Open Mathematics

Research Article

Sara D. Cardell*, Amparo Fúster-Sabater, and Adrián H. Ranea

Linearity in decimation-based generators: an improved cryptanalysis on the shrinking generator

https://doi.org/10.1515/math-2018-0058
Received October 9, 2017; accepted January 23, 2018.

Abstract: Decimation-based sequence generators are a class of non-linear cryptographic generators designed to be used in hardware implementations. An inherent characteristic of such generators is that their output sequences are interleaved sequences. This profitable characteristic can be used in the cryptanalysis of those generators. In this work, emphasis is on the most representative decimation-based generator, the shrinking generator, which has been cryptanalyzed just by solving linear equation systems. Compared with previous cryptanalysis, computational complexity and intercepted sequence requirements are dramatically reduced. Although irregularly decimated generators have been conceived and designed as non-linear sequence generators, in practice they can be easily analyzed in terms of simple linear structures.

Keywords: Decimation, Shrinking generator, Interleaved sequence, Stream cipher, Secret-key cryptography

MSC: 94A55, 94A60

1 Introduction

Nowadays stream ciphers are the fastest among the encryption procedures. They are designed to generate, from a short key, a long sequence (*keystream sequence*) of seemingly random bits. Some well known designs in stream ciphers can be found in [1, 2]. Typically, a stream cipher consists of a keystream generator whose output sequence is bit-wise XORed with the plaintext (in emission) to obtain the ciphertext or with the ciphertext (in reception) to recover the original plaintext. References [3–5] provide a solid introduction to the study of stream ciphers.

There are many proposals of keystream generators that are based on maximal-length Linear Feedback Shift Registers (LFSRs) [6]. Such registers are linear structures characterized by their length L, their characteristic polynomial $p(x)$ and their initial state *is* (currently the key of the cryptosystem). Their output sequences, the so-called PN-sequences, are usually combined in a non-linear way in order to break their linearity and to produce new pseudorandom sequences of cryptographic application. LFSRs with dynamic feedback, clock-controlled generators, nonlinear filters or irregularly decimated generators are just some of the most popular keystream generators, see above references.

***Corresponding Author: Sara D. Cardell:** Instituto de Matemática, Estatística e Computação Científica, Universidade Estadual de Campinas, Brazil, E-mail: sdcardell@ime.unicamp.br
Amparo Fúster-Sabater: Instituto de Tecnologías Físicas y de la Información, Consejo Superior de Investigaciones Científicas (CSIC), Spain, E-mail: amparo@iec.csic.es
Adrián H. Ranea: Instituto de Tecnologías Físicas y de la Información, Consejo Superior de Investigaciones Científicas (CSIC), Spain

Irregularly decimated generators produce sequences with good cryptographic properties: long periods, right correlation, excellent run distribution, balancedness, simplicity of implementation, etc. The underlying idea of this kind of generators is the irregular decimation of a PN-sequence according to the bits of another one. The result of this decimation is a binary sequence that will be used as keystream sequence in the cryptographic procedure of stream cipher.

Inside the family of irregularly decimated generators, we can enumerate:

1. The shrinking generator proposed by Coppersmith, Krawczyk and Mansour [7] that involves two LFSRs.
2. The self-shrinking generator designed by Meier and Staffelbach [8] involving only one LFSR.
3. The generalized self-shrinking generator proposed by Hu and Xiao [9] that generates a family of binary sequences.
4. The modified self-shrinking generator, a decimation-based keystream sequence generator, introduced by Kanso in [10] as an improved version of the self-shrinking generator.

In addition, different linear structures based in Cellular Automata that model such generators can also be found in the literature [11–13].

This work focuses on the most representative element in the class of decimation-based sequence generators: the shrinking generator. Taking advantage of the fact that its output sequence is an interleaved sequence, a simple cryptanalytic attack has been developed. The basic ideas of this attack can be generalized to other elements in the same class of generators.

The paper is organized as follows: in Section 2 fundamentals and basic concepts are provided. In Section 3, we introduce some important properties of the shrinking generator that will be used in Section 4 to perform a recovering algorithm for the generated sequence. Section 5 compares the attack here presented with other ones found in the literature. Finally, conclusions in Section 6 end the paper.

2 Preliminaries

Notation and basic concepts are now introduced. First of all, we introduce the concept of decimation, which will be used repeatedly throughout this paper. Let $\{u_i\}$ $(i = 0, 1, 2, \dots)$ be a linear recursive sequence over a finite field. The **decimation** of the sequence $\{u_i\}$ by d is a new sequence obtained by taking every d-th term of $\{u_i\}$ [14].

Next, the definition of interleaved sequence is provided [15].

Definition 2.1. *Let $g(x)$ be a polynomial of degree r over $GF(q)$ (the Galois field of q elements) and let n be a positive integer. For any sequence $\mathbf{w} = \{w_k\}$ over $GF(q)$, write $k = i\,n + j$ $(i = 0, 1, 2, \dots \;\; j = 0, \dots, n-1)$. If all the subsequences $\mathbf{w}_j = \{w_{i\,n+j}\}_{i\geq 0}$ $(j = 0, \dots, n-1)$ are generated by $g(x)$, then $\mathbf{w}$ is called an interleaved sequence over $GF(q)$ of size n associated with $g(x)$.*

We can write $\mathbf{w} = (\mathbf{w}_0, \mathbf{w}_1, \dots, \mathbf{w}_{n-1})$ where each $\mathbf{w}_j$ $(j = 0, \dots, n-1)$ is a subsequence of $\mathbf{w}$. In fact, each $\mathbf{w}_j$ is an n-decimation of the sequence $\mathbf{w}$ obtained from such a sequence by taking one out of n terms. In the sequel, $GF(q)$ will be the binary field $GF(2)$.

The **shrinking generator** (SG) was first introduced in [7]. It is made up of two maximal-length LFSRs denoted by R_1 and R_2. Let L_1 and L_2 $(L_1 < L_2)$ be the LFSR lengths, the primitive polynomials $p_1(x), p_2(x)$ their characteristic polynomials, and is_1 and is_2 their initial states, respectively. Moreover, let $\{a_i\}$ and $\{b_i\}$ be the PN-sequences generated by R_1 and R_2, respectively. In this case, the sequence $\{a_i\}$ decimates the other sequence $\{b_i\}$. The decimation rule is very simple: given two bits a_i and b_i, the output sequence of the generator $\{s_k\}$ is computed as

$$\begin{cases} \text{If } a_i = 1 \text{ then } s_k = b_i \\ \text{If } a_i = 0 \text{ then } b_i \text{ is discarded.} \end{cases}$$

We call the sequence $\{s_k\}$ as the **shrunken sequence** (SS). Assume that $\gcd(L_1, L_2) = 1$, then the period of SS is $T = 2^{L_1-1}(2^{L_2} - 1)$.

The linear complexity of a sequence, denoted by LC, is defined as the length of the minimum LFSR that generates such a sequence. As $\gcd(L_1, L_2) = 1$, then the linear complexity of the shrunken sequence is given by $L_2 2^{L_1-2} < LC \leq L_2 2^{L_1-1}$. Moreover, its characteristic polynomial is of the form $p(x)^m$ where $p(x)$ is a primitive polynomial of degree L_2 and m an integer satisfying $2^{L_1-2} < m \leq 2^{L_1-1}$.

As usual, the key of this generator is the initial state of the both registers R_1 and R_2.

Next a simple illustrative example is introduced.

Example 2.2. *Consider two LFSRs R_1 and R_2 with lengths $L_1 = 2$ and $L_2 = 3$, characteristic polynomials $p_1(x) = 1 + x + x^2$ and $p_2(x) = 1 + x^2 + x^3$, and initial states $is_1 = (1, 0)$ and $is_2 = (1, 0, 0)$, respectively.*

The shrunken-sequence can be computed as follows:

$$\begin{aligned}
\{a_i\} &: 1\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0\,1 \\
\{b_i\} &: 1\,\cancel{0}\,0\,1\,\cancel{1}\,1\,0\,\cancel{1}\,0\,0\,\cancel{1}\,1\,1\,\cancel{0}\,1\,0\,\cancel{0}\,1\,1\,\cancel{1}\,0 \\
\{s_k\} &: \mathbf{1\quad 0\,1\quad 1\,0\quad 0\,0\quad 1\,1\quad 1\,0\quad 1\,1\quad 0}
\end{aligned}$$

The shrunken sequence $\{s_k\}$ has period 14 and it is easy to check that its characteristic polynomial is $p(x)^2 = (1 + x + x^3)^2$, consequently its linear complexity equals 6.

3 Linear properties of the shrunken sequence

In this section, we highlight some properties of the shrunken sequence, which will be used in the algorithm proposed in Section 4. As before, we consider two LFSRs R_1 and R_2 with lengths L_1 and L_2, characteristic polynomials $p_1(x), p_2(x)$ and initial states is_1 and is_2. In addition, $T_1 = 2^{L_1} - 1$ and $T_2 = 2^{L_2} - 1$ are the periods of their corresponding PN-sequences $\{a_i\}$ and $\{b_i\}$, respectively.

According to Definition 2.1, the shrunken sequence $\mathbf{s} = \{s_k\}$ can be written as $\mathbf{s} = \{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{n-1}\}$ where $n = 2^{L_1-1}$. In fact, every subsequence $\mathbf{s}_j$ $(j = 0, \ldots, n-1)$ is a PN-sequence generated by the L_2-degree primitive polynomial $p(x)$ defined as

$$p(x) = \prod_{i=0}^{L_2-1} (x + \alpha^{e_i}),$$

where $e_i = 2^i \cdot T_1 \bmod T_2$ and α is a root of the polynomial $p_2(x)$. Recall that every subsequence $\mathbf{s}_j$ is just a decimation of $\{b_i\}$ by $d = 2^{L_1} - 1$, thus the resulting sequence is a PN-sequence too. In brief, e_i $(i = 0, \ldots, L_2 - 1)$ are the elements of the cyclotomic coset $2^{L_1} - 1$ and $p(x)$ is the polynomial associated with such a coset [6]. The subsequences $\mathbf{s}_j$ $(j = 0, \ldots, n-1)$ are called the **interleaved PN-sequences** of the shrunken sequence.

Example 3.1. *Consider two LFSRs R_1 and R_2 with lengths $L_1 = 3$ and $L_2 = 4$, characteristic polynomials $p_1(x) = 1 + x + x^3$ and $p_2(x) = 1 + x + x^4$ and initial states $is_1 = (1, 0, 0)$ and $is_2 = (1, 0, 0, 0)$, respectively. The shrunken sequence has period $T = 60$ and its characteristic polynomial is $p(x)^4 = (1 + x^3 + x^4)^4$. Since the shrunken sequence is an interleaved sequence, it is composed of 4 PN-sequences:*

$$\begin{array}{rcccc}
 & \mathbf{s}_0 & \mathbf{s}_1 & \mathbf{s}_2 & \mathbf{s}_3 \\
 & \downarrow & \downarrow & \downarrow & \downarrow \\
 & 1 & 0 & 0 & 0 \\
 & 1 & 1 & 1 & 1 \\
 & 1 & 0 & 1 & 0 \\
d_3 = 3 \rightarrow & \underline{0} & 0 & 0 & 1 \\
 & 1 & 0 & 0 & 1 \\
d_2 = 5 \rightarrow & \underline{0} & 1 & 1 & 0 \\
 & 1 & 1 & 0 & 0 \\
 & 1 & 1 & 0 & 1 \\
 & 0 & 1 & 0 & 0 \\
d_1 = 9 \rightarrow & \underline{0} & 0 & 1 & 0 \\
 & 1 & 1 & 1 & 0 \\
 & 0 & 0 & 1 & 1 \\
 & 0 & 1 & 1 & 1 \\
 & 0 & 1 & 0 & 1 \\
 & 1 & 0 & 1 & 1
\end{array}$$

All of them have the same characteristic polynomial $p(x) = 1 + x^3 + x^4$, *thus there is a unique PN-sequence but shifted. This shift depends on the positions of the* 1*s in the PN-sequence* $\{a_i\}$.

Let $\{i_0, i_1, \dots, i_{2^{L_1-1}-1}\}$ denote the position of the 2^{L_1-1} ones in the PN-sequence $\{a_i\}$ and let δ be an integer such that $(2^{L_1} - 1)\delta = 1 \bmod (2^{L_2} - 1)$. Let also d_j $(j = 1, 2, \dots, 2^{L_1-1} - 1)$ be the position over $\mathbf{s}_0$ of the first element of each subsequence $\mathbf{s}_j$ $(j = 1, \dots, n-1)$, respectively. If we know such positions d_j over $\mathbf{s}_0$, then we can compute the indices i_j by means of the following expressions:

$$d_j = \delta \cdot i_j \bmod 2^{L_2} - 1, \quad \text{for } j = 1, 2, \dots, 2^{L_1-1} - 1. \tag{1}$$

In Example 3.1, we had four interleaved subsequences $\mathbf{s}_0$, $\mathbf{s}_1$, $\mathbf{s}_2$ and $\mathbf{s}_3$. It is easy to check that $d_1 = 9$, $d_2 = 5$ and $d_3 = 3$. In this case, $T_1 = 7$ and $T_2 = 15$, then $\delta = 13$. With this information, we can determine the position of the ones in $\{a_i\}$ ($i_0 = 0$, without loss of generality):

$$13 \cdot i_1 = 9 \bmod 15 \rightarrow i_1 = 3$$
$$13 \cdot i_2 = 5 \bmod 15 \rightarrow i_2 = 5$$
$$13 \cdot i_3 = 3 \bmod 15 \rightarrow i_3 = 6$$

Therefore, the set of indices is given by $\{0, 3, 5, 6\}$ and the PN-sequence $\{a_i\}$ is given by $\{1, 0, 0, 1, 0, 1, 1\}$.

In the algorithm proposed in Section 4, the opposite situation occurs. In that case, we know the position of the ones in the PN-sequence $\{a_i\}$ and we compute the position of the first element of each subsequence $\mathbf{s}_j$ in $\mathbf{s}_0$ by means of the expressions given in (1).

The presence of PN-sequences inside the shrunken sequence reveals severe dependencies among its bits. These linear relationships will be advantageously used in the proposed attack. In fact, given N intercepted bits of this sequence, the goal is to determine the pair of initial states (is_1, is_2) of both registers.

4 Cryptanalytic attack

Prior to the attack's description, the following notation is introduced:

- $is_1 = (a_0, a_1, \dots, a_{L_1-1})$, $\quad is_2 = (b_0, b_1, \dots, b_{L_2-1})$
- $\mathbf{S} = \{s_0, s_1, \dots, s_{N-1}\}$ are the N intercepted bits of the shrunken sequence. Currently, the number N can be written as $N = N_1 + N_2$ where N_1 bits are used to compute the pair (is_1, is_2) while N_2 bits are used to check the correctness of the previous pair.

– δ as before is an integer $\delta \in \{1, 2, 3, \dots, T_2 - 1\}$, such that $T_1\delta = 1 \bmod T_2$.

The N_1 intercepted bits are elements of any interleaved PN-sequence $\mathbf{s}_j$. Nevertheless, in this attack we only focus on the first interleaved PN-sequence $\mathbf{s}_0$. For simplicity it will be denoted by $\{u_i\}$ $(i = 0, 1, \dots, 2^{L_2} - 2)$. According to the properties of the PN-sequences, any term u_k of $\{u_i\}$ can be expressed as a function of the first L_2 bits $(u_0, u_1, \dots, u_{L_2-1})$ by means of the modular expression

$$q(x) = x^k \bmod p(x),$$

where $q(x) = c_{L_2-1}x^{L_2-1} + \dots + c_1x + c_0$ with $c_i \in GF(2)$. Thus,

$$u_k = c_{L_2-1}u_{L_2-1} + \dots + c_1u_1 + c_0u_0.$$

This cryptanalytic attack is based on solving systems of linear equations of the form:

$$A\mathbf{x} = \mathbf{b}, \tag{2}$$

where A is an $(N_1 \times L_2)$ binary coefficient matrix, $\mathbf{x}$ is the $(L_2 \times 1)$ vector of unknowns and $\mathbf{b}$ is the $(L_2 \times 1)$ right side vector of intercepted bits. Each initial state is_1 parametrises the coefficient matrix A, then the Linear Consistency Test (LCT) [16] checks the consistency of the corresponding equation system (2). If is_1 considered is the right initial state, then the equation system certainly will be consistent. On the other hand, if is_1 is not the initial state used in the generation of the intercepted bits, then by [16, Theorem 1] the consistency probability of the system will be very small when the intercepted segment is long enough. In order to make the number of false consistency alarms as small as possible, the number of equations in (2) should exceed $L_1 + L_2$ significantly, see [16] and [17].

The attack is divided into two phases. In phase 1, we check the 2^{L_1-1} initial states is_1 starting by 1 (as only the 1s of $\{a_i\}$ generate bits in the shrunken sequence) to determine a set Q of possible candidates to initial state of R_1. In phase 2, for every is_1 in Q its corresponding is_2 will be computed. The pair (is_1, is_2) able to generate all the intercepted shrunken sequence will be the key of the cryptosystem. In brief, the algorithm can be described as follows:

INPUT : The lengths L_1 and L_2 of both registers, the characteristic polynomials $p_1(x), p_2(x)$ and the N intercepted bits $\mathbf{S} = \{s_0, s_1, \dots, s_{N-1}\}$ of the shrunken sequence.
1. Computation of *PHASE 1*
2. Computation of *PHASE 2*

OUTPUT : The initial states is_1 and is_2 (key of the cryptosystem) that generate the shrunken sequence.

In the sequel, the whole attack is described in detail.

PHASE 1:

For each is_1 considered **do**:

1. Starting in is_1, generate a portion of sequence $\{a_i\}$ until N_1 ones are obtained. Such ones will be located at positions i_k $(k = 0, 1, \dots, N_1 - 1)$ over $\{a_i\}$.
2. Determine N_1 positions in the sequence $\{u_i\}$ as
$$d_k = \delta \cdot i_k \bmod T_2 \quad (k = 0, 1, \dots, N_1 - 1).$$
3. Assign the N_1 intercepted bits to the previous positions
$$u_{d_k} = s_k \quad (k = 0, 1, \dots, N_1 - 1).$$
4. Express each u_{d_k} as a function of the first L_2 terms of $\{u_i\}$, that is $u_{d_k} = f_k(u_0, u_1, \dots, u_{L_2-1})$, by means of
$$x^{d_k} \bmod p(x) \quad (k = 0, 1, \dots, N_1 - 1).$$

It turns out to be a system of linear equations

$$\left\{u_{d_k} = f_k(u_0, u_1, \ldots, u_{L_2-1}) = s_k\right.$$

$(k = 0, 1, \ldots, N_1 - 1)$ with N_1 equations in the $(u_0, u_1, \ldots, u_{L_2-1})$ unknowns.

5. Apply the Linear Consistency Test (LCT) [16] to check the consistency of the previous system,
   if the system is consistent, **then** include is_1 in Q
   else is_1 is rejected.

end do

The result of this phase is the set Q of possible candidates to initial state of LFSR R_1. Once the set Q has been computed, the second step of the attack is performed.

PHASE 2:

For each is_1 in Q **do**:

1. Express each b_{i_k} as a function of the first L_2 terms of $\{b_i\}$, that is $b_{i_k} = g_k(b_0, b_1, \ldots, b_{L_2-1})$, by means of
   $$x^{i_k} \bmod p_2(x) \quad (k = 0, 1, \ldots, N_1 - 1).$$
   It turns out to be a system of linear equations
   $$\left\{b_{i_k} = g_k(b_0, b_1, \ldots, b_{L_2-1}) = s_k\right.$$
   $(k = 0, 1, \ldots, N_1 - 1)$ with N_1 equations in the $(b_0, b_1, \ldots, b_{L_2-1}) = is_2$ unknowns.
2. Apply the Linear Consistency Test (LCT) to check the consistency of the previous system,
   if the system is not consistent, **then** reject (is_1, is_2)
   else if the pair (is_1, is_2) can generate the shrunken sequence by using the N_2 bits for checking,
   then cryptosystem broken !!!
   else is_1 is rejected.

end do

The result of this phase is the pair (is_1, is_2) generating the shrunken sequence, that is the key of the cryptosystem.

A software implementation of the previous attack has been performed on a laptop device with the following specifications:

- Operative system: Arch Linux
- CPU: Dual core Intel Core i7-4510U, Cache 4096 KB, Freq. 3100 MHz
- RAM: 8 GB, Type: DDR3
- Hard Disk: Type SSD, Size 256.1 GB

Some numerical results are depicted in Table 1 where L_1, L_2 are the lengths of registers R_1 and R_2, respectively, T is the period of the corresponding shrunken sequence, N_1 is the number of intercepted bits for computation, $c(Q)$ is the cardinality of Q, that is the number of candidates to initial state of R_1, and t is the running time expressed in seconds. It must be noticed that the period of the shrunken sequence is much greater than the number of intercepted bits needed to successfully run the algorithm within a reasonable time. For our computations, $N_1 = 2 \cdot L_2$ while N_2 is chosen $N_2 = N_1$. In brief, the requirements of intercepted sequence are extremely low. In Table 2, the same results are shown but now the number of intercepted bits N_1 equals L_2. In this case, since N_1 has been reduced, the execution time has been reduced too. Nevertheless, the number of candidates has grown considerably. Table 3 shows the numerical results corresponding to the verification of a unique initial state $is1$ in the phase 1 of the algorithm. Recall that even for large values of L_1 and L_2 the execution time of such routine is very low.

Table 1. Numerical results for the algorithm

L_1	L_2	T	N_1	$c(Q)$	$t(sec)$
4	5	248	10	1	0.0064
5	6	1008	12	1	0.0173
9	10	261888	20	1	0.3856
10	11	1048064	22	1	0.8552
11	12	4193280	24	1	1.8114
12	13	16775168	26	1	4.2623
13	14	67104768	28	1	9.0739
14	15	268427264	30	1	20.0681
15	16	$1.0737{\cdot}10^9$	32	1	44.9963
16	17	$4.2949{\cdot}10^9$	34	2	98.1865
17	18	$1.7180{\cdot}10^{10}$	36	1	217.9489
18	19	$6.8719{\cdot}10^{10}$	38	2	477.1288
19	20	$2.7488{\cdot}10^{11}$	40	1	1092.7125
20	21	$1.0995{\cdot}10^{12}$	42	1	2327.2800
21	22	$4.3980{\cdot}10^{12}$	44	1	4997.0925

Table 2. Numerical results for the algorithm when $N_1 = L_2$

L_1	L_2	T	N_1	$c(Q)$	$t(sec)$
4	5	248	5	5	0.0046
5	6	1008	6	14	0.0099
6	7	4064	7	25	0.0216
7	8	16320	8	46	0.0513
8	9	65408	9	78	0.11969
9	10	261888	10	160	0.2478
10	11	1048064	11	210	0.7123
11	12	4193280	12	708	1.3290
12	13	16775168	13	1183	3.1078
13	14	67104768	14	2227	6.0204
14	15	268427264	15	4494	13.0011
15	16	$1.0737{\cdot}10^9$	16	8710	29.4033
16	17	$4.2949{\cdot}10^9$	17	6183	57.9891
17	18	$1.7180{\cdot}10^{10}$	18	35351	151.4661

The most remarkable features of the proposed attack are:

1. The low amount of intercepted bits needed for its execution. Indeed, $N_1 = n \cdot L_2$, n being a small integer ($n = 2, 3, 4$), and $N_2 \leq N_1$. Thus the amount of sequence required is linear in the length of the register R_2.
2. The running time of the attack is dominated by phase 1 which has a time complexity of $\mathbf{O}(2^{L_1-1} \cdot (N_1 \times L_2)^3)$, that is exponential in L_1 due to the number of is_1 considered and polynomial in L_2. In fact, the work factor needed for each test is that of the Gauss elimination algorithm applied to the augmented matrix $(A, \mathbf{b})$, which is cubic in the dimension of the matrix. In any case, the cubic factor is irrelevant compared with the exponential factor.
3. Both phases 1 and 2 are fully parallelizable and some tweaks can be made to optimize the LCT step.

The program makes use of SageMath, an algebraic computation systems based on Python. In order to handle polynomials over $GF(2)$, SageMath uses the libraries NLT. In order to compute with matrices over $GF(2)$, SageMath uses the libraries M4RI. In the LCT application, the system of equations is transformed into a low reduced echelon form. This step is important in the computation efficiency as the system consistency is reduced to test the existence of a row $(0, 0, \ldots, 0, 1)$ in the coefficient matrix of the system.

Table 3. Numerical results for the verification of one $is1$

L_1	L_2	N_1	$t(sec)$
5	6	12	0.00080
7	8	16	0.00112
10	11	22	0.00169
20	21	42	0.00911
30	31	62	0.01044
40	41	82	0.01980
50	51	102	0.03160
59	60	120	0.03547
60	61	122	0.03794
61	62	124	0.03806
62	63	126	0.04035
63	64	128	0.04108

5 Other attacks over the shrinking generator

Other attacks against the shrinking generator have been designed in the literature. For example, in [18], the authors proposed two fault cryptoanalysis. In that work, the attacker is supposed to have a device implementing the shrinking generator and can use it freely. They also assume that the base and control generators of the shrinking generator output bits according to the uniform distribution over $GF(2)$ and that an attacker can disturb clocking of the device, that is, he can stop the control sequence for a couple of steps, and observe the output of the generator. These attacks require injecting specific faults and restarting the device with partially the same internal state. While injecting such faults is potentially possible, it may require some design faults (so that potentially vulnerable parts of the device were placed on external layers). It shows at least that a careful examination of a chip design might be necessary. Furthermore, on the first cryptanalysis, there exists a probability of false solution and algorithm failure. As a consequence, they have to assume that the number of 0s between two 1s does not exceed a certain parameter $maxzeros$. They proved that the probability of a false result grows rapidly with the assumed length of the gap between the 1s. That is why they assume that the control sequence does not contain a block of more than $maxzeroes$ 0s. Of course, when this assumption is false, the algorithm fails.

Several correlation attacks against the shrinking generator have been proposed too. A correlation attack was proposed in [19] and was experimentally analyzed in [20], where an exhaustive search through all initial states and all possible feedback polynomials of R_2 was performed. Later, in [21] the author presented a reduced complexity correlation attack based on searching for specific subsequences of the keystream sequence, whose complexity and required keystream length are both exponential in the length of R_2.

A few years later, in [22] Golić conducted a probabilistic correlation analysis based on a recursive computation of the posterior probabilities of individual bits of R_2, which revealed the possibility of implementing certain type of fast correlation attacks on the shrinking generator. A novel distinguishing attack was also proposed in [23]. In a subsequent paper [24], the author proposed an improved linear consistency attack based on an exhaustive search through all initial states of R_1.

In [22], the author conjectured that the shrinking generator could be vulnerable against fast correlation attacks that would not require an exhaustive search through all possible initial states. In [25], the authors tried to answer this question with length of R_2 equal to 61 (as suggested in [26]). They claimed that given 140000 keystream bits, the initial state of R_2 with arbitrary weight characteristic polynomial of degree 61 could be recovered with success probability higher than 99% and complexity 2^{56}, which was a good trade-off between these parameters.

In brief, the algorithm here developed presents two main advantages against other proposals. First, compared with other cryptanalytic attacks, the original key of the cryptosystem is always obtained. As pointed in [16], there is a trade-off between the number of equations to consider and the false positive ratio. Nevertheless, in our experiments we consider a minimum number of equations and in most cases only the original key was retrieved. Furthermore, with the knowledge of the LFSRs' parameters the attacker just needs to intercept a part of the keystream sequence and perform the algorithm; our method does not need further assumptions. Second, the results given in Table 1 show that the required keystream length in our algorithm grows linearly in the length of R_2, in contrast with other proposals where the amount of required sequence is exponential in the length of any register.

6 Conclusions

The shrinking generator obtains an implicit non-linearity originated from the decimation process. This process is an attempt to create strong pseudorandom sequences with cryptographically good properties out of weak components. It is proved that the shrunken sequence has a long period, a desirably high linear complexity and good statistical properties. However, the linear properties presented in this work make this generator vulnerable against attacks. This paper presents a cryptanalysis over the shrinking generator based on solving linear systems. Besides, the number of intercepted bits needed to successfully perform the algorithm is substantially lower than the period of the sequence, growing linearly with the length of the register R_2.

Acknowledgement: The work of the first author was supported by FAPESP with process number 2015/07246-0. This research has been partially supported by Ministerio de Economía, Industria y Competitividad (MINECO), Agencia Estatal de Investigación (AEI), and Fondo Europeo de Desarrollo Regional (FEDER, UE) under project COPCIS, reference TIN2017-84844-C2-1-R, and by Comunidad de Madrid (Spain) under project reference S2013/ICE-3095-CIBERDINE-CM, also co-funded by European Union FEDER funds.

References

[1] eSTREAM: the ECRYPT Stream Cipher Project, ECRYPT II, eSTREAM portfolio. [Online]. Available: http://www.ecrypt.eu.org/stream/
[2] Robshaw M., Billiet O., New Stream Cipher Designs: The eSTREAM Finalists, Springer, 2008
[3] Menezes A. J., van Oorschot P. C. , Vanstone S. A., Handbook of Applied Cryptography, Boca Raton, FL: CRC Press, 1996
[4] Paar C., Pelzl J., Understanding Cryptography, Berlin: Springer, 2010
[5] Rueppel R. A., Analysis and Design of Stream Ciphers New York, NY: Springer Verlag, 1986
[6] Golomb S. W., Shift Register-Sequences, Laguna Hill, California: Aegean Park Press, 1982
[7] Coppersmith D., Krawczyk H., Mansour Y., The shrinking generator, Advances in Cryptology – CRYPTO '93, Lecture Notes in Computer Science, Springer-Verlag, 1993, 773, 23–39
[8] Meier W., Staffelbach O., The self-shrinking generator, Advances in Cryptology – EUROCRYPT '94, Lecture Notes in Computer Science, Springer-Verlag, 1994, 950, 205–214
[9] Hu Y., Xiao G., Generalized Self-Shrinking Generator, IEEE Trans. Inf. Theory, 2004, 50(4), 714–719
[10] Kanso A., Modified self-shrinking generator, Computers and Electrical Engineering, 2010, 36(5), 993–1001
[11] Fúster-Sabater A., Caballero-Gil P., Linear solutions for cryptographic nonlinear sequence generators, Physics Letters A, 2007, 369, 432–437

[12] Cardell S. D., Fúster-Sabater A., Modelling the shrinking generator in terms of linear CA, Advances in Mathematics of Communication, 2016, 10(4), 797–809
[13] Cardell S. D., Fúster-Sabater A., Linear models for the self-shrinking generator based on CA, Journal of Cellular Automata, 2016, 11(2-3), 195–211
[14] Duvall P. F., Mortick J. C., Decimation of periodic sequences, SIAM Journal on Applied Mathematics, 1971, 21(3), 367–372
[15] Gong G., Theory and Applications of q-ary Interleaved Sequences, IEEE Trans. Inf. Theory, 1995, 41(2), 400–411
[16] Zeng K., Yang C. H., Rao T. R., On the Linear Consistency Test (LCT) in Cryptanalysis with Applications, Advances in Cryptology – CRYPTO ’89, Lecture Notes in Computer Science, Springer-Verlag, 1990, 435, 164–174.
[17] Boztas S., Alamer A., Statistical dependencies in the Self-Shrinking Generator, Seventh International Workshop on Signal Design and its Applications in Communications, IWSDA 2015, Bengaluru, India, 2015, 42–46.
[18] Gomulkiewicz M., Kutylowski M., Wlaź P., Fault cryptanalysis and the shrinking generator, 5th International Workshop on Experimetal Algorithms (WEA 2006), Lecture Notes in Computer Science, Berlin: Springer-Verlag, 2006, 4007, 61–72.
[19] Golić J. D., Embedding and probabilistic correlation attacks on clock-controlled shift registers, Advances in Cryptology-EUROCRYPT’94, Lecture Notes in Computer Science, Berlin: Springer-Verlag, 1994, 950, 230–243.
[20] Simpson L., Golić J. D., A probabilistic correlation attack on the shrinking generator, ACISP ’98 – Third Australasian Conference on Information Security and Privacy, Lecture Notes in Computer Science, Berlin: Springer-Verlag, 1998, 1438, 147–158.
[21] Johansson T., Reduced complexity correlation attacks on two clock-controlled generators, Advances in Cryptology – ASIACRYPT’98, Lecture Notes in Computer Science, Berlin: Springer-Verlag, 1998, 1514, 342–357
[22] Golić J. D., Correlation analysis of the shrinking generator, Advances in Cryptology-Crypto’2001, Lecture Notes in Computer Science, Berlin: Springer-Verlag, 2001, 2139, 440–457
[23] Ekdahl P., Johansson T., Predicting the shrinking generator with fixed connections, Advances in Cryptology-EUROCRYPT’2003, Lecture Notes in Computer Science, Berlin: Springer-Verlag, 2003, 2656, 330–344
[24] Molland, H., Improved linear consistency attack on irregular clocked keystream generators, Fast Software Encryption-FSE’2004, Lecture Notes in Computer Science, Springer-Verlag, 2004, 3017, 109–126
[25] Zhang B., Wu H., Feng D., Bao F., A fast correlation attack on the shrinking generator, Topics in Cryptology – CT-RSA 2005, Lecture Notes in Computer Science, Berlin: Springer-Verlag, 2005, 537, 72–86
[26] Krawczyk H., The shrinking generator: Some practical considerations, Fast Software Encryption-FSE’94, Lecture Notes in Computer Science, Berlin: Springer-Verlag, 1994, 809, 45–46