7.5}$. The center positions of the windows will be positioned approximately at $t - 2^{61.68}$ and $t + 2^{61.68}$, where t denotes the position of the center symbol in the recurrence.

Determine the Size of the Windows: As stated in Section 4.2 we know that $E(C) = 2.5$ and $\sigma_c = \sqrt{7.5}$ for LILI-II. We will use a window size of four standard deviations, i.e., $r = 4\sqrt{7.5 \cdot n}$.

Using the expected positions of the windows for LILI-II from previous section the expected window sizes for a weight three relation are $r = 4\sqrt{7.5 \cdot 2^{61.68}} = 2^{34.29}$.

Estimate the Number of Bits We Need to Observe: If we use the estimated numbers from the previous section and Equation (5) we get the following estimate on the number of bits we need to observe to distinguish LILI-II from a random source. For $w = 3$,

$$N \approx \frac{2^{34.29 \cdot 2}}{2^{(-13.22) \cdot 2} \cdot 2^{(-4) \cdot 2}} \approx 2^{103.02}.$$

5 Simulations on a scaled down version of LILI-II

To verify the correctness of the attack we performed the attack on a scaled down versions of LILI-II. In the scaled down version we kept the original clock control part unchanged, but used a weaker data generation part. Instead of the original $LFSR_d$ we used the primitive trinomial,

$$x^{3660} + x^{1637} + 1.$$

We fix the center member of the feedback polynomial, and the center position for window one will be positioned at $t - \tau_1/E(C) = t - \frac{3660-1637}{2.5} = t - 809$, and at $t + \frac{\tau_2-\tau_1}{E(C)} = t - \frac{1637}{2.5} = t + 655$ for window two. We use window sizes of four standard deviations, i.e., $r_1 = 4\sqrt{7.5 \cdot 809} = 312$ and $r_2 = 4\sqrt{7.5 \cdot 655} = 280$.

The boolean function f_d was replaced with the 3-resilient 7-input plateaued function also used in [27],

$$f_d(x) = 1 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_1x_7 + x_2(x_3 + x_7) + x_1x_2(x_3 + x_6 + x_7).$$

The bias of this boolean function for a weight three relation is $\varepsilon_{f_d} = 2^{-4}$. For a weight three relation the probability that all bits are included in the keystream is $p_{dec} = 2^{-4}$. The number of bits we need to observe can now be estimated as

$$N \approx \frac{r_1 \cdot r_2}{\varepsilon_{f_d}^2 \cdot p_{dec}^2} = 2^{33}.$$

We used $N = 2^{36.8054}$ in our simulated attack. The number of combinations fulfilling the recurrence equation, when simulating the attack was

$$W = 2^{52.2201} - 2^{29.2829},$$

where $2^{52.2201}$ is half of the total number of collected samples. This gives a deviation of $2^{29.2829}$ from the expected value of a random sequence and hence a simulated value of $\varepsilon_{tot} = 2^{-23.9372}$. This can be compared with the theoretically derived value which is $\varepsilon_{tot} = 2^{-24.4147}$. The standard deviation can be calculated as $\sigma = \sqrt{N\frac{r_1 r_2}{4}} = 2^{25.6101}$. We reliably distinguish the cipher from a random source.

To verify the expression on the variance (Appendix A) we also performed the attack on a random sequence of bits. The results matched the theory well.

6 Results

In this section we summarize the results of the attack applied on LILI-II, we also show the results for the attack if performed on LILI-128. Observe that there exist many better attacks on LILI-128. These attacks all use the fact that one of the LFSRs only has degree 39, if this degree would be increased the attacks would become significantly less effective, the complexity of our attack would not be affected at all.

In Table 2 we list the sizes on the windows used to attack the generator and the total number of keystream bits we need to observe to reliably distinguish the ciphers from a random source. The results in the table is calculated for a weight three recurrence relation.

Function	r_1	r_2	$\sharp$ bits needed
LILI-128	$2^{25.45}$	$2^{25.45}$	$2^{76.95}$
LILI-II	$2^{34.29}$	$2^{34.29}$	$2^{103.02}$

Table 2. The number of bits needed for the distinguisher for two members of the LILI family.

7 Conclusion

In this paper we have described a distinguisher applicable to irregularly clocked stream ciphers. The attack has been applied on a member of the LILI family, namely LILI-II. The attack on LILI-II needed 2^{103} bits of keystream and a computational complexity of approximately 2^{103} operations to reliably distinguish the cipher from random data. This is the best known attack of this kind so far.

Acknowledgment We thank the anonymous reviewer who pointed out a small improvement in our approach.

References

1. A. Clark, E. Dawson, J. Fuller, J. Golic, H-J. Lee, W. Millan, S-J. Moon, and L. Simpson. The LILI-II keystream generator. In L. Batten and J. Seberry, editors, *Information Security and Privacy: 7th Australasian Conference, ACISP 2002*, volume 2384 of *Lecture Notes in Computer Science*, pages 25–39. Springer-Verlag, 2002.
2. J. Daemen and V. Rijmen. *The Design of Rijndael*. Springer-Verlag, 2002.
3. T. Siegenthaler. Correlation-immunity of non-linear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, 30:776–780, 1984.
4. W. Meier and O. Staffelbach. Fast correlation attacks on stream ciphers. In C.G. Günter, editor, *Advances in Cryptology—EUROCRYPT'88*, volume 330 of *Lecture Notes in Computer Science*, pages 301–316. Springer-Verlag, 1988.
5. A. Canteaut and M. Trabbia. Improved fast correlation attacks using parity-check equations of weight 4 and 5. In B. Preneel, editor, *Advances in Cryptology—EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 573–588. Springer-Verlag, 2000.
6. V. Chepyzhov, T. Johansson, and B. Smeets. A simple algorithm for fast correlation attacks on stream ciphers. In B. Schneier, editor, *Fast Software Encryption 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 181–195. Springer-Verlag, 2000.
7. T. Johansson and F. Jönsson. A fast correlation attack on LILI-128. In *Information Processing Letters*, volume 81, pages 127–132, 2002.
8. T. Johansson and F. Jönsson. Fast correlation attacks through reconstruction of linear polynomials. In M. Bellare, editor, *Advances in Cryptology—CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 300–315. Springer-Verlag, 2000.
9. T. Johansson and F. Jönsson. Fast correlation attacks based on turbo code techniques. In M.J. Wiener, editor, *Advances in Cryptology—CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 181–197. Springer-Verlag, 1999.
10. T. Johansson and F. Jönsson. Improved fast correlation attacks on stream ciphers via convolutional codes. In J. Stern, editor, *Advances in Cryptology—EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 347–362. Springer-Verlag, 1999.
11. N. Courtois and WS. Meier. Algebraic attacks on stream ciphers with linear feedback. In E. Biham, editor, *Advances in Cryptology—EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer-Verlag, 2003.
12. N. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In D. Boneh, editor, *Advances in Cryptology—CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 176–194. Springer-Verlag, 2003.
13. P. Ekdahl and T. Johansson. Distinguishing attacks on SOBER-t16 and SOBER-t32. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 210–224. Springer-Verlag, 2002.

14. J.D. Golić and R. Menicocci. A new statistical distinguisher for the shrinking generator. Available at *http://eprint.iacr.org/2003/041*, Accessed September 29, 2003, 2003.
15. P. Junod. On the optimality of linear, differential and sequential distinguishers. In *Advances in Cryptology—EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 17–32. Springer-Verlag, 2003.
16. D. Watanabe, A. Biryukov, and C. De Canniere. A distinguishing attack of SNOW 2.0 with linear masking method. In *Selected Areas in Cryptography—SAC 2003*, To be published in Lecture Notes in Computer Science. Springer-Verlag, 2003.
17. H. Englund and T. Johansson. A new simple technique to attack filter generators and related ciphers. In *Selected Areas in Cryptography—SAC 2004*, Lecture Notes in Computer Science. Springer-Verlag, 2004.
18. NESSIE. New European Schemes for Signatures, Integrity, and Encryption. Available at *http://www.cryptonessie.org*, Accessed November 10 , 2004, 1999.
19. A. Clark, E. Dawson, J. Fuller, J. Golic, H-J. Lee, William Millan, S-J. Moon, and L. Simpson. The LILI-128 keystream generator. In *Selected Areas in Cryptography—SAC 2000*, volume 2012 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.
20. H. Molland and T. Helleseth. An improved correlation attack against irregular clocked and filtered keystream generators. In *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 373–389. Springer-Verlag, 2004.
21. J.D. Golić and L. O'Connor. A unified markow approach to differential and linear cryptanalysis. In *Advances in Cryptology—ASIACRYPT'94*, Lecture Notes in Computer Science, pages 387–397. Springer-Verlag, 1994.
22. J.D. Golić. Towards fast correlation attacks on irregularly clocked shift registers. In L.C. Guillou and J-J. Quisquater, editors, *Advances in Cryptology—EUROCRYPT'95*, volume 921 of *Lecture Notes in Computer Science*, pages 248–262. Springer-Verlag, 1995.
23. J.D. Golić. Computation of low-weight parity-check polynomials. *Electronic Letters*, 32(21):1981–1982, October 1996.
24. D. Wagner. A generalized birthday problem. In M. Yung, editor, *Advances in Cryptology—CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer-Verlag, 2002.
25. D. Coppersmith, S. Halevi, and C.S. Jutla. Cryptanalysis of stream ciphers with linear masking. In M. Yung, editor, *Advances in Cryptology—CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 515–532. Springer-Verlag, 2002.
26. LILI-II design. Available at *http://www.isrc.qut.edu.au/resource/lili/lili2design.php*, Accessed November 10, 2004, 2004.
27. S. Leveiller, G. Zémor, P. Guillot, and J. Boutros. A new cryptanalytic attack for pn-generators filtered by a boolean function. In K. Nyberg and H. Heys, editors, *Selected Areas in Cryptography—SAC 2002*, volume 2595 of *Lecture Notes in Computer Science*, pages 232–249. Springer-Verlag, 2003.

[0] The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability

A Variance of the number of combinations

Let X_t denote the number of zeros in window one and similarly Y_t denotes the number of zeros in window two at the time t. r_1, r_2 denotes the sizes of windows.

$$
\begin{array}{ccc}
E(X_t) = \frac{r_1}{2} & E(X_t^2) = \frac{r_1^2+r_1}{4} & V(X_t) = \frac{r_1}{4} \\
E(Y_t) = \frac{r_2}{2} & E(Y_t^2) = \frac{r_2^2+r_2}{4} & V(Y_t) = \frac{r_2}{4}
\end{array}
$$

Let Z_t denote the bit in the center position at time t, and W_t' the number of samples fulfilling the recurrence relation at time t. To make the computations a bit simpler we denote $W_t = W_t' - \frac{r_1 r_2}{2}$, i.e., we subtract the expected value of W_t', hence $E(W_t) = 0$. We also introduce the symbol $A_t = X_t Y_t + (r_1 - X_t)(r_2 - Y_t) - r_1 r_2/2$.

$$
W_t = \begin{cases} \underbrace{X_t Y_t + (r_1 - X_t)(r_2 - Y_t) - r_1 r_2/2}_{A_t} & \text{if } Z_t = 0, \\ -\underbrace{\big(X_t Y_t + (r_1 - X_t)(r_2 - Y_t) - r_1 r_2/2\big)}_{A_t} & \text{if } Z_t = 1. \end{cases}
$$

We define W as the sum of W_t for N bits, $W = \sum_{t=0}^{N-1} W_t$. Hence

$$
E(W) = E(\sum_{t=0}^{N-1} W_t) = \sum_{t=0}^{N-1} E(W_t) = 0.
$$

We are trying to calculate $V(\sum_{i=0}^{N-1} W_t') = V(\sum_{i=0}^{N-1} W_t + N\frac{r_1 r_2}{2}) = V(\sum_{i=0}^{N-1} W_t) = V(W)$.

$$
E(W^2) = E\big((\sum_{t_1=0}^{N-1} W_{t_1})(\sum_{t_2=0}^{N-1} W_{t_2})\big) = \sum_{t_1=0}^{N-1}\sum_{t_2=0}^{N-1} E(W_{t_1} \cdot W_{t_2})
$$

– For $t_1 \neq t_2$

$$
\begin{aligned}
E(W_{t_1} W_{t_2}) = \tfrac{1}{4}\Big(& E(W_{t_1} W_{t_2} | Z_{t_1} = 0, Z_{t_2} = 0) + E(W_{t_1} W_{t_2} | Z_{t_1} = 0, Z_{t_2} = 1) + \\
& + E(W_{t_1} W_{t_2} | Z_{t_1} = 1, Z_{t_2} = 0) + E(W_{t_1} W_{t_2}) | Z_{t_1} = 1, Z_{t_2} = 1)\Big) = \\
= & \tfrac{1}{4} E\big(A_{t_1} A_{t_2} - A_{t_1} A_{t_2} - A_{t_1} A_{t_2} + (-A_{t_1})(-A_{t_2})\big) = 0.
\end{aligned}
$$

– For $t_1 = t_2$

$$
\begin{aligned}
E(W_t^2) &= \tfrac{1}{2}\big(E(W_t^2 | Z_t = 0) + E(W_t^2 | Z_t = 1)\big) = E(A^2) = \\
&= 4E(X^2)E(Y^2) + 4r_1 r_2 E(X)E(Y) - 4r_1 E(X)E(Y^2) - 4r_2 E(X^2)E(Y) + \tfrac{r_1^2 r_2^2}{4} - \\
&\quad - r_1^2 r_2 E(Y) - r_1 r_2^2 E(X) + r_1^2 E(Y^2) + r_2^2 E(X) = \\
&= \tfrac{r_1 r_2}{4}
\end{aligned}
$$

So

$$E(W^2) = \sum_{t_1=0}^{N-1} \sum_{t_2=0}^{N-1} E(W_{t_1} W_{t_2}) = \sum_{t=0}^{N-1} E(W_t^2) = \sum_{t=0}^{N-1} \frac{r_1 r_2}{4} = N \cdot \frac{r_1 r_2}{4}.$$

Finally we can give and expression or the variance.

$$V(W) = E(W^2) - E(W)^2 = N \cdot \frac{r_1 r_2}{4}.$$